**PONTIFICIA UNIVERSIDAD JAVERIANA**

# Prototipo del protocolo NETCONF para la Gestión Integrada de elementos de red

**Adriana Fernanda Amarillo Rojas**

Pontificia Universidad Javeriana
Facultad de Ingeniería
Bogotá, Colombia
2013

# Prototipo del protocolo NETCONF para la Gestión Integrada de elementos de red

**Adriana Fernanda Amarillo Rojas**

Tesis en modalidad Profundización presentada como requisito parcial para optar al título de:
**Magister en Ingeniería Electrónica con énfasis en Telecomunicaciones**

Director:
Ing. LUIS CARLOS TRUJILLO, MsC.

Línea de Investigación:
Redes y Sistemas de Telecomunicaciones

Pontificia Universidad Javeriana
Facultad de Ingeniería
Bogotá, Colombia
2013

*A Dios por permitirme alcanzar las metas importantes que me he propuesto en la vida, a mi mamá por enseñarme con su ejemplo y amor los valores que forman excelentes personas, y a mi hermano Gelberth, que más que un familiar es un amigo, un soporte, un colega.*

# Tabla de contenido


**1. INTRODUCCIÓN** ..... 1
**2. MARCO TEÓRICO** ..... 3
**2.1. NETCONF (Network Configuration protocol)** (9) **y** (10) ..... 3
**2.1.1.1. Definición del protocolo** ..... 3
**2.1.1.2. Arquitectura Conceptual** ..... 4
**2.1.1.3. Descripción general del protocolo** ..... 5
**2.1.1.4. Terminología** ..... 5
**2.1.1.5. Requerimientos de Protocolo de Transporte** ..... 5
**2.1.1.5.1. Operación orientada a la conexión** ..... 6
**2.1.1.5.2. Autenticación, Integridad y confiabilidad** ..... 6
**2.1.1.6. Consideraciones de XML** ..... 7
**2.1.1.6.1. Espacio de nombres** ..... 7
**2.1.1.7. Modelo RPC** ..... 7
**2.1.1.7.1. Elemento RPC** ..... 8
**2.1.1.7.2. Elemento <rpc-reply>** ..... 8
**2.1.1.7.3. Elemento <rpc-error>** ..... 9
**2.1.1.7.4. Elemento <ok>** ..... 10
**2.1.1.8. Operaciones base del protocolo NETCONF** ..... 10
**2.1.1.8.1. <get-config>** ..... 11
**2.1.1.8.2. <edit-config>** ..... 12
**2.1.1.8.3. <copy-config>** ..... 16
**2.1.1.8.4. <delete-config>** ..... 17
**2.1.1.8.5. <lock>** ..... 17
**2.1.1.8.6. <unlock>** ..... 20
**2.1.1.8.7. <get>** ..... 20
**2.1.1.8.8. <close-session>** ..... 21
**2.1.1.8.9. <kill-session>** ..... 22
**2.1.1.9. Capacidades** ..... 23
**2.1.1.9.1. Intercambio de capacidades** ..... 24

**2.1.1.10. Separación de datos de configuración de datos de estado** ..... 25
**2.1.1.11. Listas de errores del protocolo NETCONF** ..... 26
**2.1.1.12. Esquema XML del protocolo de gestión de red NETCONF** ..... 26
**2.1.1.13. Módulo YANG para las operaciones del protocolo NETCONF** ..... 26
**2.1.1.14. Typedefs importados por el módulo YANG** ..... 26
**2.2. YANG Lenguaje de modelado de datos para el protocolo de configuración de red NETCONF** (11) ..... 26
**2.2.1. Definición y descripción general del lenguaje** ..... 26
**2.2.2. Terminología** ..... 29
**2.2.3. Descripción de los componentes básicos para definir un modelo de datos** ..... 29
**2.2.3.1. Módulos y Submódulos** ..... 29
**2.2.3.1.1. Bases de modelado de datos** ..... 30
**2.2.3.1.1.1. Nodos hoja** ..... 30
**2.2.3.1.1.2. Nodos Leaf-List** ..... 30
**2.2.3.1.1.3. Nodos container** ..... 31
**2.2.3.1.1.4. Lista de nodos** ..... 32
**2.2.3.1.1.5. Ejemplo de módulo** ..... 33
**2.2.4. Equipos que actualmente integran el lenguaje de modelado de datos YANG en su sistema operativo** ..... 35
**3. ESPECIFICACIONES** ..... 35
**3.1. Características necesarias de los equipos a gestionar utilizando el prototipo desarrollado** ..... 35
**3.2. Especificaciones del prototipo desarrollado** ..... 36
**3.2.1. Inicio de sesión** ..... 36
**3.2.2. Operaciones de configuración de los equipos de red de datos** ..... 36
**3.2.2.1.** Operación de recuperar toda la configuración del equipo de red de datos ..... 37
**3.2.2.2.** Operación de configuración de una interface física del equipo de red de datos 37
**3.2.3. Estructura de los elementos rpc XML** ..... 37
**3.2.4. Finalización de la sesión netconf** ..... 37
**4. DESARROLLOS** ..... 40
**4.1. Actividades realizadas para la implementación en software de la arquitectura del prototipo de gestión de red de datos.** ..... 41
**4.2. Diagrama de máquina de estados del prototipo de software de gestión.** ..... 45
**4.3. Diagrama de clases del prototipo de software de gestión.** ..... 47
**4.4. Diagramas de secuencia del prototipo de software de gestión.** ..... 50

**4.5. Scripts de preconfiguración de los Routers que se desean gestionar con el prototipo de la plataforma de gestión** ..... 52
**4.6. Implementación topología de red para verificación de funcionalidad del prototipo de software de gestión** ..... 54
**5. ANÁLISIS DE RESULTADOS** ..... 56
**5.1. Pruebas de validación del prototipo de software de gestión** ..... 56
**5.1.1. Visualización ventana principal y ventana de configuración del equipo** ..... 56
**5.1.2. Envío de mensaje Hello para inicio de sesión netconf** ..... 58
**5.1.2.1.** Inicio de sesión con router Cisco ..... 59
**5.1.2.2.** Inicio de sesión con router Juniper ..... 61
**5.1.3. Validación de las RPC de configuración teniendo en cuenta los esquemas de los equipos** ..... 63
**5.1.4. Validación de las RPCs teniendo en cuenta el esquema de modelado de datos yang** 63
**5.1.4.1. Validación RPCs utilizando notepad ++** ..... 64
**5.1.4.1.1.** Validación de la rpc-request get-config ..... 64
**5.1.4.1.2.** Validación de la rpc-reply enviada por el servidor como respuesta a la rpc-request get-config ..... 65
**5.1.4.1.3.** Validación de la rpc-request edit-config ..... 67
**5.1.4.1.4.** Validación de la rpc-reply enviada por el servidor como respuesta a la rpc-request edit-config ..... 69
**5.1.4.1.5.** Validación de la rpc-request closse-session ..... 70
**5.1.4.1.6.** Validación de la rpc-reply enviada por un servidor, como respuesta a una rpc-request close-session ..... 70
**5.1.4.2. Validación RPCs utilizando módulo YANG Parser de java** ..... 71
**5.1.4.2.1.** Validación del rpc-request get-config ..... 71
**5.1.4.2.2.** Validación de la rpc-reply enviada por el servidor como respuesta a la rpc-request get-config ..... 72
**5.1.4.2.3.** Validación de la rpc-request edit-config ..... 73
**5.1.4.2.4.** Validación de la rpc-reply enviada por el servidor como respuesta a la rpc-request edit-config ..... 74
**5.1.4.2.5.** Validación de la rpc-request close-session ..... 75
**5.1.4.2.6.** Validación de la rpc-reply de una solicitud rpc-request closse-session ..... 75
**5.1.5. Envío de operación get-config para la recuperación de datos de configuración del equipo** ..... 77
**5.1.6. Envío de operación edit-config para la configuración de una interfaz física del equipo** 78

**5.1.7. Envío operación close-session para finalizar la sesión netconf entre el cliente y el servidor** ..... 80
**6. CONCLUSIONES** ..... 81
7. Bibliografía ..... 85
**8. ANEXOS** ..... 86
Anexo 1. Terminología NETCONF ..... 86
Anexo 2. Módulo YANG para la capa de operaciones de NETCONF ..... 86
Anexo 3. Typedefs importados por el módulo YANG ..... 86
Anexo 4. Terminología YANG ..... 86
Anexo 5. Mensaje Hello ..... 86
Anexo 6. Mensaje Hello enviado por el servidor Cisco ..... 86
Anexo 7. Mensaje Hello enviado por el servidor Juniper ..... 86
Anexo 8. Operación get-config ..... 86
Anexo 9. Operación edit-config para router Cisco ..... 86
Anexo 10. Operación edit-config para router Juniper ..... 87
Anexo 11. Esquema netconf del router cisco ..... 87
Anexo 12. Esquema jerárquico de las interfaces del router Juniper ..... 87
Anexo 13. Mensaje rpc-reply ..... 87
Anexo 14. Operación close-session ..... 87
Anexo 15. Clase principal del proyecto ..... 87
Anexo 16. Código fuente en java de la clase User Interface ..... 87
Anexo 17. Código fuente en java de la clase Canal SSH ..... 87
Anexo 18. Código fuente en java de la clase Comandos ..... 87
Anexo 19. Código fuente en java de la clase Netconf ..... 87
Anexo 20. Código fuente en java de la clase YANG Parser ..... 87
Anexo 21. Esquema XSD para las operaciones del protocolo de gestión de red NETCONF ..... 87
Anexo 22. Esquema XSD para la capa de mensajes del protocolo NETCONF ..... 87

**LISTA DE FIGURAS**

Figura 1 Arquitectura protocolo de gestión de red NETCONF. Tomado de(10) ..... 4
Figura 2 Atributo de un elemento rpc. Tomado de (10) ..... 8
Figura 3 Mensajes rpc y rpc-reply. Tomado de (10) ..... 9
Figura 4 Mensaje ok. Tomado de (10) ..... 10
Figura 5 Mensaje rpc operación get-config. Tomado de (10) ..... 12
Figura 6 Mensaje rpc operación edit-config. Tomado de (10) ..... 16
Figura 7 Mensajes rpc y rpc-reply de la operación close-session. ..... 22
Figura 8 Mensaje Hello en el que se anuncian distintas capacidades ..... 25
Figura 9. Arquitectura de software del prototipo de la plataforma de gestión de red. Tomado de (17) ..... 38
Figura 10. Diagrama de flujo del módulo Netconf del prototipo de la plataforma de gestión de red de datos. Tomado de (17) ..... 39
Figura 11. Escenario real del sistema de gestión de red desarrollado. Tomado de (17) ..... 40
Figura 12. Inclusión de la librería JSch al classpath de eclipse. Tomado de (17) ..... 42
Figura 13. Diagrama de máquina de estados del prototipo de software de gestión. Tomado de (17) ..... 46
Figura 14. Diagrama de Clases del prototipo de software de gestión. Tokmado de (17) ..... 48
Figura 15. Diagrama de secuencia de inicio de conexión del prototipo de software de gestión. Tomado de (17) ..... 51
Figura 16. Diagrama de secuencia de las operaciones NETCONF del prototipo de software de gestión. Tomado de (17) ..... 52
Figura 17. Topología de la red gestionada utilizando el prototipo de gestión. Tomado de (17) ..... 55
Figura 18. Consola del equipo Cisco donde se visualiza versión IOS del equipo. ..... 55
Figura 19. Consola del equipo Juniper donde se visualiza versión de JUNOS del equipo .. 56
Figura 20. Interfaz gráfica principal para indicar parámetros del dispositivo con el que se desea establecer conexión. Tomado de (17). ..... 56
Figura 21. Mensaje de error de la ventana principal del prototipo de gestión. ..... 57
Figura 22. Ventana de configuración del equipo. Tomado de (17) ..... 58
Figura 23. Intercambio de mensajes ssh V2 durante el inicio de conexión entre el cliente y el router Cisco ..... 59
Figura 24. Consola equipo Cisco ..... 60
Figura 25. Ventana de configuración del equipo en la que la consola le indica al usuario del éxito de la conexión. Tomado de (17) ..... 60
Figura 26. Intercambio de mensajes ssh V2 durante el inicio de conexión entre el cliente y el router Juniper ..... 61
Figura 27. Ventana principal del prototipo de gestión en la que el mensaje de consola indica que se puede realizar la configuración del equipo en la nueva ventana abierta de configuración del equipo. Tomado de (17) ..... 62
Figura 28. Ventana de configuración del equipo en la que la consola le indica al usuario del éxito de la conexión. Tomado de (17) ..... 63
Figura 29. Configuración de la herramienta XML Tool de notepad++ para que utilice el esquema de datos netconf para analizar el contenido de las RPCs ..... 65
Figura 30. Validación de la rpc get-config utilizando la herramienta XML Tool del programa notepad++ ..... 65

Figura 31. Validación con notepad++ de la rpc-reply enviada por un router Cisco, como respuesta a la rpc-request get-config proveniente del cliente. ..... 66
Figura 32. Validación con notepad++ de la rpc-reply enviada por un router Juniper, como respuesta a la rpc-request get-config proveniente del cliente. ..... 67
Figura 33. Validación con notepad++ de la rpc edit-config para un router Cisco ..... 68
Figura 34. Validación con notepad++ de la rpc edit-config para un router Juniper ..... 68
Figura 35. Validación con notepad++ de la rpc-reply enviada por un router Cisco, como respuesta a la rpc-request edit-config proveniente del cliente. ..... 69
Figura 36. Validación con notepad++ de la rpc-reply enviada por un router Juniper, como respuesta a la rpc-request edit-config proveniente del cliente. ..... 69
Figura 37. Validación con notepad++ de la rpc close-session enviada a los servidores Cisco y Juniper. ..... 70
Figura 38. Validación con notepad++ de la rpc-reply enviada por un router Cisco, como respuesta a la rpc-request close-session proveniente del cliente. ..... 70
Figura 39. Validación con notepad++ de la rpc-reply enviada por un router Juniper, como respuesta a la rpc-request close-session proveniente del cliente. ..... 71
Figura 40. Validación con eclipse de la rpc get-config. ..... 72
Figura 41. Validación con eclipse de la rpc-reply enviada por el router Cisco, como respuesta de la rpc-request get-config enviada por el cliente. ..... 72
Figura 42. Validación con eclipse de la rpc-reply enviada por el router Juniper, como respuesta de la rpc-request get-config enviada por el cliente. ..... 73
Figura 43. Validación con eclipse de la rpc edit-config para un router Cisco. ..... 73
Figura 44. Validación con eclipse de la rpc edit-config para un router Juniper. ..... 74
Figura 45. Validación con eclipse de la rpc-reply enviada por el router Cisco, como respuesta de la rpc-request edit-config enviada por el cliente. ..... 74
Figura 46. Validación con eclipse de la rpc-reply enviada por el router Juniper, como respuesta de la rpc-request edit-config enviada por el cliente. ..... 75
Figura 47. Validación con eclipse de la rpc close-session. ..... 75
Figura 48. Validación con eclipse de la rpc-reply enviada por el router Cisco, como respuesta de la rpc-request close-session enviada por el cliente. ..... 76
Figura 49. Validación con eclipse de la rpc-reply enviada por el router Juniper, como respuesta de la rpc-request close-session enviada por el cliente. ..... 76
Figura 50. Ventana de configuración del equipo con resultado de la ejecución de la operación "Obtener toda la configuración" en el equipo Cisco. Tomado de (17) ..... 77
Figura 51. Ventana de configuración del equipo con resultado de la ejecución de la operación "Obtener toda la configuración" en el equipo Juniper. Tomado de (17) ..... 78
Figura 52. Ventana de configuración del equipo con resultado de la ejecución de la operación "Configurar interface" en el equipo Cisco. Tomado de (17) ..... 79
Figura 53. Ventana de configuración del equipo con resultado de la ejecución de la operación "Configurar interface" en el equipo Juniper. Tomado de (17) ..... 79
Figura 54. Visualización de la configuración de la interfaz del Router Cisco configurada con el prototipo de gestion ..... 80
Figura 55. Visualización de la configuración de la interfaz del Router Juniper configurada con el prototipo de gestion. ..... 80
Figura 56. Resultado de la ejecución de la operación Cerrar sesión en el router Cisco. Tomado de (17) ..... 81

Figura 57. Resultado de la ejecución de la operación Cerrar sesión en el router Juniper. Tomado de (17) ..... 81

# 1. INTRODUCCIÓN

Actualmente, las redes de telecomunicaciones son una parte fundamental para la productividad de las economías y sociedades, ya que permiten la intercomunicación entre diferentes organizaciones que interactúan como colaboradores estratégicos, logrando un aumento de la productividad e ingresos de dichas organizaciones (1), (2). Estas redes de telecomunicaciones tienen como uno de sus componentes principales, los dispositivos de transmisión y procesamiento de la información como los son los *Routers* y *Switches*, que generalmente pertenecen a diferentes fabricantes, creando de esta manera un entorno de red multiproveedor (3).

Dado que las implementaciones de redes de telecomunicaciones generalmente están a cargo de un proveedor de servicios, éste debe garantizar la disponibilidad de los recursos y/ó servicios pactados con el cliente, siendo por tanto de vital importancia, la implementación de un sistema de gestión de red de datos, que además de ser capaz de detectar y corregir los posibles fallos de la red (1) mediante su supervisión y establecimiento de parámetros (4), también sea lo más rentable económica y logísticamente posible para los operadores de red que finalmente son los usuarios de estos elementos de red. Aunque lo ideal es tener una única plataforma de gestión de red que satisfaga las anteriores necesidades de los usuarios de estos elementos de red, se tiene que cada proveedor cuenta con su propio sistema de gestión y monitoreo (5), es decir, no existe interoperabilidad de gestión de red entre diferentes equipos de red de diferente proveedor.

En los años 80’s, el principal protocolo de gestión de equipos de red fue SNMP (Simple Network Management Protocol)(1) desarrollado por la IETF (Internet Engineering Task Force) (6), (7), y aunque ha existido todo un proceso de evolución de este protocolo a lo largo de aproximadamente 17 años, no se ha logrado que SNMP se ajuste a las necesidades de escalabilidad para la gestión de red de datos de las redes de Telecomunicaciones emergentes, porque: 1. utiliza UDP (*User Datagram Protocol*) como protocolo de transmisión, lo que se traduce en falta de seguridad en la comunicación entre los elementos de red (no se

establece una conexión segura, dando lugar a posibles ataques informáticos externos), y 2.porque no es de utilidad a la hora de realizar configuraciones especializadas y complejas en los equipos de red, causando que SNMP fuera delegado únicamente para efectos de monitoreo de las redes de Telecomunicaciones.

Dada la falencia de SNMP para efectuar configuraciones especializadas y complejas en los equipos de red, se hizo necesario recurrir a la utilización de CLI (Command Line Interface) para suplir esta necesidad, pero tampoco es una buena opción a tomar, ya que CLI no cuenta con un estándar de gestión de la información, ni cuenta con respuestas estructuradas para errores (8). Así, se observa que la implementación de estas herramientas para el monitoreo y la gestión de una red de telecomunicaciones, no es la mejor opción para los proveedores de servicios de telecomunicaciones, ya que se incrementan los costos de operación y mantenimiento al ser necesarias diversas personas especializadas que ejecuten las operaciones deseadas en los diferentes equipos de red (8), puesto que los equipos de cada proveedor tiene sus propias configuraciones y mecanismos para su gestión (5).

Siendo conscientes de la problemática señalada anteriormente de falta de interoperabilidad de gestión de diferentes equipos de red indistintamente del fabricante, la IETF crea en Diciembre del año 2006 el protocolo de gestión de red NETCONF (Network Configuration Protocolo) bajo el RFC4741 (9), y realiza su revisión en junio de 2011 bajo el RFC6241 (10). También, crea en Octubre del año 2010, el modelado de datos YANG para ser utilizado por el protocolo NETCONF para modelar los datos de estado y configuración del protocolo, para modelar los datos de las RPC (*Remote Procedure Calls*) y para modelar las notificaciones de NETCONF (11).

Así, NETCONF no tiene las deficiencias mencionadas de SNMP y CLI, ya que garantiza la seguridad de la información mediante la utilización de protocolos de transporte de datos seguros como SSH (*Secure Shell*), además de permitir el desarrollo de un sistema de gestión de información que permite la instalación, manipulación, y borrado de la configuración de dispositivos de red sin importar de

qué proveedor sean. También, NETCONF cuenta con respuestas estructuradas para errores y permite la aplicación de capacidades de protocolo (ejemplo, recuperar cierta porción de configuración de un equipo de red), dando la opción de desarrollo de nuevas capacidades aún no estandarizadas, lo que permite la escalabilidad del protocolo y se ajuste a necesidades futuras (10).

De esta manera, en el presente proyecto se muestra el desarrollo para un prototipo de una plataforma de gestión de red de datos que utiliza el protocolo de gestión de red de datos NETCONF y el lenguaje de modelado de datos YANG, ello teniendo en cuenta sus ventajas y características. De esta manera, el software desarrollado permite como primera y única instancia, realizar configuraciones de dos dispositivos de red de datos.

# 2. MARCO TEÓRICO

## 2.1. NETCONF (Network Configuration protocol) (9) y (10)

### 2.1.1.1. Definición del protocolo

NETCONF provee mecanismos para instalar, manipular, y borrar la configuración de dispositivos de red. Utiliza XML (*Extensible Markup Language*) como método de configuración y codificación de datos (12), y para la operación y el control del protocolo entre un gestor y un agente de comunicación, se utiliza el método *Remote Procedure Call* (RPC) (12).

Una característica diferenciadora del protocolo NETCONF frente a otros desarrollados, es que permite a los dispositivos exponer una API (*Application Programming Information*), la cual es utilizada por las aplicaciones para enviar y recibir configuraciones completas o parciales.

Así, se utilizan sesiones para realizar el intercambio de datos de configuraciones de los dispositivos de red. Una sesión NETCONF, es la conexión lógica entre un administrador de red ó una aplicación de configuración de red y un dispositivo de red, el cual debe soportar al menos una sesión NETCONF.

### 2.1.1.2. Arquitectura Conceptual

Conceptualmente, NETCONF puede ser particionado en cuatro capas:

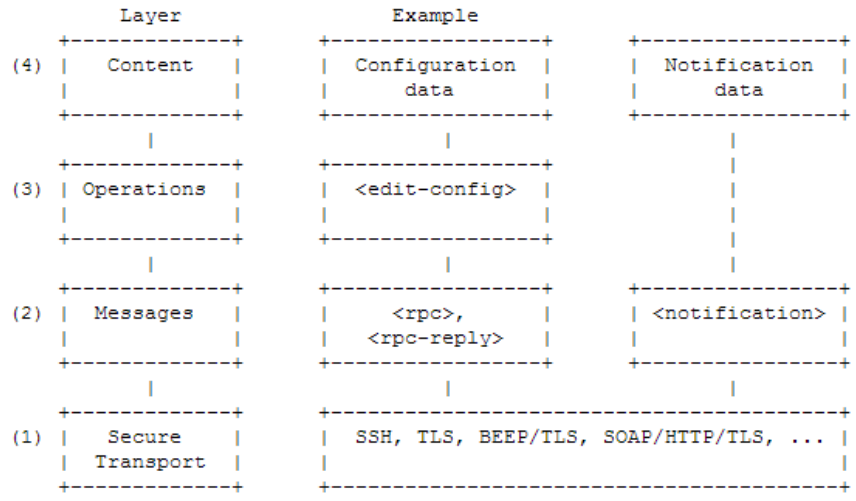


**Figura 1 Arquitectura protocolo de gestión de red NETCONF. Tomado de(10)**

1. La capa segura de trasporte provee un camino de comunicación entre el cliente y el servidor. NETCONF se puede superponer sobre cualquier protocolo de transporte que tenga un conjunto básico de requerimientos (En la sección 2 del RFC 6241 se discuten los requerimientos)
2. La capa de mensajes proporciona un mecanismo simple de estructuración de mensajes de transporte, para la codificación de RPCs y notificaciones. La sección 4 del RFC 6241 documenta los mensajes RPC, y el RFC 5717 documenta las notificaciones.
3. La capa de operaciones define el conjunto de operaciones base del protocolo, invocadas como métodos RPC con parámetros codificados en XML. La sección 7 del RFC 6241 se detalla la lista de las operaciones base del protocolo.
4. La capa de contenido está fuera de alcance del RFC6241

El lenguaje de modelado de datos YANG RFC6020 ha sido desarrollado para especificar los modelos de datos NETCONF y las operaciones de protocolo, cubriendo las capas de Operación y Contenido.

#### 2.1.1.3. Descripción general del protocolo

NETCONF utiliza un mecanismo simple de RPC para facilitar la comunicación entre un cliente y un Servidor. Un Cliente codifica una RPC en XML y la envía a un servidor usando una conexión segura, estableciendo una sesión orientada a la conexión. El servidor responde con una respuesta codificada en XML. El contenido de las peticiones y las respuestas son completamente descritos en XML DTDs ó en esquemas XML, ó en ambos, permitiendo el reconocimiento de las restricciones de sintaxis impuestas en el intercambio.

El cliente puede ser un *script* ó una aplicación corriendo típicamente como parte de un gestor de red. El servidor es típicamente un dispositivo de red. Los términos 'dispositivo' y 'servidor' son usados intercambiablemente en el RFC 6241, así como 'cliente' y 'aplicación'.

Una sesión NETCONF es la conexión lógica entre un administrador de red ó una aplicación de configuración de red y un dispositivo de red. Un dispositivo debe soportar al menos una sesión NETCONF y puede soportar múltiples sesiones. Atributos globales de configuración pueden ser cambiados durante alguna sesión autorizada, y los efectos son visibles en todas las sesiones. Los atributos específicos de una sesión afecta solo a la sesión en la cual fueron cambiados.

#### 2.1.1.4. Terminología

Ver Anexo 1

#### 2.1.1.5. Requerimientos de Protocolo de Transporte

NETCONF utiliza un modelo de comunicación basado en RPC. Un cliente envía series de uno ó más mensajes de petición RPC, causando que el servidor responda con las series correspondientes de mensajes de respuesta RPC.

El protocolo de transporte debe proveer un mecanismo para indicar el tipo de sesión (cliente ó servidor) a la capa de transporte de NETCONF.

#### 2.1.1.5.1. Operación orientada a la conexión

NETCONF está orientado a la conexión, lo que requiere una conexión permanente entre los pares. La entrega de los datos durante una conexión, debe ser confiable y secuencial. Adicionalmente, los recursos solicitados por un servidor para una conexión en particular, deben ser automáticamente liberados cuando se cierra la conexión, haciendo que la recuperación de fallos sea simple y robusta. Por ejemplo, cuando un cliente envía una petición al servidor para que impida que desde otra sesión se puedan realizar cambios en ciertos almacenamientos de datos, el bloqueo persiste hasta que este es explícitamente liberado por el cliente, ó el servidor determina que la conexión ha finalizado.

#### 2.1.1.5.2. Autenticación, Integridad y confiabilidad

Las conexiones NETCONF deben proveer autenticación, integridad de los datos, confidencialidad y protección de repetición. NETCONF depende de la capa de transporte para esta capacidad. Ejemplo: SSH (Secure Shell) RFC4251, TLS (Transport Layer Security) RFC5246.

El protocolo de transporte obligatorio es ssh, esto con el fin de permitir la interoperatividad de gestión, y es el encargado de la autenticación del servidor en el cliente y viceversa. Cada uno de los pares asume que la información de la autenticación de la conexión ha sido validada por el protocolo de transporte, usando mecanismos lo suficientemente confiables, y que la identidad de los pares ha sido lo suficientemente probada.

Un servidor NETCONF debe tener configurado el control de acceso ssh (o de otros protocolos seguros de transporte soportados por el equipo), para que un cliente pueda autenticarse utilizando los parámetros requeridos. Los permisos de acceso otorgados a un cliente, deben ser cumplidos durante toda la sesión NETCONF. Los

detalles de cómo el control de acceso es configurado está fuera del alcance del RFC6241. Una implementación de NETCONF con SSH se evidencia en el RFC6242 (13)

#### 2.1.1.6. Consideraciones de XML

XML sirve como formato de codificación para NETCONF, permitiendo una jerarquización compleja de los datos, los cuales se expresan en formato de texto que pueden ser leidos, guardados, y manipulados, tanto con herramientas tradicionales de texto, como con herramientas específicas para XML.

Todos los mensajes NETCONF deben estar bien formados en XML y codificados en UTF-8 especificado en el RFC3629. Si un par recibe un mensaje <rpc> que no está bien formado en XML ó no está codificado en UTF-8, deberá responder con un mensaje de error de "mensaje malformado". Si la respuesta no es enviada por alguna razón, el servidor debe terminar la sesión.

##### 2.1.1.6.1. Espacio de nombres

Todos los elementos del protocolo NETCONF están definidos en el siguiente espacio de nombres:

urn:ietf:params:xml:ns:netconf:

Los nombres de las capacidades NETCONF deben ser URIs (Identificador de Recurso Uniforme) RFC3986. Las capacidades de NETCONF son discutidas en la sección 8 del RFC6241.

#### 2.1.1.7. Modelo RPC

El protocolo NETCONF utiliza un modelo de comunicación basado en RPCs. Los pares NETCONF utilizan elementos <rpc> y <rpc-reply> como marcos para contener las solicitudes y respuestas NETCONF.

La sintaxis y la codificación XML de los RPC de los mensajes son definidos formalmente en el esquema XML que se encuentra en el apéndice B del RFC6241.

#### 2.1.1.7.1. Elemento RPC

El elemento <rpc> es usado para encerrar las solicitudes NETCONF enviadas de un cliente a un servidor. Este elemento tiene un atributo obligatorio: 'message-id', el cual es un *string* escogido por quien envía el RPC y el cual comúnmente se va incrementando en una unidad. Quien recibe la RPC no debe decodificar ó interpretar este string, sino que simplemente debe guardarlo y usarlo como atributo de 'message-id' en el mensaje <rpc-reply>. Quien envía el RPC debe asegurarse que el valor del 'message-id' está normalizado de acuerdo a las reglas de normalización de los valores de atributo de XML definidas en (14), si es que quiere que sea devuelto sin ser modificado. Ver Figura 2

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <some-method>
    <!-- method parameters here... -->
  </some-method>
</rpc>
```

**Figura 2 Atributo de un elemento rpc. Tomado de (10)**

Si atributos adicionales están presentes en el elemento <rpc>, el par NETCONF debe retornarlos sin realizarles modificaciones en el elemento <rpc-reply>. Esto incluye cualquier atributo "xmlns".

#### 2.1.1.7.2. Elemento <rpc-reply>

El mensaje <rpc-reply> es enviado en respuesta a un mensje <rpc>. El elemento <rpc-reply> tiene un atributo obligatorio: 'message-id', el cual es igual al atributo "message-id" del <rpc> al cual está respondiendo. El servidor NETCONF debe también retornar en el

elemento <rpc-reply>, atributos adicionales incluidos en el elemento <rpc> si ningún tipo de cambio. Los datos de respuesta son codificados como uno o más elementos hijos del elemento <rpc-reply>. Ver Figura 3

```
<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>
```

**Figura 3 Mensajes rpc y rpc-reply. Tomado de (10)**

#### 2.1.1.7.3. Elemento <rpc-error>

El elemento <rpc-error> es enviado en mensajes <rpc-reply> si un error ocurre durante el procesamiento de una solicitud <rpc>. Si un servidor encuentra múltiples errores durante el procesamiento de una solicitud <rpc>, el <rpc-reply> puede contener múltiples elementos <rpc-error>. Sin embargo, un servidor no necesariamente debe detectar o reportar más que un elemento <rpc-error>, si por ejemplo una solicitud contiene múltiples errores.

Un servidor no debe retornar errores de nivel de aplicación ó de errores específicos del modelo de datos a clientes que no tengan suficientes derechos de acceso. El elemento <rpc-error> incluye la siguiente información:

- error-type: define la capa conceptual en la que el error ocurre. Puede ser una de las siguientes:
  - transport (capa: transporte)
  - rpc (capa: mensajes)
  - protocol (capa: operaciones)

- application (capa: contenido)

- error-tag: contiene un string que identifica la condición del error. En el apéndice A del RFC6241 se encuentran los valores permitidos.

#### 2.1.1.7.4. Elemento <ok>

El elemento <ok> es enviado en mensajes <rpc-reply> si no hay errores ó alarmas durante el procesamiento de una solicitud <rpc>, y no se retornan datos de la operación realizada. Ver Figura 4

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

**Figura 4 Mensaje ok. Tomado de (10)**

### 2.1.1.8. Operaciones base del protocolo NETCONF

El protocolo NETCONF provee un pequeño conjunto de operaciones de bajo nivel para gestionar la configuración de dispositivos y recuperar información de estado de estos dispositivos. La base del protocolo provee operaciones para recuperar, configurar, copiar y borrar configuraciones de almacenamientos de datos. Las operaciones adicionales son dadas, basándose en las capacidades anunciadas por el dispositivo. La base del protocolo incluye las siguientes operaciones del protocolo:

- get
- get-config
- edit-config
- copy-config
- delete-config
- lock
- unlock
- close-session
- kill-session

Una operación de protocolo puede fallar por varias razones, entre las que se encuentra que la operación no es soportada. El dispositivo que inicia una solicitud, no debe asumir que toda operación siempre será exitosa. Los valores retornados en cualquier rpc-reply, deben ser revisados para verificar si corresponden a errores.

La sintaxis y codificación XML de las operaciones del protocolo, son formalmente definidas en el módulo YANG en el apéndice C del RFC 6241. Las siguientes secciones describen la semántica de cada operación de protocolo.

**2.1.1.8.1. <get-config>**

Descripción:

Recupera todo o parte de un almacenamiento de configuración especificado. La Figura 5 muestra un ejemplo de rpc de esta operación

Parámetros:

- Source: nombre del almacenamiento de configuración que se consulta, tal como <running/>.
- Filter: este parámetro identifica las porciones de almacenamiento de configuración de un dispositivo a recuperar. Si este parámetro no está presente, toda la configuración es retornada.

  Respuesta positiva: si el dispositivo puede satisfacer la solicitud, el servidor envía un elemento <rcp-reply> en el cual va el elemento <data> con los resultados de la solicitud.

  Respuesta negativa: un elemento <rpc-error> es incluido en el elemento <rpc-reply> si la solicitud no puede ser completada por alguna razón.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <filter type="subtree">
      <top xmlns="http://example.com/schema/1.2/config">
        <users/>
      </top>
    </filter>
  </get-config>
</rpc>
```

**Figura 5 Mensaje rpc operación get-config. Tomado de (10)**

#### 2.1.1.8.2. <edit-config>

Descripción:

La operación <edit-config> carga todo o parte de una configuración especificada a un almacenamiento de configuración especificado.

El dispositivo analiza la fuente y el destino de las configuraciones y desarrolla los cambios solicitados. La configuración del destino no es necesariamente remplazada, como ocurre con el mensaje <copy-config>. En lugar de ello, la configuración destino es cambiada de acuerdo con los datos de origen y las operaciones solicitadas. La Figura 6 muestra un ejemplo de rpc de esta operación.

Atributos:

- Operation: los elementos en el subárbol <config> pueden contener un atributo de "operación", que pertenece al espacio de nombre NETCONF definido en la sección 3.1 del RFC 6241. El atributo identifica el punto en la configuración donde se debe ejecutar la operación, y puede aparecer en múltiples elementos a través de todo el sub árbol <config>.

  Si el atributo "operación" no es especificado, la configuración se combina en el almacenamiento de datos de configuración.

  El atributo "operación" tiene uno de los siguientes valores:

- Merge: los datos de configuración en el parámetro <config>, son fusionados con los datos que están en el nivel correspondiente del almacén de datos destino.

- Replace: los datos de configuración identificados por el elemento que contiene este atributo, remplaza cualquier configuración relacionada en el almacén de datos de configuración identificado por el parámetro <target>. Si no existe tales datos de configuración en el almacenamiento de configuración, los mismos son creados. A diferencia de la operación <copy-config> que remplaza la totalidad de la configuración objetivo, en este caso, solo es afectada la configuración que esté actualmente presente en el parámetro <config> .

- Create: los datos de configuración identificados por el elemento que contiene este atributo, son adicionados a la configuración, si y solo si, los datos de configuración actualmente no existen en el almacenamiento de configuración. Si los datos de configuración existen, un elemento <rpc-error> es retornado con un valor <error-tag> de “datos-existentes”.

- Delete: los datos de configuración identificados por el elemento que contiene este atributo son borrados de la configuración, si y solo si, los datos de configuración actualmente existen en el almacenamiento de configuración. Si los datos de configuración no existen, un elemento <rpc-error> es devuelto con un valor <error-tag> de “datos-faltantes”.

- Remove: los datos de configuración identificados por el elemento que contiene este atributo son borrados de la configuración, si y solo si, los datos de configuración actualmente existen en el almacenamiento de configuración. Si los datos de configuración no existen, la operación "remove" es ignorada por el servidor.

Parámetros:

- Target: nombre del almacenamiento de configuración que se está editando, como por ejemplo <running/> ó <candidate/>.
- Default-operation: selecciona la operación por defecto (como se describe en el parámetro "operación"), para esta solicitud <edit-config>. El valor por defecto para el parámetro <default-operation> es "merge".

  El parámetro <default-operation> es opcional, pero si se proporciona, este tiene uno de los siguientes valores:

  - Merge
  - Replace
  - None
- Test option: el elemento <test-opction> puede ser especificado solo si el dispositivo anuncia la capacidad :validate:1.1. Ver sección 8.6 del RFC 6241.

  El elemento <test-opction> tiene uno de los siguientes valores:

  - test-then-set:
  - set:
  - test-only:

- error-option: el elemento <error-option> tiene uno de los siguientes valores:

- stop-on-error: anula la operación <edit-config> en el primer error. Este es el valor por defecto de error-option.
- continue-on-error: continuación para procesar los datos de configuración en caso de error, se registra el error, y la respuesta negativa es generada si se produce algún error.
- rollback-on-error: si una condición de error ocurre, de tal manera que es generado un elemento de error de severidad <rpc-error>, el servidor parará el procesamiento de la operación <edit-config> y restaurará la configuración especificada a su estado completo e inicial de la operación <edit-config>. Esta opción requiere que el servidor soporte la capacidad :rollback-on-error, descrita en la seción 8.5 del RFC 6241.

▪ Config: es una jerarquía de datos de configuración tal y como se define por uno de los modelos de datos del dispositivo. Los contenidos deben ser colocados en un espacio de nombres apropiado para permitirle al dispositivo detectar el modelo de datos apropiado, y los contenidos deben seguir las restricciones de este modelo de datos, tal como es definido por su definición de capacidad.

Respuesta positiva: si el dispositivo es capaz de satisfacer la solicitud, un <rpc-replay> es enviado conteniendo un elemento <ok>.

Respuesta negativa: una respuesta <rps-error> es enviada si la solicitud no puede ser completada por alguna razón.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
      <top xmlns="http://example.com/schema/1.2/config">
        <interface xc:operation="replace">
          <name>Ethernet0/0</name>
          <mtu>1500</mtu>
          <address>
            <name>192.0.2.4</name>
            <prefix-length>24</prefix-length>
          </address>
        </interface>
      </top>
    </config>
  </edit-config>
</rpc>
```

**Figura 6 Mensaje rpc operación edit-config. Tomado de (10)**

#### 2.1.1.8.3. <copy-config>

Descripción: crea o remplaza la totalidad de un almacenamiento de datos de configuración con los contenidos de otro almacenamiento de datos de configuración. Si el almacenamiento de datos destino existe, éste es sobrescrito. De otra manera, uno nuevo es creado, si se permite.

Incluso, si lo que se anuncia es la capacidad :writable-running, un dispositivo puede escoger no soportar la configuración de almacenamiento de datos <running/>, como parámetro <targert> de la operación <copy-config>. Un dispositivo puede escoger no soportar operaciones remotas de copiado remote-to-remote, donde ambos parámetros <source> y <target> usen el elemento <url>. Si los parámetros <source> y <target> identifican el mismo URL ó configuración de almacenamiento de datos, un error debe ser devuelto con el error error-tag conteniendo el valor “invalid-value”.

Parámetros:

- Target: nombre del almacenamiento de datos a utilizar como destinatario de la operación <copy-config>

- Source: nombre del almacenamiento de datos a utilizar como la fuente de la operación <copy-config>, ó el elemento <config> conteniendo la configuración completa a copiar.

Respuesta positiva: Si el dispositivo es capaz de satisfacer la solicitud, un <rpc-reply> es enviado y el cual incluye el elemento <ok>.

Respuesta negativa: Un elemento <rpc-error> es incluido dentro del <rpc-reply> si la solicitud no puede ser completadapor alguna razón.

#### 2.1.1.8.4. <delete-config>

Descripción: borra un almacenamiento de datos de configuración. El almacenamiento de datos de configuración <running> no puede ser borrado.

Parámetros:

- Target: nombre del almacenamiento de datos de configuración a ser borrado.

Respuesta positiva: Si el dispositivo puede satisfacer la solicitud, un <rpc-reply> es devuelto y el cual incluye el elemento <ok>

Respuesta negativa: un elemento <rpc-error> es incluido dentro de una <rpc-reply> si la solicitud no puede ser completada por alguna razón.

#### 2.1.1.8.5. <lock>

Descripción: la operación <lock> le permite al cliente bloquear todo el sistema de almacenamiento de configuración de un dispositivo. Este bloqueo está destinado a ser de corta duración y le permite al cliente realizar un cambio sin el temor de que exista una interacción con otros clientes NETCONF, o con clientes que no sean

NETCONF (por ejemplo, SNMP y CLI) o usuarios conectados a RS232.

Un intento para bloquear el almacenamiento de datos de configuración debe fallar si una sesión existente, u otra entidad, está sosteniendo un bloqueo en alguna porción del objetivo de bloqueo.

Cuando el bloqueo es adquirido, el servidor debe prevenir cualquier cambio a los recursos bloqueados, a excepción de las solicitadas por la sesión en curso.

Solicitudes SNMP y CLI para modificar los recursos deben fallar con un error apropiado.

La duración del bloqueo se define a partir del comienzo cuando el bloqueo es adquirido, hasta que el mismo es liberado ó la sesión termina. La sesión cerrada puede ser desarrollada explícitamente por el cliente, ó implícitamente desarrollada por el servidor basado en el criterio como falla del protocolo de transporte señalado, por tiempo de inactividad, o detección de comportamiento abusivo por parte del cliente. Este criterio depende de la implementación y del protocolo señalado de transporte.

La operación <lock> tiene obligatoriamente el campo <target>, el cual corresponde al nombre del almacenamiento de datos que será bloqueado. Cuando un bloqueo está activo, usando la operación <edit-config> en el almacenamiento de datos de configuración bloqueado, y utilizando la configuración bloqueada como objetivo de la operación <copy-config>, la operación será anulada por cualquier otra sesión NETCONF. Adicionalmente el sistema se asegurará que esos recursos de configuración bloqueados, no serán modificados por otro gestor de operaciones no NETCONF, tales como SNMP ó CLI. La operación <kill-session> puede ser usada para forzar la liberación de un bloqueo realizado por otra sesión NETCONF. Esto va más allá de lo abarcado por el RFC 6241 para definir cómo romper bloqueos manejados por otras entidades.

Un bloqueo no debe concederse si alguna de las siguientes condiciones son ciertas:

- Un bloqueo está actualmente en curso y manejado por cualquier otra sesión NETCONF u otra entidad.
- El almacenamiento de datos destino es el <candidate>, éste está siendo modificado actualmente, y estos cambios no han sido confirmados o devueltos.
- El almacenamiento de datos destino es el <running>, y otra sesión NETCONF tiene un compromiso confirmado en curso. Ver sección 8.4 del RFC 6241.

El servidor debe responder un elemento <ok> ó con un <rpc-error>.

Un bloqueo debe ser liberado por el sistema, si la sesión que está sosteniendo este bloqueo, es terminada por alguna razón.

Parámetros:

- Target: nombre del almacenamiento de datos de configuración a ser bloqueado.

Respuesta positiva: si el dispositivo está en la capacidad de responder a la solicitud, un <rpc-reply> es enviado y el cual contendrá el elemnto <ok>.

Respuesta negativa: un elemento <rpc-error> es incluido en el <rpc-reply>, si la solicitud no puede ser completada por alguna razón.
Si el bloqueo está actualmente sostenido, el elemento <error-tag> será “lock-denied” y el elemento <error-info> incluirá la <sesión-id> de quien está realizando el bloqueo. Si el bloqueo está siendo sostenido por una entidad no NETCONF, una <sesión-id> de ‘0’ (cero) es incluida. Notar que cualquier otra entidad aparte de

NETCONF que desarrolle un bloqueo, incluso si el mismo es parcial sobre el objetivo (target), impedirá un bloqueo NETCONF.

#### 2.1.1.8.6. <unlock>

Descripción: la operación <unlock> es usada para liberar una configuración bloqueada, previamente obtenida con la operación <lock>.

Una operación <unlock> no será exitosa si alguna de las siguientes condiciones es cierta:

- El bloqueo especificado actualmente no está activo
- La sesión que emite la operación <unlock> no es la misma sesión que obtuvo el bloqueo.

El servidor debe responder con un elemento <ok> o con un <rpc-error>

Parámetros: nombre del almacenamiento de datos de configuración a ser desbloqueado.

Respuesta positiva: si el dispositivo está en la capacidad de responder a la solicitud, un <rpc-reply> es enviado y el cual contendrá el elemnto <ok>.

Respuesta negativa: un elemento <rpc-error> es incluido en el <rpc-reply> si la solicitud no puede ser completada por alguna razón.

#### 2.1.1.8.7. <get>

Descripción: recupera la configuración running y la información de estado del dispositivo.

Parámetros:

- Filter: este parámetro especifica la porción del sistema de configuración y datos de estado a recuperar. Si este parámetro no está presente, toda la configuración e información de estado del dispositivo es retornada.

  El elemento <filter>, opcionalmente puede contener un atributo "type", el cual indica el tipo de sintaxis de filtrado usada dentro del elemento <filter>. El mecanismo por defecto de filtrado de NETCONF es referenciado como un filtrado de subárbol y es descrito en la sección 6 del RFC 6241. El valor "subtree", identifica explícitamente este tipo de filtrado.

Respuesta positiva: Si el dispositivo puede satisfacer la solicitud, un <rpc-reply> es enviado. La sección <data> contiene el subconjunto apropiado.

Respuesta negativa: un elemento <rpc-error> es incluido en el <rpc-reply> si la solicitud no puede ser completada por alguna razón.

#### 2.1.1.8.8. <close-session>

Descripción: Solicitud de finalización de precaución de una sesión NETCONF.

Cuando un servidor recibe una solicitud <close-session>, cerrará la sesión, liberará cualquier bloqueo y recurso asociado con la sesión, y realizará una finalización de precaución de cualquier conexión asociada. Cualquier solicitud NETCONF recibida después de un <close-session>, será ignorada. La figura Figura 7 muestra un ejemplo de rpc y rpc-reply de la operación closse-session.

Respuesta positiva: si el dispositivo puede satisfacer la solicitud, un <rpc-reply> es enviado e incluye el elemento <ok>.

Respuesta negativa: un elemento <rpc-error> es incluido en el <rpc-reply> si la solicitud no puede ser completada por alguna razón.

```
<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>
```

**Figura 7 Mensajes rpc y rpc-reply de la operación close-session.**

#### 2.1.1.8.9. <kill-session>

Descripción: forza la terminación de una sesión NETCONF.

Cuando una entidad NETCONF recibe una solicitud <kill-session> para una sesión abierta, esta abortará cualquier operación que esté actualmente en proceso, liberará cualquier bloqueo y recursos asociados con la sesión, y cerrará cualquier conexión asociada.

Si un servidor NETCONF recibe una solicitud <kill-session> durante el procesamiento de un compromiso confirmado (sección 8.4 del RFC 6241), el servidor debe restaurar la configuración a su estado previo antes de la confirmación del compromiso.

Por otro lado, la operación <kill-session> no deshace las modificaciones de configuración u otras modificaciones de estado del dispositivo, realizadas por la entidad que sostiene el bloqueo.

Parámetros:

- Session-id: identificador de sesión de la sesión NETCONF a ser terminada. Si este valor es igual al ID de sesión actual, un error “invalid-value” es devuelto.

Respuesta positiva: si el dispositivo puede satisfacer la solicitud, un <rpc-reply> es enviado y el cual incluye el elemento <ok>.

Respuesta negativa: un elemento <rpc-error> es incluido en el <rpc-reply> si la solicitud no puede ser completada por alguna razón.

#### 2.1.1.9. Capacidades

En la sección 8 del RFC6241, se define el conjunto de capacidades que un cliente ó un servidor pueden implementar. Cada uno de ellos, anuncia sus capacidades enviándolas en el intercambio inicial de capacidades. Cada par necesita entender solo aquellas capacidades que podría utilizar, e ignorar cualquier capacidad que reciba de otro par y la cual no entienda o no requiera. Las capacidades adicionales pueden ser definidas utilizando la plantilla que se encuentra en el apéndice D del RFC6241, y pueden ser publicadas como normas por los organismos de normalización, ó publicadas como extensiones propietarias..

Una capacidad NETCONF es identificada por un Identificador de Recurso Uniforme URI [RFC3986], y las capacidades base son definidas utilizando URNs (RFC3553).

Las capacidades aumentan las operaciones base de un dispositivo, describiendo tanto las operaciones adicionales como el contenido permitido dentro de las mismas. El cliente puede descubrir las capacidades del servidor y utilizar cualquier operación adicional, parámetros y contenido definido por estas capacidades

El URI de cada capacidad debe estar lo suficientemente diferenciado de los demás para evitar colisiones de nombramiento. Las capacidades definidas en el RFC6241 tienen el siguiente formato:

urn:ietf:params:netconf:capability:{name}:1.x

donde '*name*' hace referencia al nombre de la capacidad.

#### 2.1.1.9.1. Intercambio de capacidades

Las capacidades son anunciadas en mensajes enviados por cada par durante el establecimiento de una sesión. Cuando la sesión NETCONF es abierta, cada par (cliente y servidor), deben enviar un elemento <hello> en el que están listadas las capacidades que soporta cada par. Cada par debe enviar al menos la capacidad base de NETCONF: "urn:ietf:params:netconf:base:1.1". Un par puede incluir capacidades de versiones anteriores de NETCONF para indicar que soporta múltiples versiones de protocolo.

Ambos pares NETCONF deben verificar que el otro par ha anunciado una versión común de protocolo. Cuando se compara la versión del protocolo en las capacidades URI, solo la parte "base:1.x" es utilizada, y si no se encuentra una versión de protocolo en común, el par NETCONF no debe continuar la sesión. Si hay más de una versión de protocolo en común, entonces la versión más reciente de protocolo (número más alto de versión), debe ser utilizado por los dos pares.

Cuando un servidor envía el elemento <hello> debe incluir el elemento <sesión-id>, el cual tiene el ID de sesión para esa sesión NETCONF. Un cliente que envíe el elemento <hello>, no debe incluir el elemento <sesión-id>.

Un servidor que reciba el elemento <hello> con el elemento <sesión-id> debe terminar la sesión NETCONF. Similarmente, un cliente que no recibe el elemento <sesión-id> en el mensaje <hello> proveniente del servidor, debe terminar la sesión NETCONF, no sin antes enviar el elemento <closse-session>

En la Figura 8, se muestra el mensaje hello que envía un servidor. En este mensaje se anuncia la capacidad base de NETCONF, una capacidad de NETCONF definida en el documento base de NETCONF, y una capacidad de implementación específica:

```
<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>
```

**Figura 8 Mensaje Hello en el que se anuncian distintas capacidades**

Cada par NETCONF envía su elemento <hello> simultáneamente tan pronto como la conexión es abierta. Un par no debe esperar recibir el conjunto de capacidades del otro par para enviar su propio conjunto de capacidades.

#### 2.1.1.10. Separación de datos de configuración de datos de estado

La información que puede ser recuperada de un sistema en ejecución es separada en dos clases: datos de configuración y datos de estado.

El protocolo NETCONF reconoce la diferencia entre datos de Configuración y datos de Estado y provee operaciones para cada uno. Por ejemplo, la operación <get-config> recupera solo datos de configuración, mientras que la operación <get > recupera datos de configuración y de estado.

El protocolo NETCONF se centra en que el dispositivo cuente con la información necesaria para que opere en su estado de ejecución deseado. La inclusión de otros datos importantes y persistentes es específica de la aplicación. Por ejemplo, archivos y bases de datos de usuarios no son tratados como datos de configuración por NETCONF. Así, si una base de autenticación de usuario es almacenada en el dispositivo, esto se convierte en un asunto dependiente de implementación si esto es incluido en los Datos de Configuración.

#### 2.1.1.11. Listas de errores del protocolo NETCONF

En el apéndice A del RFC6241 están las listas de errores del protocolo NETCONF. Para cada error-tag, los valores válidos de error-type y error-severity son listados, junto con alguna error-info obligatoria.

#### 2.1.1.12. Esquema XML del protocolo de gestión de red NETCONF

El RFC6241 registra un URI en el registro XML de la IETF RFC3688, para el esquema del protocolo NETCONF. La IANA (Internet Assigned Numbers Authority) ha actualizado el siguiente URI para referenciar este esquema XML:

URI: urn:ietf:params:xml:schema:netconf

En el apéndice B del RFC6241 está el esquema XML de la capa de mensajes del protocolo NETCONF.

#### 2.1.1.13. Módulo YANG para las operaciones del protocolo NETCONF

El módulo YANG utilizado en el presente proyecto para modelar las operaciones del protocolo NETCONF, fue tomado del apéndice C del RFC6241. Ver Anexo 2.

#### 2.1.1.14. Typedefs importados por el módulo YANG

El módulo YANG importa los typedefs del RFC6021. Ver Anexo 3.

## 2.2. YANG Lenguaje de modelado de datos para el protocolo de configuración de red NETCONF (11)

### 2.2.1. Definición y descripción general del lenguaje

YANG es un lenguaje de modelado de datos usado para modelar los datos de configuración y los datos de estado manipulados por el protocolo de configuración de red NETCONF, por las RPCs, y por las notificaciones de NETCONF. También, YANG es usado para modelar

las capas de operación y contenido del protocolo de gestión de red de datos NETCONF.

Así, el lenguaje YANG es definido en el RFC 6020, donde se describe la sintaxis y semántica del lenguaje, cómo el modelo de datos definido en un modulo de YANG es representado en XML (Extensible Markup Languaje), y cómo las operaciones de NETCONF son usadas para manipular los datos.

Un módulo YANG define una jerarquía de datos que puede ser utilizada para las operaciones base del protocolo NETCONF, para los datos de configuración, para los datos de estado, para los datos de las *Remote Procedure Calls* (RPCs), y para los datos de las notificaciones. Esto permite una descripción completa de todos los datos enviados entre un cliente y un servidor NETCONF.

Así, YANG modela la organización jerárquica de los datos como un árbol, en el que cada nodo tiene un nombre y un valor ó un conjunto de nodos hijos. YANG proporciona una descripción clara y concisa de los nodos, así como de la interacción entre estos nodos. Los módulos y submódulos son utilizados por YANG, para contener la estructura de los modelos de datos. Un módulo puede importar datos provenientes de otros módulos externos, e incluir datos provenientes de submódulos. También, el contenido de la jerarquía de un módulo puede ser aumentado, adicionándole otros nodos de datos definidos en otro módulo. Este aumento puede ser condicional, donde los nuevos nodos solo aparecen si ciertas condiciones se cumplen.

Los modelos YANG pueden describir limitaciones que se aplican en los datos, restringiendo la aparición ó el valor de nodos, basándose en la presencia ó valor de otros nodos en la jerarquía. Estas restricciones son cumplidas por el cliente ó por el servidor, y el contenido de los datos debe cumplir con dichas restricciones.

YANG define un conjunto de tipos built-in, y tiene un mecanismo a través del cual tipos adicionales pueden ser definidos. Tipos derivados

pueden restringir los valores válidos de los tipos base, usando mecanismos como restricciones de rango ó patrón, que pueden ser aplicados por clientes ó servidores. También, los tipos derivados pueden definir convenciones de cómo manejar otros tipos derivados, tal como un tipo basado en strings que puede ser el nombre de un host.

YANG permite la definición de agrupaciones reutilizables de nodos. La creación de instancias de estas agrupaciones puede mejorar ó aumentar el contenido de los nodos, permitiendo adaptar los nodos a sus necesidades particulares. Los tipos derivados y las agrupaciones, pueden ser definidos en un módulo ó submódulo, y ser usados en estas ubicaciones ó en otro módulo ó submódulo que los importa ó los incluye.

Las construcciones jerárquicas de datos YANG incluyen definiciones de listas, donde las entradas de las listas son identificadas mediante claves que las distinguen unas de otras. Tales listas pueden ser ordenadas por el usuario ó automáticamente ordenadas por el sistema. Para listas ordenadas por el usuario, las operaciones son definidas para manipular el orden de las entradas de las lista.

Los módulos YANG pueden ser trasladados a una sintaxis XML equivalente llamada YANG *Independent Notation* (YIN), permitiéndole a las aplicaciones utilizar analizadores XML y scripts *Extensible Stylesheet Language Transformations* (XSLT) para operar en los modelos. La conversión de YANG a YIN se realiza sin pérdidas, de tal manera que el contenido de YIN puede ser devuelto a YANG.

El lenguaje YANG, establece un balance entre el modelado de datos de alto nivel y el de bajo nivel. El lector del módulo YANG mantiene la vista de alto nivel del modelo de datos, mientras comprende cómo los datos serán codificados en las operaciones NETCONF.

También, el lenguaje YANG es un lenguaje extensible, permitiendo que declaraciones de extensión puedan ser definidas por organismos de normalización, proveedores e individuales. La sintaxis de la declaración

les permite a estas extensiones coexistir de forma natural con declaraciones YANG normalizadas.

En la medida de lo posible, YANG mantiene compatibilidad con Simple Network Management Protocol's (SNMP's) SMIv2 (Structure of Management Information version 2 [RFC2578], [RFC2579]). Los módulos MIB de SMIv2-based pueden ser automáticamente trasladados en módulos YANG para acceso de solo lectura. Sin embargo, YANG no se ocupa de la reversión de YANG a SMIv2.

Al igual que NETCONF, YANG apunta a la integración con la infraestructura nativa de gestión del dispositivo. Esto permite implementaciones que aprovechan los mecanismos existentes de control de acceso del dispositivo.

### 2.2.2. Terminología

Ver Anexo 4

### 2.2.3. Descripción de los componentes básicos para definir un modelo de datos

Esta sección introduce algunas construcciones importantes usadas en el lenguaje de modelado de datos YANG, y que ayudarán para el entendimiento de los detalles más relevantes del módulo YANG descrito en el RFC6241 para la definición del modelo de datos para las operaciones del protocolo NETCONF.

#### 2.2.3.1. Módulos y Submódulos

Un módulo contiene tres tipos de declaraciones: declaración de cabecera del módulo (module-header), declaraciones de revisión, y declaraciones de definición. Las declaraciones de cabecera del módulo, describen el módulo, y otorga información sobre el propio módulo. Las declaraciones de revisión otorgan información sobre la historia del módulo. Finalmente, las declaraciones de definición son el cuerpo del módulo donde el modelo de datos es definido.

Un servidor NETCONF puede implementar un número de módulos, permitiendo múltiples vistas de los mismos datos, ó múltiples vistas de disjuntas subsecciones de los datos del dispositivo. Alternativamente, el servidor puede implementar solo un módulo que defina todos los datos disponibles.

Un módulo puede ser dividido en submódulos, basado en las necesidades del propio módulo. La vista externa sigue siendo el de un único módulo, independientemente de la presencia ó el tamaño de estos submódulos.

La declaración “include” permite a un módulo ó submódulo referenciar material en submódulos, y la declaración “import” permite la referencia de material definida en otros módulos.

#### 2.2.3.1.1. Bases de modelado de datos

YANG define cuatro tipos de nodos para el modelado de datos. En cada una de las subsiguientes secciones, el ejemplo muestra la sintaxis YANG así como su correspondiente representación NETCONF XML.

##### 2.2.3.1.1.1. Nodos hoja

Un nodo hoja contiene datos simples como un entero ó un string. Tiene exáctamente un valor de un tipo particular y no tiene nodos hijos.

Ejemplo YANG:

```
leaf host-name {
        type string;
        description "Hostname for this system";
    }
```

Ejemplo NETCONF XML:

```
<host-name>my.example.com</host-name>
```

##### 2.2.3.1.1.2. Nodos Leaf-List

Una lista de hoja es una secuencia de nodos hoja con exactamente un valor de un tipo particular por hoja

Ejemplo YANG:

```
leaf-list domain-search {
      type string;
      description "List of domain names to search";
   }
```

Ejemplo NETCONF XML:

```
   <domain-search>high.example.com</domain-search>
   <domain-search>low.example.com</domain-search>
   <domain-search>everywhere.example.com</domain-
search>
```

La declaración “leaf-list” es cubierta en la sección 7.7 del RFC6020

#### 2.2.3.1.1.3. Nodos container

Un nodo container es utilizado para agrupar en un subárbol, nodos relacionados. Un container solo tiene nodos hijos más no valores. Un container puede contener cualquier número de nodos hijos de cualquier tipo, incluyendo nodos leafs, lists, containers, y leaf-lists.

Ejemplo YANG:

```
container system {
      container login {
         leaf message {
            type string;
            description
               "Message given at start of login session";
         }
      }
   }
```

Ejemplo NETCONF XML:

```
<system>
```

```
    <login>
      <message>Good morning</message>
    </login>
  </system>
```

La declaración “container” es cubierta en la sección 7.5 del RFC6020.

#### 2.2.3.1.1.4. Lista de nodos

Una lista define una secuencia de entradas de lista. Cada lista es como una estructura ó un registro de la instancia, y es identificado únicamente por los valores de sus hojas principales.

Una lista puede definir múltiples hojas clave y puede contener cualquier número de nodos hijos de cualquier tipo (incluyendo hojas, listas, contenedores, etc).

Ejemplo YANG:

```
list user {
      key "name";
      leaf name {
         type string;
      }
      leaf full-name {
         type string;
      }
      leaf class {
         type string;
       }
   }
```

Ejemplo NETCONF XML:

```
<user>
    <name>glocks</name>
    <full-name>Goldie Locks</full-name>
```

```
    <class>intruder</class>
  </user>
  <user>
    <name>snowey</name>
    <full-name>Snow White</full-name>
    <class>free-loader</class>
  </user>
  <user>
    <name>rzell</name>
    <full-name>Rapun Zell</full-name>
    <class>tower</class>
  </user
```

La declaración “lista” es cubierta en la sección 7.8 del RFC6020.

#### 2.2.3.1.1.5. Ejemplo de módulo

Estas declaraciones son combinadas para definir el módulo:

```
// Contents of "acme-system.yang"
    module acme-system {
        namespace "http://acme.example.com/system";
        prefix "acme";
        organization "ACME Inc.";
        contact "joe@acme.example.com";
        description
            "The module for entities implementing the ACME
system.";
        revision 2007-06-09 {
            description "Initial revision.";
        }
        container system {
            leaf host-name {
```

```
            type string;
            description "Hostname for this system";
        }
        leaf-list domain-search {
            type string;
            description "List of domain names to search";
        }
    container login {
            leaf message {
                type string;
                description
                    "Message given at start of login session";
            }
            list user {
                key "name";
                leaf name {
                    type string;
                }
                leaf full-name {
                    type string;
                }
                leaf class {
                    type string;
                }
            }
        }
    }
```

### 2.2.4. Equipos que actualmente integran el lenguaje de modelado de datos YANG en su sistema operativo

Se consulta qué equipos soportan el esquema de modelado de datos YANG (especialmente de los proveedores Cisco y Juniper), pero no se encuentra nada en la bibliografía de la IEEE ni de Springer. Se consulta en el buscador Web Google, y se encuentra que algunos equipos robustos de los proveedores Siemens, y Netlron soportan el esquema de modelado de datos YANG, pero no se encuentra ninguna información referente a los proveedores Cisco ó Juniper. En el caso de Siemens, en (15) se valida que los equipos Siemens que tengan el sistema operativo Rox II, soportan el modelado de datos YANG. Observar que la documentación es bastante reciente (Julio de 2013). En el caso de Netlron, en (16) se valida que los equipos Netlron que tengan la versión de software 05.2.00, soportan el lenguaje de modelado de datos YANG.

## 3. ESPECIFICACIONES

El prototipo desarrollado en el presente proyecto es una plataforma de gestión de red que está basada en el protocolo de gestión de red NETCONF y en el lenguaje de modelado de datos YANG. A continuación se describen las características necesarias de los equipos de red que se desean gestionar, y las especificaciones propias del prototipo:

### 3.1. Características necesarias de los equipos a gestionar utilizando el prototipo desarrollado

**3.1.1.** Deben ser equipos de red de datos clasificados dentro de las capas 2 y 3 del modelo TCP/IP.

**3.1.2.** Deben ser equipos de los proveedores Cisco y Juniper.

**3.1.3.** Los equipos deben soportar el protocolo de gestión de red Netconf.

**3.1.4.** Cada uno de los equipos que se desean gestionar, requieren ser previamente configurados con:

- ✓ Hostname (nombre del equipo)
- ✓ Dominio

- ✓ Clave de encripción para conexión ssh
- ✓ Protocolo de transmisión ssh
- ✓ Usuario de inicio de conexión ssh
- ✓ Contraseña de inicio de conexión ssh
- ✓ Habilitar acceso remoto
- ✓ IP de acceso remoto

## 3.2. Especificaciones del prototipo desarrollado

El prototipo desarrollado en este proyecto cumple con las características más importantes del protocolo de gestión de red de datos NETCONF descritas en el RFC6241, y con las características más importantes del modelado de datos YANG descritas en el RFC6020. A continuación se listan las especificaciones del prototipo desarrollado:

### 3.2.1. Inicio de sesión

**3.2.1.1.** Los parámetros de inicio de sesión netconf son: usuario (string), host (dirección IP) y contraseña (string).

**3.2.1.2.** La interfaz visual principal del prototipo de software de gestión de equipos de red de datos (cliente), tiene habilitados campos de texto para que un usuario ingrese los parámetros de inicio de sesión netconf para poder establecer esta conexión con un equipo de red de datos (servidor).

**3.2.1.3.** El cliente envía al servidor un mensaje Hello, cuando el usuario le indica que desea establecer una sesión netconf con el servidor.

**3.2.1.4.** El mensaje Hello cumple con las especificaciones del RFC6241. Ver Anexo 5.

### 3.2.2. Operaciones de configuración de los equipos de red de datos

Después de iniciada una sesión netconf utilizando el prototipo de software de gestión, la interfaz visual le permite al usuario escoger entre las siguientes opciones:

**3.2.2.1.** Operación de recuperar toda la configuración del equipo de red de datos

Para realizar esta operación, se utiliza la operación get-config descrita en el RFC6241 (Ver Anexo 8). Esta operación se envía desde el cliente al servidor utilizando la rpc especificada en el RFC6241, cuando el usuario le indica al cliente que desea ejecutar esta operación. Para que el cliente ejecute la operación que el usuario ha escogido, se debe pulsar un botón de enviar en la interfaz visual.

**3.2.2.2.** Operación de configuración de una interface física del equipo de red de datos

Para realizar esta operación, se utiliza la operación edit-config descrita en el RFC6241 y ajustada al esquema particular de los equipos (Ver Anexo 9 y Anexo 10). Esta operación se envía desde el cliente al servidor utilizando la rpc especificada en el RFC6241, cuando el usuario le indica al cliente que desea ejecutar esta operación. Para que el cliente ejecute la operación que el usuario ha escogido, se debe pulsar un botón de enviar en la interfaz visual.

### 3.2.3. Estructura de los elementos rpc XML

Los elementos rpc enviados por el cliente al servidor son del tipo rpc-request, y cumplen con las especificaciones descritas en el RFC6241( Ver Anexo 8, Anexo 9 y Anexo 10). También, estas rpc cumplen con los esquemas propios de los equipos que se van a gestionar (Ver Anexo 11 y Anexo 12 ).

### 3.2.4. Finalización de la sesión netconf

El cliente finaliza la sesión netconf establecida con el servidor, enviando la operación close-session descrita en el RFC6241 (Ver Anexo 14). Para que el cliente finalice la sesión netconf, es necesario el usuario haciendo uso de la interfaz visual, pulse el botón "enviar".

Para el cumplimiento de las anteriores especificaciones, se diseñó la siguiente arquitectura de software:

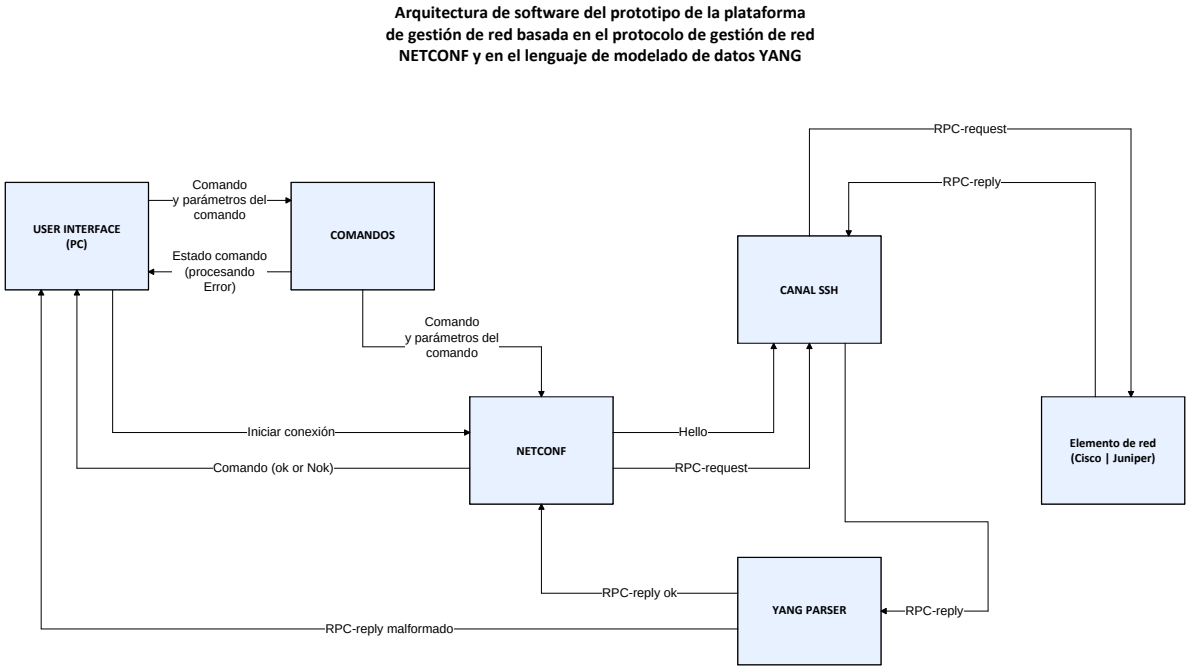


**Figura 9. Arquitectura de software del prototipo de la plataforma de gestión de red. Tomado de (17)**

En la Figura 10, se ilustra el diseño del módulo Netconf del prototipo de la plataforma de gestión de red de datos.

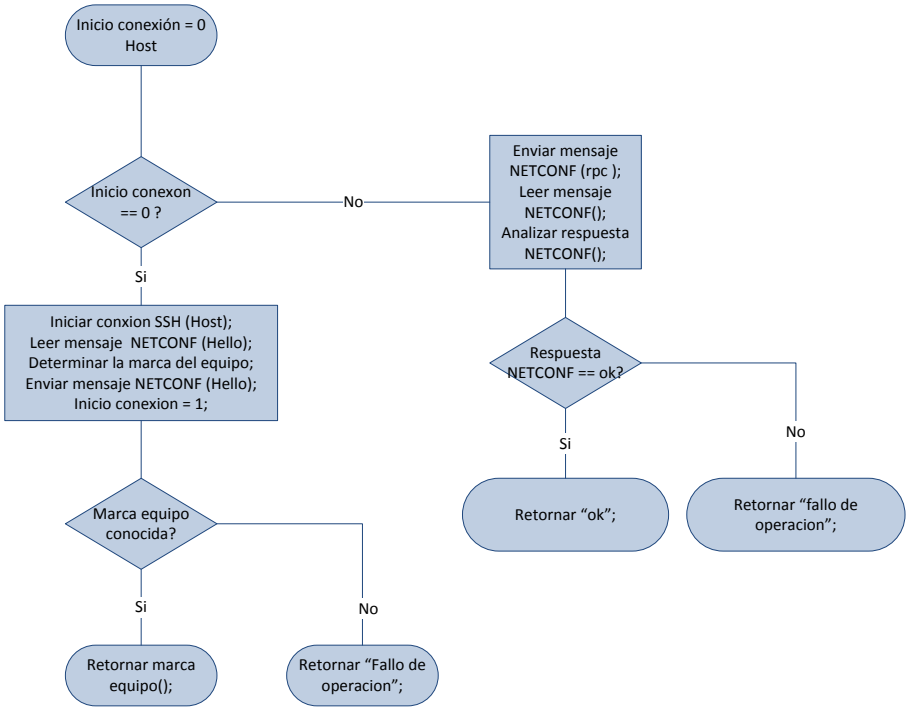


**Figura 10. Diagrama de flujo del módulo Netconf del prototipo de la plataforma de gestión de red de datos. Tomado de (17)**

Mediante la implementación en software de la arquitectura de la Figura 9, se obtiene el siguiente escenario de gestión de equipos de red, y aunque solo se muestran dos equipos de red, esto no es restrictivo, ya que el prototipo desarrollado tiene la capacidad de gestionar múltiples equipos adicionales:

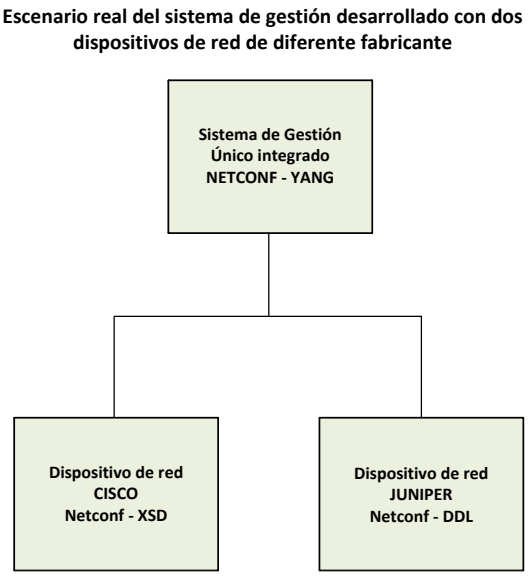


**Figura 11. Escenario real del sistema de gestión de red desarrollado. Tomado de (17)**

## 4. DESARROLLOS

La implementación en software de la arquitectura de la Figura 9, se realizó en el lenguaje de programación java y se utilizó el IDE de eclipse. Así, para el desarrollo de este prototipo de la plataforma de gestión de red basada en el protocolo de gestión de red NETCONF y en el lenguaje de modelado de datos YANG, se tuvo en cuenta los principios y fases de la metodología RUP, cuyas principales recomendaciones se enfocan en dar pautas para ejecutar las mejores prácticas para el análisis, diseño, implementación, pruebas y entrega final del producto desarrollado. Así, siguiendo el principio de adaptar el proceso de esta metodología, se desarrolló el prototipo de software de gestión para que se adaptara a las necesidades del usuario teniendo en cuenta los objetivos y restricciones del proyecto.

En el transcurso del desarrollo del proyecto, se presentaron circunstancias en las que ciertas actividades demandaban más tiempo y esfuerzo que otras, siendo necesaria la implementación del principio de equilibrar prioridades(18). De esta manera, se debió establecer un equilibrio en la distribución y uso de estos recursos, con el objetivo de realizar todas las actividades de la mejor forma posible y así poder satisfacer todos los requerimientos del proyecto.

En las actividades referentes a programación en java, se tuvo en cuenta la recomendación del principio de elevar el nivel de abstracción, reutilizando códigos fuente del mismo proyecto creado para el desarrollo del prototipo de la plataforma de gestión de red. También, para continuar con el ahorro de recursos de tiempo y esfuerzo de desarrollo, se utilizaron librerías ya existentes y desarrolladas por otros programadores, evitando de esta manera realizar las mismas tareas ya realizadas por otras personas.

Finalmente, cabe mencionar que los principios de demostrar el valor iterativamente, colaboración entre equipos, y enfocarse en la calidad, se tuvieron en cuenta en todo el proceso de desarrollo de este proyecto, ya que se tuvo una comunicación fluida con el director del proyecto, manteniendo una revisión contínua de los avances del mismo, permitiendo su evaluación, detección y corrección de errores. También, en las reuniones de estas revisiones, se coordinaron los planes a ejecutar para obtener los resultados esperados en el tiempo definido. Así, todas estas actividades realizadas en todos los procesos y etapas de desarrollo del proyecto, siempre estuvieron enfocadas en obtener la mejor calidad del prototipo final.

## 4.1. Actividades realizadas para la implementación en software de la arquitectura del prototipo de gestión de red de datos.

Para la implementación en software de la arquitectura del prototipo de gestión ilustrada en la Figura 9, se requirió realizar las siguientes actividades:

### 4.1.1. Adicionar la librería SSH jsch en eclipse. Ver Figura 12

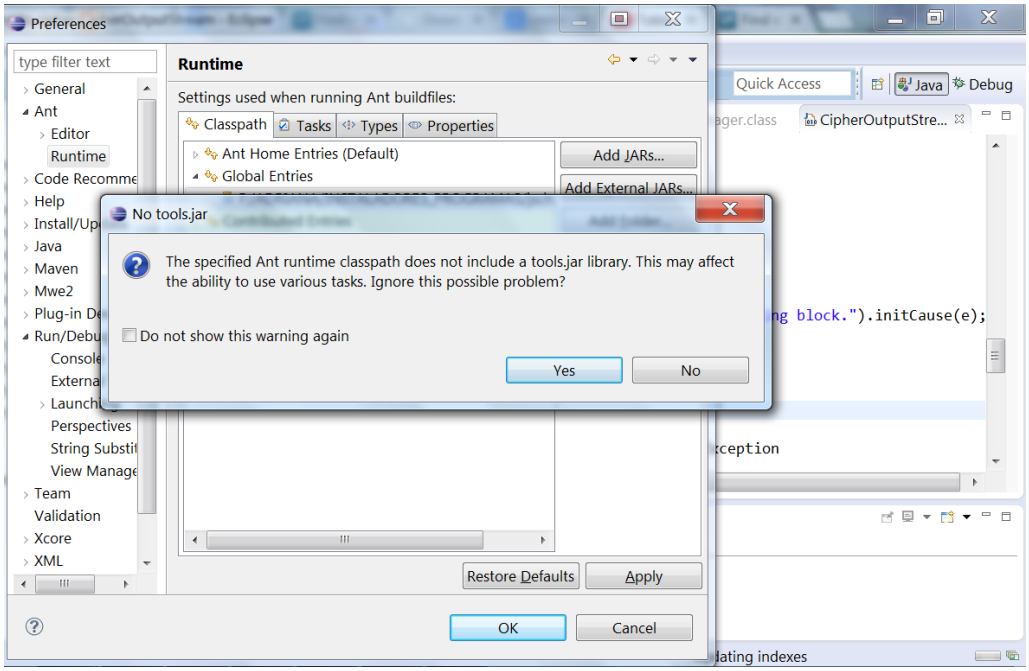


Figura 12. Inclusión de la librería JSch al classpath de eclipse. Tomado de (17)

**4.1.2.** Descargar e instalar Windows Builder en Eclipse (19). En la página referenciada, se encuentra tanto el link de descarga del Windows Builder, como los pasos para instalarlo según la versión de Eclipse.

**4.1.3.** Crear un proyecto java en la plataforma eclipse, y por cada módulo que está en la arquitectura de software del prototipo (a excepción del componente "Elemento de red Cisco | Juniper", que hace referencia a un elemento externo del software), crear una clase java con su respectivo nombre. También, se debió crear una clase principal que se encargara de suministrarle a la clase User Interface, los parámetros para el inicio de conexión entre el cliente y el servidor. Para el caso de la clase Principal del programa y de la clase User Interface, estas se crearon como clases del tipo Windows Builder

A continuación se describen cada una de las clases que se crearon en el proyecto de eclipse:

**4.1.3.1.** Clase principal del proyecto

En el Anexo 15 se encuentra el código fuente de esta clase.

La creación de esta clase es fundamental, ya que le permite al usuario del prototipo de software de gestión, establecer simultáneamente varias sesiones netconf con distintos equipos de red, ya que esta clase siempre ejecuta una interfaz visual donde se le permite ingresar al usuario los parámetros de inicio de conexión con un equipo de red. Esta clase contempla que cada vez que se pulsa el botón "Configurar equipo", se abra una nueva ventana donde se se encuentra la opción de establecer la sesión netconf con el equipo de red.

#### **4.1.3.2.** Clase User Interface

En el Anexo 16 se encuentra el código fuente de esta clase, la cual crea una interfaz visual, donde se encuentran en un desplegable, las diferentes operaciones definidas en el numeral 3 de especificaciones del presente trabajo.

Inicialmente se debe escoger la operación "Iniciar Conexión" para que se inicie la sesión NETCONF con el dispositivo de red. Los parámetros que se utilizan para establecer esta conexión son los ingresados en la interfaz visual de la clase principal del proyecto. Si estos parámetros son incorrectos para el dispositivo de red, la sesión NETCONF no se establecerá. Posterior al inicio de conexión, se podrá enviar cualquier otra operación al equipo de red.

#### **4.1.3.3.** Clase Comandos

En el Anexo 18 se encuentra el código fuente de esta clase, la cual realiza la interpretación de los comandos de operación que se muestran en las ventanas de configuración de los equipos. Asi, el comando que se muestra en la interfaz gráfica, se traduce a la rpc en XML que se necesita enviar al equipo, de tal forma que se obtenga el resultado esperado.

#### **4.1.3.4.** Clase Netconf

En el Anexo 19 se encuentra el código fuente de esta clase. La clase Netconf envía las operaciones NETCONF a la clase Canal SSH para que sean enviadas al equipo de red que se esté gestionando. También se encarga de recibir las respuestas XML NETCONF del equipo gestionado que le transmite la clase Canal SSH, verificando siempre si la totalidad de la rpc ha sido recibida y deacuerdo a su contenido, retornar una respuesta al operador de red indicando si la operación enviada se ejecutó en el dispositivo de red de manera exitosa ó si no fue de esta manera. Para que el operador de red visualice en la consola de la ventana de configuración la información anterior, la clase Netconf le debe proveer esta información a la clase Comandos y esta a su vez conforma el mensaje que imprimirá la clase User Interface en la consola de la ventana de configuración del equipo.

#### **4.1.3.5.** Clase Canal SSH

En el Anexo 17 se encuentra el código fuente de la clase Canal SSH. Esta clase permite establecer el canal de comunicación para que el prototipo de la plataforma de gestión, pueda establecer una conexión SSH V2 con el equipo gestionado.

#### **4.1.3.6.** Clase YANG parser

En el Anexo 20 se encuentra el código fuente de la clase YANG parser. Esta clase hace uso del esquema de datos Yang de las operaciones del protocolo Netconf, para realizar el análisis de las rpc enviadas al servidor, y para realizar también el análisis de las rpc-reply provenientes del servidor como respuestas a las rpc que recibió. Es necesario realizar la conversión del formato .yang de este esquema, a un formato .xml, ya que no se encontró ninguna herramienta en java disponible que realizara el análisis de archivos xml deacuerdo a un formato .yang. Esta conversión

se realiza sin pérdidas utilizando la herramienta Pyang disponible en la página de YANG Central Ver (20).

La conversión al esquema XSD del esquema de datos Yang para las operaciones del protocolo Netconf, se relaciona en el Anexo 21. También, en el anterior anexo se encuentra incluido el módulo de ietf-inet-types que el módulo YANG originalmente importa. Se realizó de esta manera para efectos prácticos de programación. Adicionalmente, el esquema XSD referente al esquema de datos Yang para las operaciones del protocolo Netconf, utiliza un esquema adicional definido en otro archivo llamado "netconf.xml" y el cual, contiene el esquema de datos para la capa de mensajes del protocolo NETCONF, y al cual pertenecen los mensajes Hello, rpc y rpc-reply. Como ya se mencionó en la sección 2 de teoría del presente documento, este esquema de datos se encuentra referenciado como el apéndice B del RFC6241(Anexo 22).

Para que esta clase funcione, es necesario que el archivo correspondiente al esquema de datos para las operaciones del protocolo Netconf (ietf-netconf_00.xsd), el archivo correspondiente al esquema de datos de la capa de mensajes del protocolo NETCONF (netconf.xsd), y el archivo correspondiente a la RPC que se desea analizar (rpc.xml), se encuentren en el mismo directorio raíz (Observar en el anexo de la clase, que los archivos ietf-netconf_00.xsd, y rpc.xml, se encuentran en el mismo directorio raíz que es D://).

## 4.2. Diagrama de máquina de estados del prototipo de software de gestión.

En la Figura 13 se muestra la máquina de estados del prototipo de software de gestión

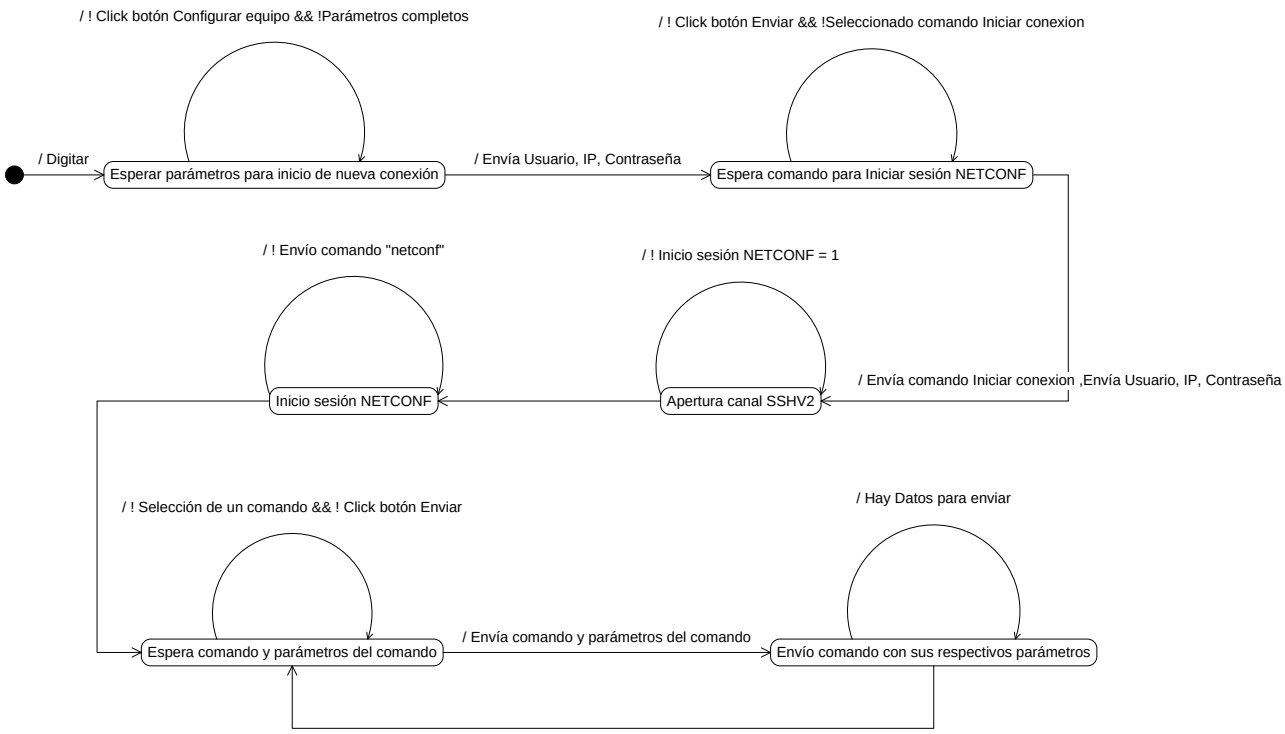


**Figura 13. Diagrama de máquina de estados del prototipo de software de gestión. Tomado de (17)**

La máquina de estados del prototipo de software de gestión se resume en palabras de la siguiente manera:

**4.2.1.1.** El estado inicial de la máquina de estados es la espera del ingreso de los tres parámetros de inicio de conexión Usuario, IP, Contraseña. Se sale de este estado cuando el gestor de red oprime el botón de configurar equipo y están todos los tres parámetros de inicio de conexión.

**4.2.1.2.** El siguiente estado es la espera del comando de inicio de sesión NETCONF. Se pasa al siguiente estado escogiendo el comando "Iniciar conexión" y pulsando el botón "Enviar".

**4.2.1.3.** El estado que sigue es la espera para la apertura de un canal SSH V2. Se pasa al siguiente estado cuando se haya establecido el canal SSH V2 haciendo uso de los parámetros correctos ingresados por el usuario.

**4.2.1.4.** Posteriormente se envía comando "netconf" para iniciar sesión NETCONF. Se pasa al siguiente estado cuando se haya enviado al equipo

de red con el que se estableció conexión SSH V2, el comando “netconf”, cuando haya llegado el mensaje hello Netconf del equipo de red y cuando se haya enviado el mensaje hello Netconf al equipo de red.

**4.2.1.5.** Ya iniciada la sesión NETCONF, se ingresa al estado de espera de comando y parámetros del comando, donde el administrador del equipo puede enviar cualquiera de las siguientes operaciones “Obtener toda la configuración” , “Configurar interface” y “Cerrar sesión”. Se mantiene este estado hasta que no se seleccione un comando y no se pulse el botón “Enviar”.

**4.2.1.6.** El siguiente estado de Envío de comando con sus respectivos parámetros se mantiene hasta que se hayan enviado todos estos parámetros al dispositivo de red utilizando XML. Se finaliza este estado cuando ya no hay más datos para transmitir al equipo de red.

**4.2.1.7.** Finalmente, se pasa nuevamente al estado de espera de comando y parámetros del comando, y se repite la dinámica de los numerales 4.2.1.6 y 4.2.1.7.

**4.2.1.8.** Cuando el administrador de equipos de red lo desee, puede comenzar con el inicio de la máquina de estados para iniciar nuevas sesiones Netconf, ello siguiendo el procedimiento anteriormente descrito.

## 4.3. Diagrama de clases del prototipo de software de gestión.

En la Figura 14 se ilustra el diagrama de clases del prototipo de software de gestión, mostrándose claramente las interacciones entre estas clases.

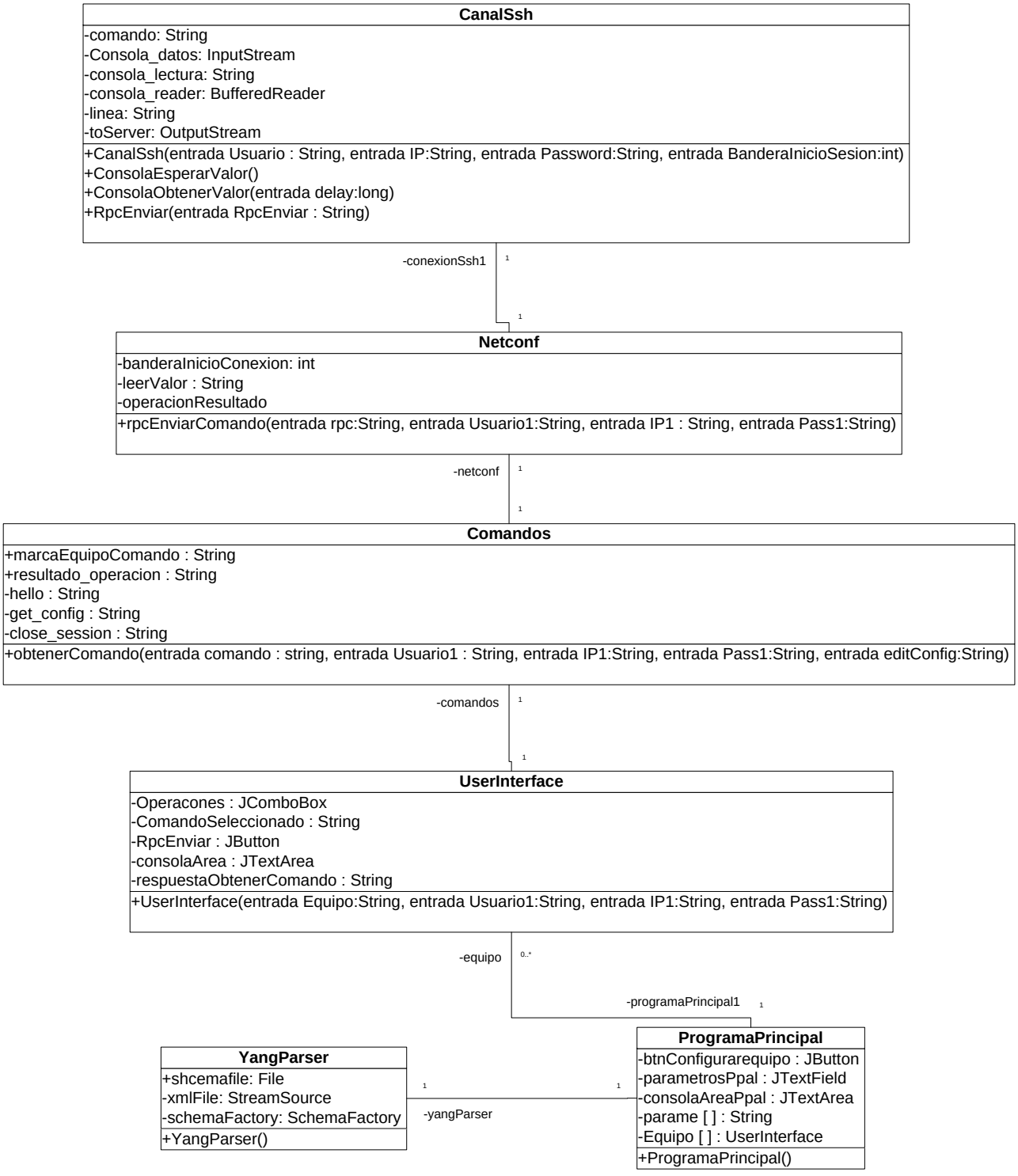


**Figura 14. Diagrama de Clases del prototipo de software de gestión. Tokmado de (17)**

Se observa que hay una clase principal llamada ProgramaPrincipal, que como ya se ha dicho anteriormente, se encarga de visualizar una interfaz principal y la cual está a la espera de parámetros de inicio de conexión con un dispositivo.

Cuando un usuario ingresa los tres parámetros de inicio de conexión (contador "i" de los token es igual a 3) y pulsa el botón Enviar, el programa principal crea una nueva ventana (nuevo frame) y pasa como parámetros de referencia, los parámetros de inicio de conexión a la clase UserInterface. La creación de esta nueva ventana se da al ejecutarse la clase UserInterface, y en la cua hay un desplegable (JComboBox) con cada una de las operaciones que puede ejecutar el operador de red en el dispositivo de red. Tal como se observa, la clase ProgramaPrincipal puede abrir múltiples User Interface y esto se hace con la creación de múltiples objetos de la clase UserInterface.

Seguidamente, la clase UserInterface está a la espera que el operador de red escoja la opción "IniciarConexión" y pulse el botón "Enviar". Cuando ocurre esto, se realiza llamado del módulo Comandos, y se le pasan los parámetros necesarios para el inicio de la sesión NETCONF, así como el comando que el usuario escogió, que en este caso es "IniciarConexion". El string respuestaObtenerComando de la clase UserInterface es igual a lo que retorna el método "obtenerComando" de la clase comandos y sirve para filtrar la respuesta del resultado del envío de la operación, y la cual se deberá imprimir en la consola de la ventana de configuración del equipo. También, indica la marca del dispositivo con el cual se inició la sesión NETCONF.

En la clase Comandos, se detecta la operación que la clase UserInterface pasó como parámetro de referencia, y la cual en el caso que se está describiendo fue "IniciarConexion". Así, se llama el método rpcEnviar de la clase Netconf y se pasa como parámetros de referencia el mensaje hello (String XML), y los parámetros Usuario, IP y contraseña, necesarios para el inicio de la sesión NETCONF. Lo que retorna este método se almacena en el String resultado_operación, y sirve para indicarle a la clase UserInterface la marca del equipo con el cual se inició sesión NETCONF, y para indicarle que la conexión se estableció exitosamente con ese dispositivo.

El método rpcEnviar se encarga de pasar como parámetros de referencia los parámetros necesarios para el inicio de la sesión NETCONF (hello, usuario, IP y contraseña), al método CanalSsh de la clase CanalSsh. Ejecutado este método CanalSsh, mediante la librería Jsch se crea una sesión netconf através de una conexión SSH V2 y la cual se puede establecer con la revisión o la no revisión de llaves. Posteriormente al establecimiento de la conexión SSH V2, el método rpcEnviar de la clase Netconf, ejecuta el método RpcEnviar de la clase CanalSsh y através de este método se realiza el envío del hello através del canal SSH V2 establecido.

Cuando ya se ha establecido la sesión NETCONF, el operador de red puede ejecutar las operaciones de Obtener toda la configuración, Configurar Interface y Cerrar Sesión en el dispositivo de red con el que se estableció la sesión NETCONF.

## 4.4. Diagramas de secuencia del prototipo de software de gestión.

Los siguientes diagramas reflejan la secuencia de mensajes intercambiados entre las diferentes clases del prototipo de software de gestión y el equipo de red gestionado, cuando no ocurre un error de conexión entre el prototipo de software de gestión y el elemento de red, así como tampoco errores en la digitación de parámetros requeridos para enviar las operaciones al elemento de red.

En la Figura 15 se ilustra el diagrama de secuencia del inicio de sesión NETCONF entre el prototipo de software de gestión y el equipo de red.

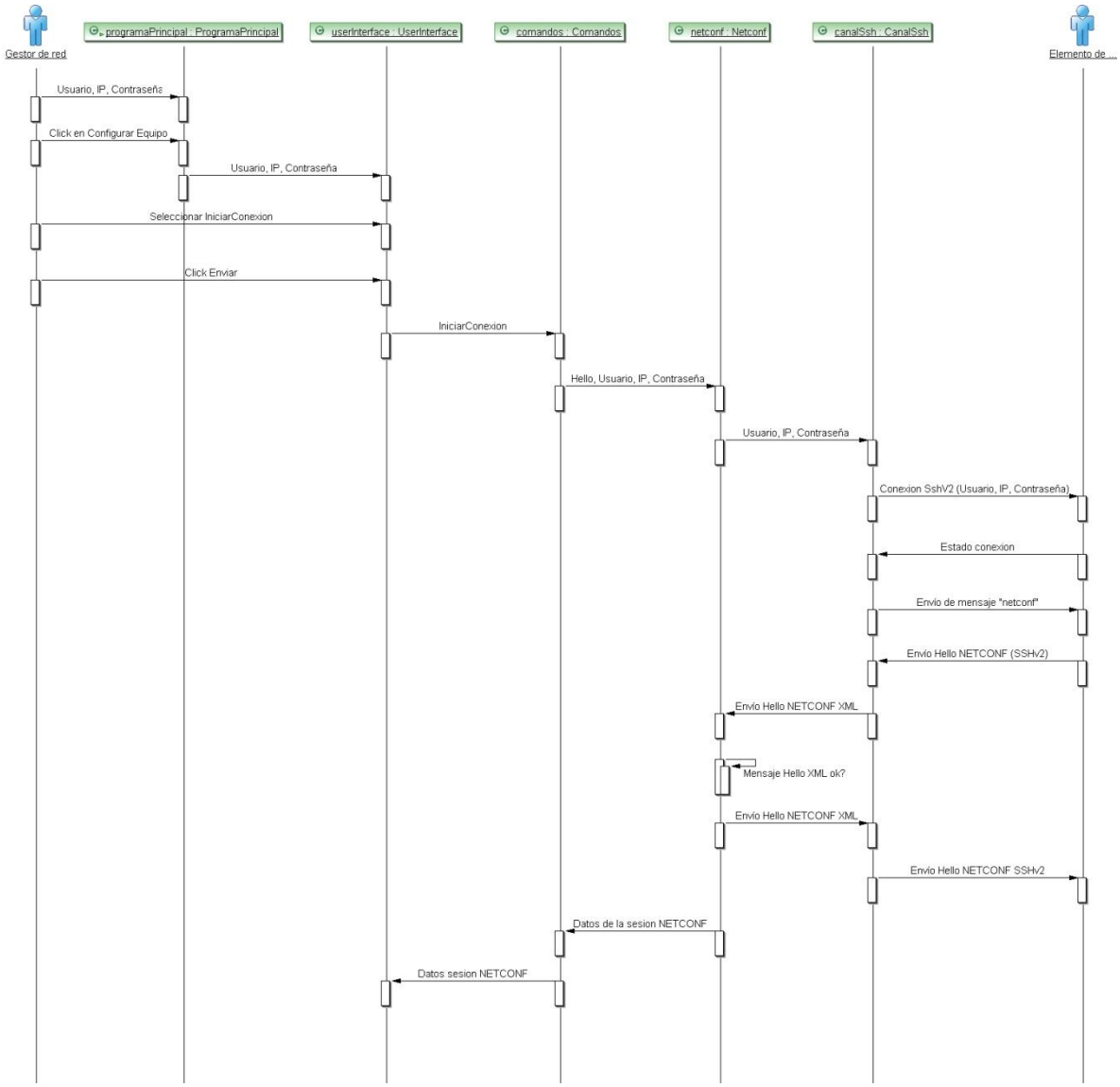


**Figura 15. Diagrama de secuencia de inicio de conexión del prototipo de software de gestión. Tomado de (17)**

En la Figura 16 se muestra el diagrama de secuencia del envío de operaciones desde el prototipo de software de gestión al equipo de red.

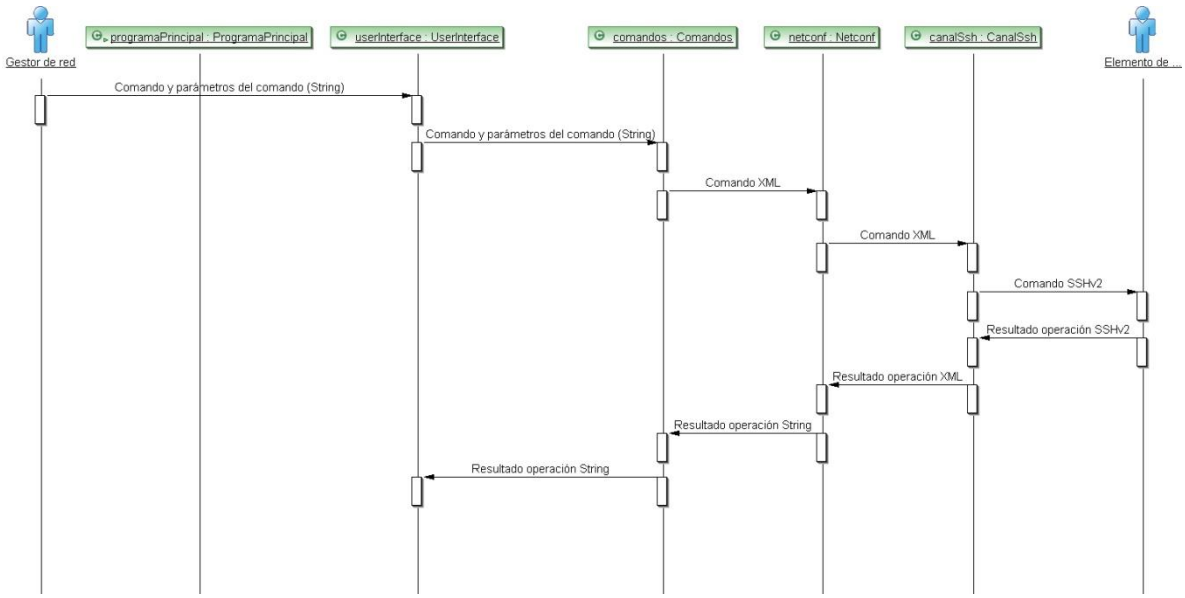


**Figura 16. Diagrama de secuencia de las operaciones NETCONF del prototipo de software de gestión. Tomado de (17)**

## 4.5. Scripts de preconfiguración de los Routers que se desean gestionar con el prototipo de la plataforma de gestión

Para que un usuario pueda gestionar equipos Cisco y Juniper utilizando este prototipo de gestión, tal y como se describió en la sección de especificaciones del presente documento, es necesario que se realice una preconfiguración mínima de estos equipos. A continuación se relaciona un script de configuración (a modo de ejemplo), de cada uno de estos equipos.

### 4.5.1. Script de configuración Router Cisco (21)

Para que un router cisco tome la configuración de netconf, es necesario que el IOS ser igual ó superior a la versión 12.2IRA(22).

conf t

hostname javeriana

ip domain-name netconfyang

```
crypto key generate rsa
800
ip ssh time-out 60
ip ssh authentication-retries 2
ip ssh version 2
username telmex secret telmex1
line vty 0 4
transport input ssh
login local
exit
username telmex privilege 15 secret telmex1
access-list 10 permit 10.29.1.0 0.0.0.255
netconf ssh acl 10
netconf lock-time 30
netconf max-sessions 4
int f1/0
ip address 10.29.1.1 255.255.255.0
no shut
```

**4.5.2.** Script de configuración Router Juniper

```
root# set system root-authentication encrypted-password juniper

[edit]
root# commit

commit complete

[edit]
root# set system login user javeriana class super-user
```

[edit]
root# set system login user javeriana authentication plain-text-password
New password: ---> telmex1
Retype new password: ---> telmex1

[edit]
root#set interfaces em0 unit 0 family inet address 10.29.1.3/24

[edit]
root#
[edit]
root#set system services ssh

[edit]
root#edit system services
root#set netconf ssh
[edit system services]
root#commit

### 4.6. Implementación topología de red para verificación de funcionalidad del prototipo de software de gestión

Ya realizadas las anteriores actividades de desarrollo e implementación del prototipo de software de gestión, y preconfiguración de equipos de red de los proveedores Cisco y Juniper para soportar conexiones SSHv2, un usuario podrá realizar las operaciones descritas en las especificaciones del prototipo desarrollado que se encuentran en la sección 3 del presente documento. Así, para efectos de desarrollo y validación de la funcionalidad del prototipo realizado, se utilizaron routers de los proveedores Cisco y Juniper, los cuales estuvieron en funcionamiento en la plataforma de simulación gns3. En la Figura 17 se muestra la topología de red utilizada para la gestión de estos equipos haciendo uso del prototipo desarrollado:

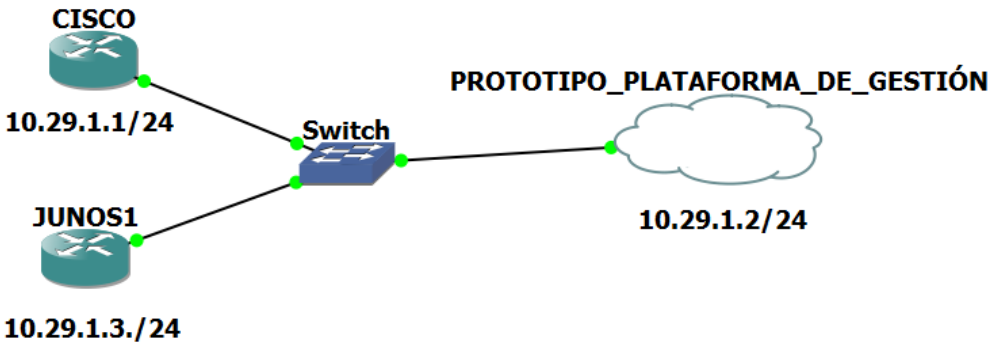


**Figura 17. Topología de la red gestionada utilizando el prototipo de gestión. Tomado de (17)**

Se observa que se tienen dos router conectados a una nube, la cual es la interface virtual entre el programa gns3 y el computador que contiene el prototipo de la plaforma de gestión de red. Es indispensable que el equipo gestionado y la interfaz virtual estén en la misma red para que se establezca conexión. Si se desea gestionar equipos que estén en diferentes IP, es necesario incluir a la topología de la Figura 17, un router que se encargue de realizar el enrutamiento. Cada uno de los equipos gestionados funcionan con una imagen de un equipo real, es decir, en el caso del router Cisco la imagen con la que opera es un IOS real (15.1(4) M5 Ver Figura 18), y en el caso del router juniper, la imagen con la que opera es un JUNOS real (12.1R1.9 Ver Figura 19).

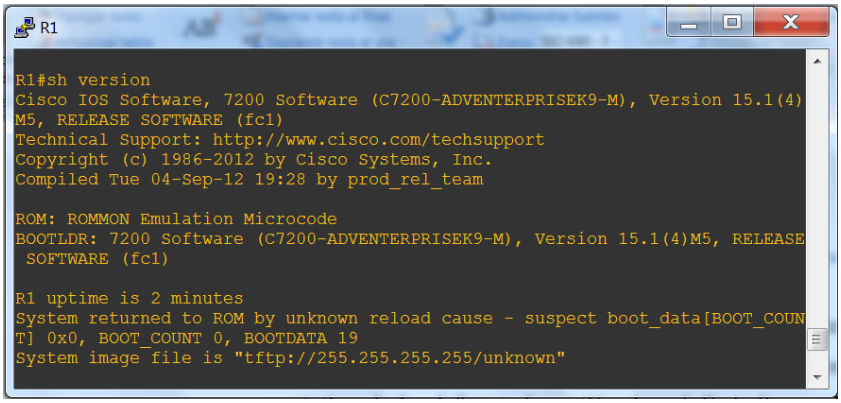


**Figura 18. Consola del equipo Cisco donde se visualiza versión IOS del equipo.**

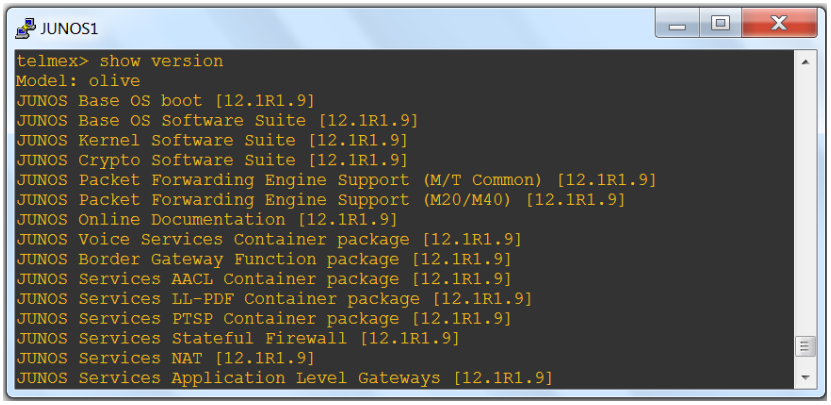


**Figura 19. Consola del equipo Juniper donde se visualiza versión de JUNOS del equipo**

# 5. ANÁLISIS DE RESULTADOS

A continuación se especifican las diferentes pruebas de validación que se realizaron para comprobar la funcionalidad del prototipo de la plataforma de gestión de red basada en el protocolo de gestión de red NETCONF, y en el lenguaje de modelado de datos YANG. De esta manera, con las siguientes pruebas de validación, se comprueba el cumplimiento de los objetivos presentados en el documento del anteproyecto.

## 5.1. Pruebas de validación del prototipo de software de gestión

### 5.1.1. Visualización ventana principal y ventana de configuración del equipo

Cuando se ejecuta el programa principal del proyecto en eclipse, se despliega la ventana de la Figura 20.

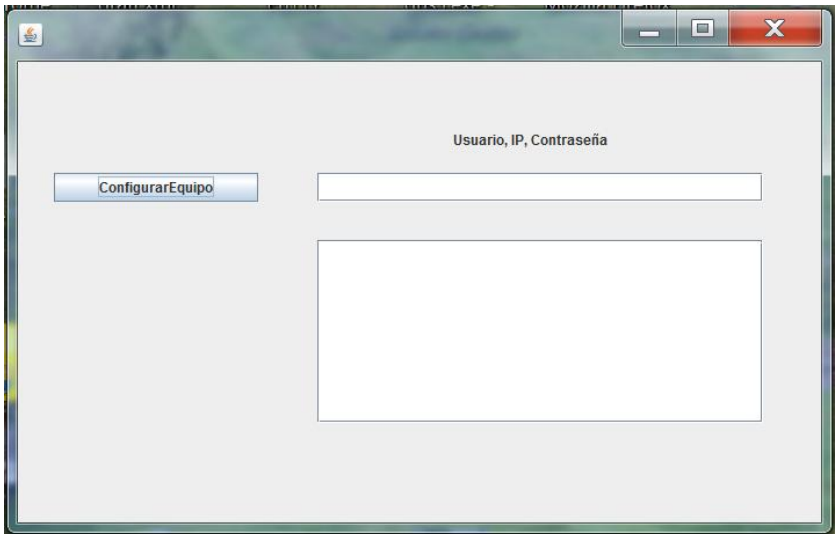


**Figura 20. Interfaz gráfica principal para indicar parámetros del dispositivo con el que se desea establecer conexión. Tomado de (17).**

Cuando se ingresa el Usuario, IP, y Contraseña en el campo de texto correspondiente, y se le indica al prototipo de software de gestión que se desea establecer conexión con este elemento de red, si estos parámetros están incompletos, se mostrará en la consola el siguiente mensaje: "Para continuar, por favor ingrese en la caja de texto superior, todos los parámetros correctamente en el siguiente orden: Usuario, IP y Contraseña. Sepárelos con comas. Gracias" Ver Figura 21.

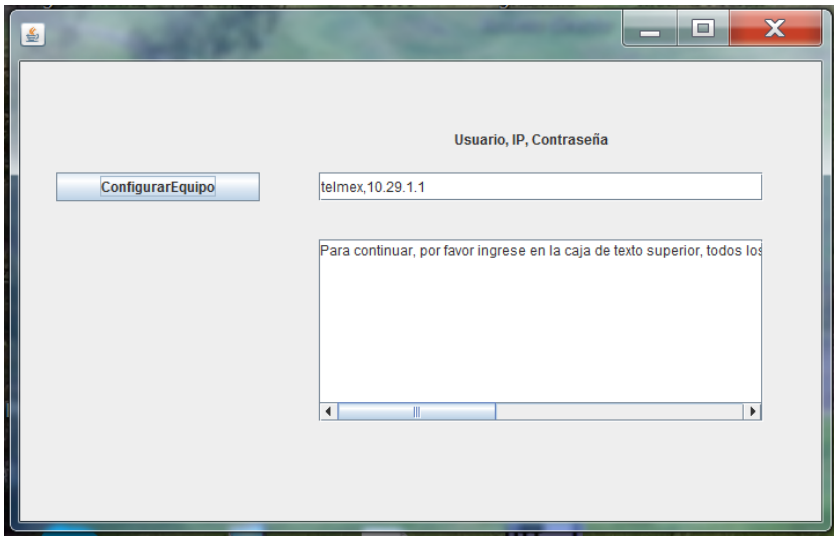


**Figura 21. Mensaje de error de la ventana principal del prototipo de gestión.**

Para leer todo el mensaje, es necesario desplazar el *scroll bar* de la caja de texto. Pero cuando se ingresa en la ventana principal todos los parámetros correctamente en el orden: Usuario, IP y Contraseña, y se pulsa el botón "Configurar Equipo", se mostrará el siguiente mensaje de texto: "Ahora puede configurar el equipo en la otra ventana", e inmediatamente se abrirá otra ventana a la que se llamará "ventana de configuración del equipo" Ver **¡Error! No se encuentra el origen de la referencia.**.

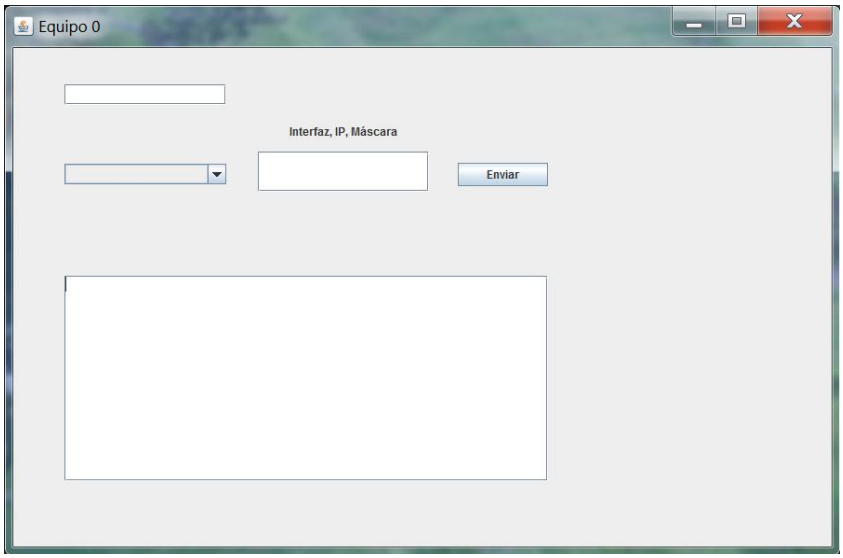


**Figura 22. Ventana de configuración del equipo. Tomado de (17)**

### 5.1.2. Envío de mensaje Hello para inicio de sesión netconf

Continuando con el proceso del numeral anterior, hasta el momento el cliente no ha establecido conexión con el servidor, sino que simplemente se le indicó al prototipo de software de gestión el equipo de red que se desea gestionar. Para establecer conexión con este elemento de red, es necesario escoger en la ventana de configuración del equipo la opción "Iniciar sesión" de la lista desplegable y pulsar el botón "Enviar". Realizado este procedimiento, el cliente iniciará la conexión con el servidor haciendo uso de los parámetros "Usuario", "IP" y "Contraseña" ingresados por el usuario en la ventana principal del prototipo de gestión. Cuando la conexión se establece, el cliente le envía al servidor su mensaje Hello Anexo 5. Igualmente, tan pronto la conexión se establece, el servidor le envía al cliente su mensaje Hello donde anuncia las capacidades del dispositivo. Si la versión netconf anunciada en los mensajes Hello es la misma, la sesión netconf se establece.

A continuación se lista las pruebas de validación para el inicio de sesión con los routers Cisco y Juniper

### 5.1.2.1. Inicio de sesión con router Cisco

Haciendo uso de la herramienta de análisis de tráfico Wireshark, se verifica el tráfico entre el cliente y el servidor en el inicio de conexión SSHv2 entre el cliente y el servidor. En la Figura 23 se observa el intercambio de mensajes ssh entre estos pares:

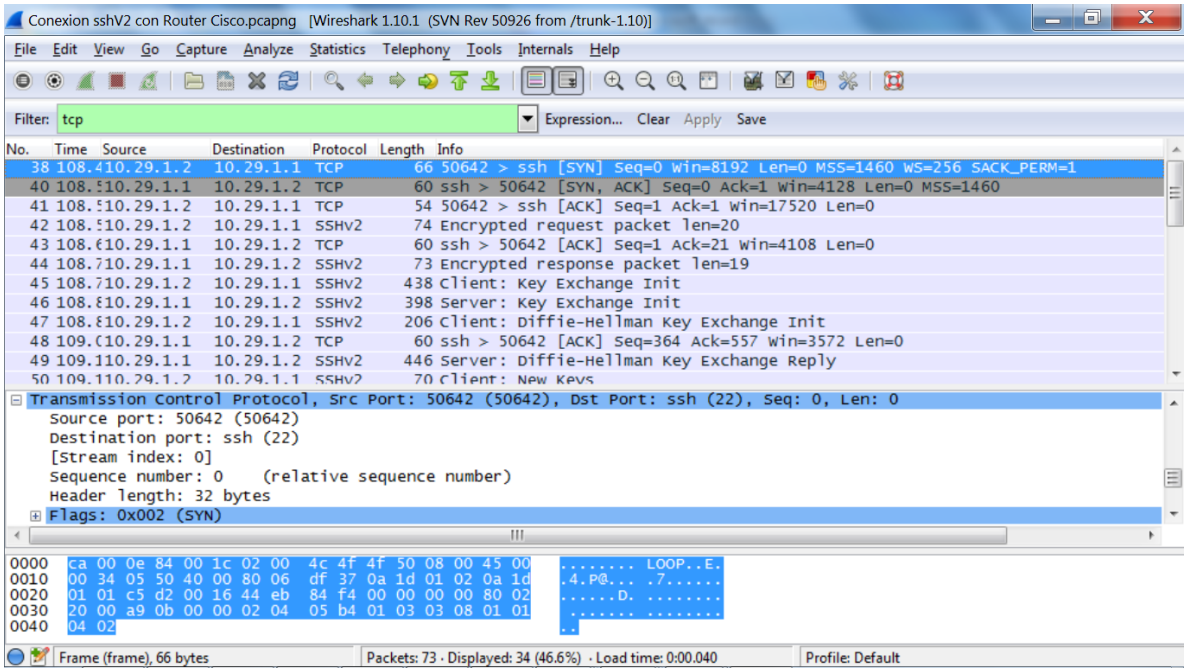


**Figura 23. Intercambio de mensajes ssh V2 durante el inicio de conexión entre el cliente y el router Cisco.**

En la Figura 24, se muestra la activación en el Router Cisco de la captura de eventos netconf, y se observa que se recibe el mensaje Hello enviado por el cliente. Ver Anexo 5.

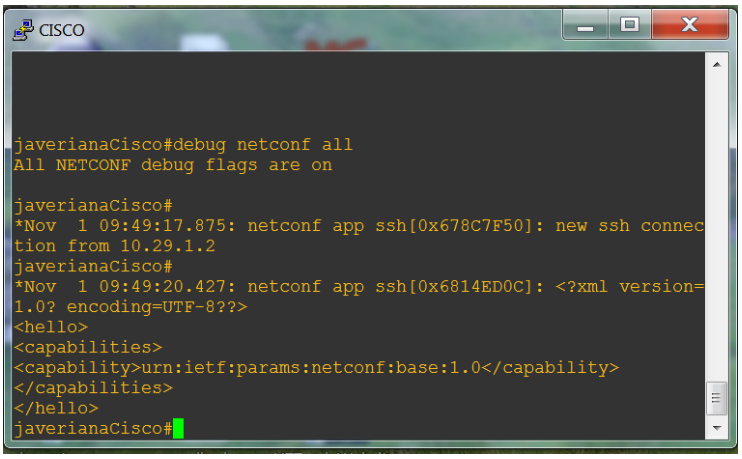


Figura 24. Consola equipo Cisco

En la **¡Error! No se encuentra el origen de la referencia.**, se muestra en la consola de la ventana de configuración del equipo, el mensaje que le indica al usuario que la sesión netconf se ha establecido exitosamente.

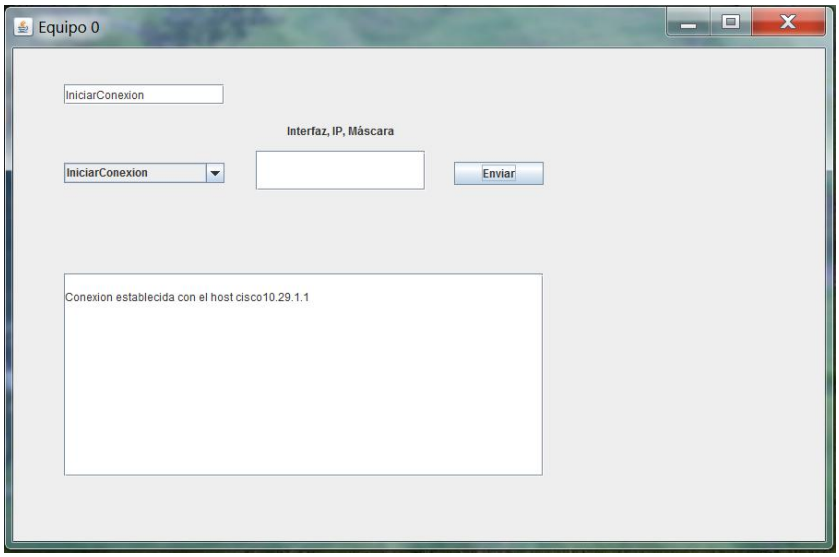


Figura 25. Ventana de configuración del equipo en la que la consola le indica al usuario del éxito de la conexión. Tomado de (17)

En el Hello que envía el servidor al cliente, se indica el ID de la sesión(Ver Anexo 6). Ya establecida la sesión netconf, el usuario puede realizar las operaciones de configuración descritas en las especificaciones

del prototipo desarrollado que se encuentran en la sección 3 del presente documento.

#### 5.1.2.2. Inicio de sesión con router Juniper

Haciendo uso de la herramienta de análisis de tráfico Wireshark, se verifica el tráfico entre el cliente y el servidor en el inicio de conexión SSHv2 entre el cliente y el servidor. En la Figura 26 se observa el intercambio de mensajes ssh entre estos pares:

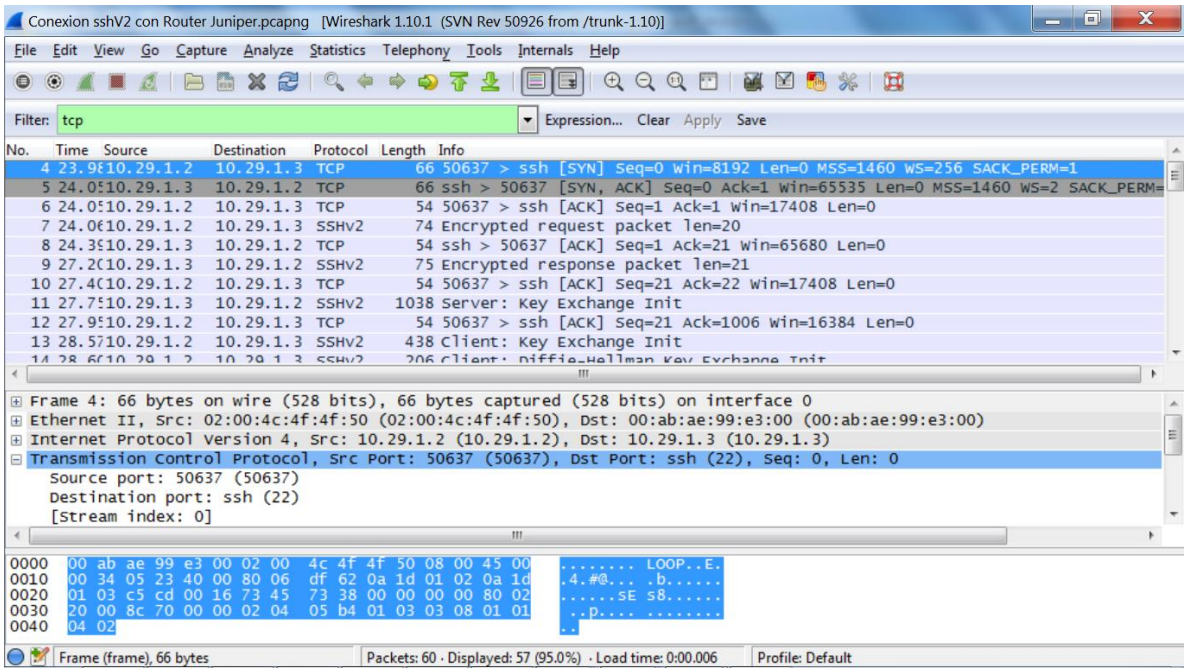


**Figura 26. Intercambio de mensajes ssh V2 durante el inicio de conexión entre el cliente y el router Juniper**

Tal y como ocurrió con el caso anterior, cuando se ingresa todos los parámetros correctamente en el orden: Usuario, IP y Contraseña correspondientes al router Juniper, y se pulsa el botón "Configurar Equipo", se mostrará el siguiente mensaje de texto: "Ahora puede configurar el equipo en la otra ventana" y se abrirá otra ventana de configuración del equipo. En la Figura 27 se observa el mensaje de consola en la ventana principal del prototipo de gestión, que indica que ya se puede configurar el equipo en la nueva ventana de configuración del equipo.

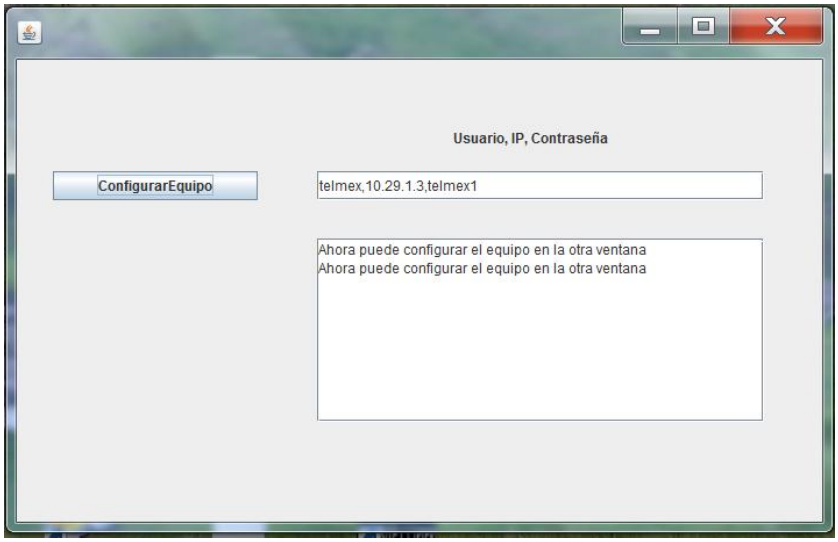


**Figura 27. Ventana principal del prototipo de gestión en la que el mensaje de consola indica que se puede realizar la configuración del equipo en la nueva ventana abierta de configuración del equipo. Tomado de (17)**

Al igual que se hizo con el router Cisco, para que se establezca la sesión netconf con el router Juniper, se requiere se escoja de la lista desplegable de la ventana de configuración del equipo (Ver **¡Error! No se encuentra el origen de la referencia.**), la opción "Iniciar sesión" y pulsar el botón "Enviar". Realizado este procedimiento, el cliente iniciará la conexión con el servidor haciendo uso de los parámetros "Usuario", "IP" y "Contraseña" ingresados por el usuario en la ventana principal del prototipo de gestión. Cuando la conexión se establece, el cliente le envía al servidor su mensaje Hello Anexo 5. Igualmente, tan pronto la conexión se establece, el servidor le envía al cliente su mensaje Hello donde anuncia las capacidades del dispositivo. Si la versión netconf anunciada en los mensajes Hello es la misma, la sesión netconf se establece. En el Hello que envía el servidor al cliente, se indica el ID de la sesión(Ver Anexo 7). Ya establecida la sesión netconf, el usuario puede realizar las operaciones de configuración descritas en las especificaciones del prototipo desarrollado que están en la sección 3 del presente documento.

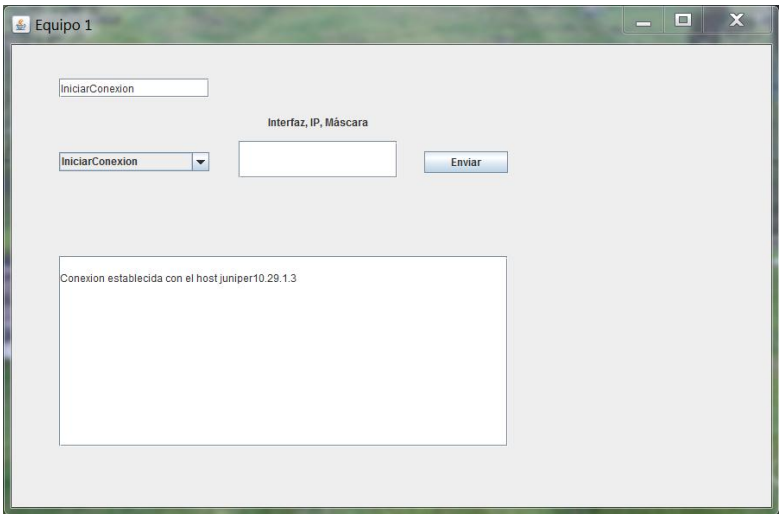


**Figura 28. Ventana de configuración del equipo en la que la consola le indica al usuario del éxito de la conexión. Tomado de (17)**

### 5.1.3. Validación de las RPC de configuración teniendo en cuenta los esquemas de los equipos

La configuración de los equipos de red se hace mediante el envío de RPCs, por lo que a continuación se enumeran las pruebas de validación que se realizaron sobre las mismas teniendo en cuenta los esquemas de los equipos.

#### 5.1.3.1. Esquema Router Cisco

Se observa que las operaciones de configuración enviadas al router Cisco (Ver Anexo 8 y Anexo 9), cumplen con la jerarquía de configuración establecida en el esquema netconf del router Cisco (Ver Anexo 11)

#### 5.1.3.2. Esquema Router Juniper

Se observa que las operaciones de configuración enviadas al router Juniper (Ver Anexo 8 y Anexo 10), cumplen con la jerarquía de configuración de interface establecida en el esquema del router Juniper (Ver Anexo 12)

### 5.1.4. Validación de las RPCs teniendo en cuenta el esquema de modelado de datos yang

La validación de las rpc-request enviadas al servidor (router), no va ligada al proceso de configuración de un elemento de red, sino que sirve para comprobar que estas RPC cumplen con las especificaciones establecidas en los RFC6241 y RFC6020. En el caso de las rpc-reply provenientes del servidor y con destino el cliente, éstas si son analizadas por el prototipo de gestión, ya que se deberá corroborar que estén bien formadas y que cumplen con el esquema de datos definido para la capa de mensajes del protocolo NETCONF.

A continuación se muestran las pruebas realizadas sobre las rpc-request enviadas al servidor, y de las rpc-reply recibidas del servidor, ello utilizando dos herramientas de validación de archivos xml con base a un esquema de datos, que en este caso, es el esquema de datos yang para el protocolo netconf (Ver Anexo 21 y Anexo 22).

#### 5.1.4.1. Validación RPCs utilizando notepad ++

Para la validación de archivos xml con base a un esquema de datos determinado, es necesario que el programa notepad++ tenga instalada la herramienta XML Tools.

##### 5.1.4.1.1. Validación de la rpc-request get-config

En la figura Figura 29 se muestra la escogencia en el programa notepad ++, del esquema del modelo de datos de las operaciones del protocolo netconf (Ver Anexo 21), dentro del cual, se realiza la importación del esquema del modelo de datos de la capa de mensajes netconf (Ver Anexo 22). El primer esquema se encuentra almacenado en un archivo llamado netconf_00.xsd y el segundo se encuentra almacenado en un archivo llamado ietf-netconf.xsd. Estos dos esquemas, se utilizan para realizar la validación de las distintas RPCs.

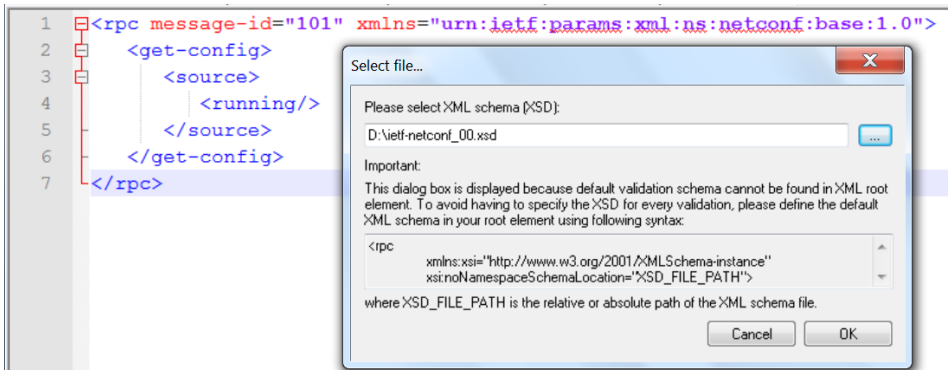


**Figura 29. Configuración de la herramienta XML Tool de notepad++ para que utilice el esquema de datos netconf para analizar el contenido de las RPCs**

En la Figura 30 se muestra el resultado de la validación de la rpc get-config, y que es utilizada por el prototipo de la plataforma de gestión para realizar la operación "Obtener toda la configuracion".

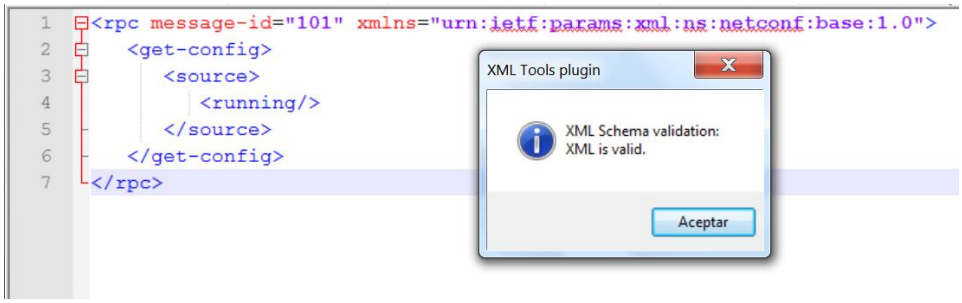


**Figura 30. Validación de la rpc get-config utilizando la herramienta XML Tool del programa notepad++**

**5.1.4.1.2.** Validación de la rpc-reply enviada por el servidor como respuesta a la rpc-request get-config

En la Figura 31 se observa el resultado de la validación de la rpc-reply enviada por el router Cisco como respuesta a la rpc-request get-config.

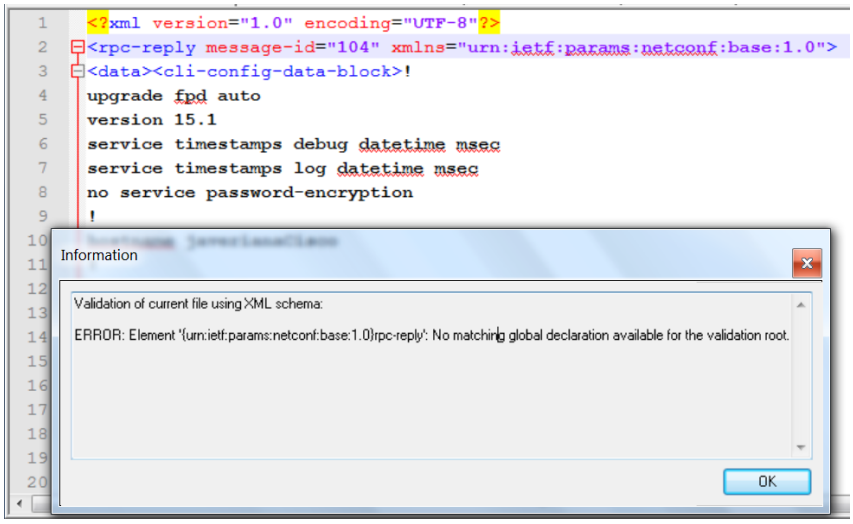


**Figura 31. Validación con notepad++ de la rpc-reply enviada por un router Cisco, como respuesta a la rpc-request get-config proveniente del cliente.**

En esta Figura 31 se observa que el esquema del modelo de datos yang para las operaciones del protocolo netconf (Ver Anexo 21), no valida esta respuesta, debido a que los esquemas del modelo de datos de las capas de operación y de mensajes netconf (Ver Anexo 21 y Anexo 22), tienen como xmlns "urn:ietf:params:xml:ns:netconf:base:1.0" y no el indicado en la rpc-reply que envía el Router Cisco y el cual es "urn:ietf:params:netconf:base:1.0". En dado caso que la rpc-reply del Router Cisco indicara correctamente el xmlns, tampoco se podría realizar la validación de la rpc, ya que el esquema del modelo de datos de la capa de mensajes del protocolo netconf, no cuenta con una opción que valide que dentro de una rpc-reply pueda existir una jerarquía <data>. Se recuerda que el esquema del modelo de datos yang para las operaciones del protocolo netconf, importa el modelo de datos de la capa de mensajes del protocolo netconf, de tal forma que estos dos esquemas son utilizados para la validación de las RPCs intercambiadas entre el cliente y el servidor. Así, en el modelo de datos de la capa de mensajes del protocolo netconf, para el

elemento rpc-reply, los únicos datos válidos son el elemento ok y el elemento rpc-error.

En la Figura 32 se observa el resultado de la validación de la rpc-reply enviada por el router Juniper como respuesta a la rpc-request edit-config

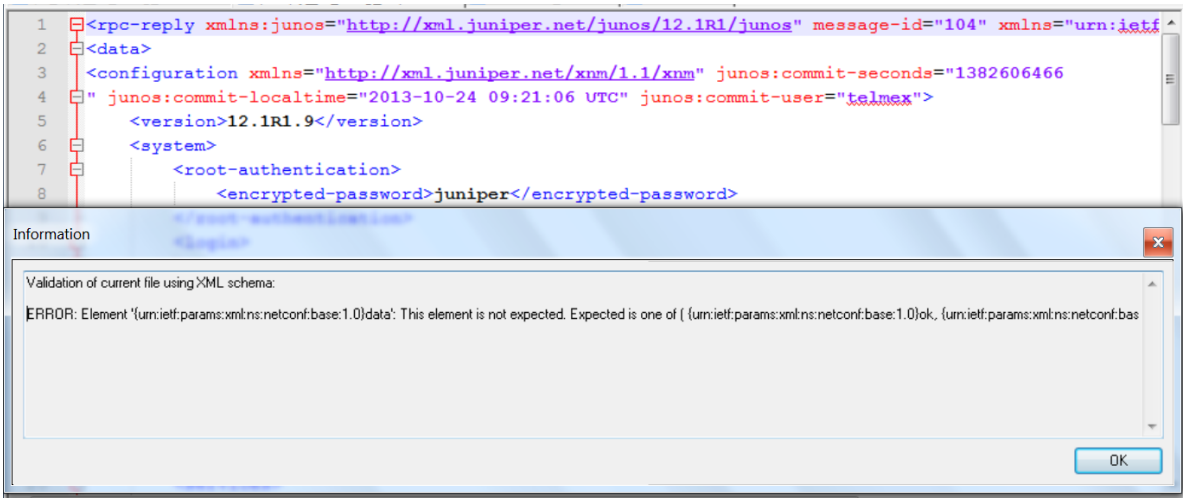


**Figura 32. Validación con notepad++ de la rpc-reply enviada por un router Juniper, como respuesta a la rpc-request get-config proveniente del cliente.**

Tal y como se explicó en el caso de la validación del rpc-reply de la operación get-config del router Cisco, el modelo de datos yang para las operaciones del protocolo netconf y que importa el modelo de datos de la capa de mensajes del protocolo netconf, no valida el rpc-reply de la operación get-config del router Juniper, debido a que en el modelo de datos correspondiente, no se encuentra una jerarquía con la opción de <data>, por lo que este rpc-reply es desconocido para los esquemas de datos de las capas de operación y mensajes del protocolo netconf.

#### 5.1.4.1.3. Validación de la rpc-request edit-config

En la Figura 33 se muestra el resultado de la validación de la rpc edit-config, utilizada por el prototipo de la plataforma de gestión

para realizar la operación “Configurar interface” en un equipo Cisco.

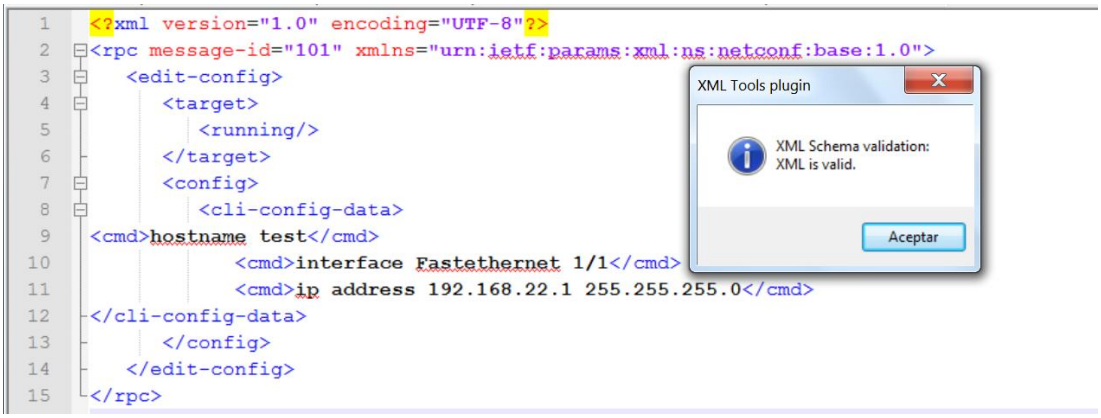


Figura 33. Validación con notepad++ de la rpc edit-config para un router Cisco

En la Figura 34 se muestra el resultado de la validación de la rpc edit-config, utilizada por el prototipo de la plataforma de gestión, para realizar la operación “Obtener toda la configuracion” en un equipo Juniper.

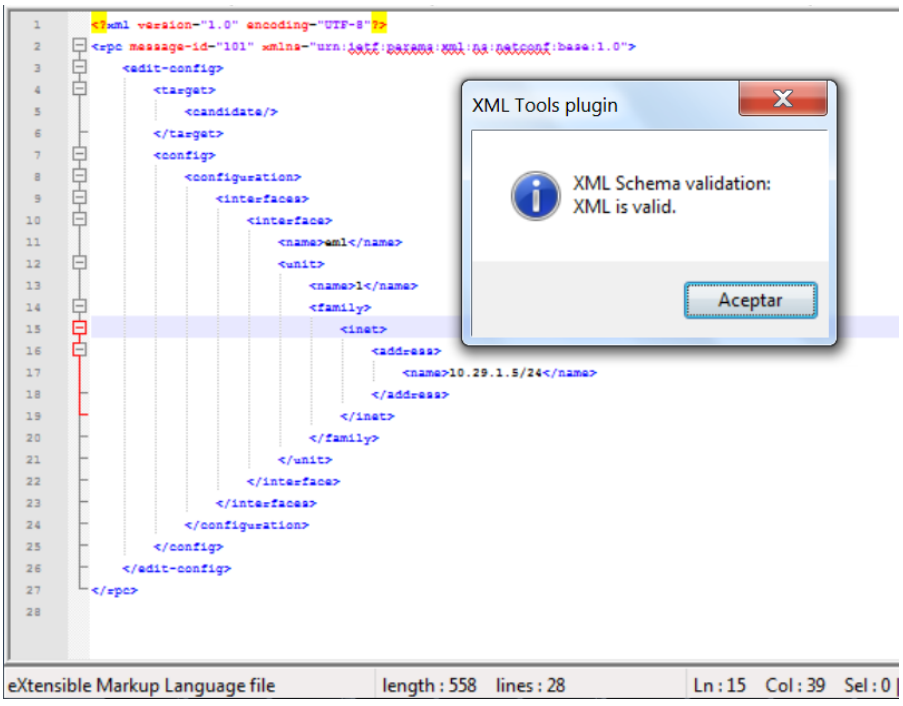


Figura 34. Validación con notepad++ de la rpc edit-config para un router Juniper

**5.1.4.1.4.** Validación de la rpc-reply enviada por el servidor como respuesta a la rpc-request edit-config

En la Figura 35 se observa el resultado de la validación de la rpc-reply enviada por el router Cisco como respuesta a la rpc-request edit-config.

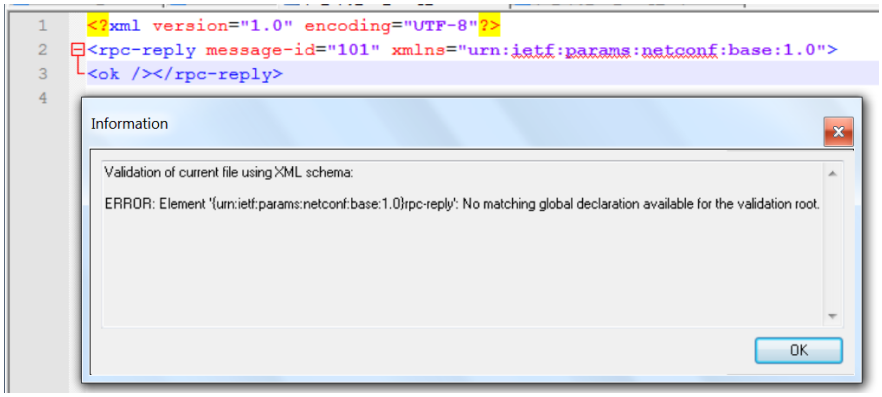


**Figura 35. Validación con notepad++ de la rpc-reply enviada por un router Cisco, como respuesta a la rpc-request edit-config proveniente del cliente.**

Como ya se explicó para las rpc-reply de las operaciones get-config, en esta Figura 35, se observa que los esquemas del modelo de datos de las capas de operación y de mensajes netconf (Ver Anexo 21 y Anexo 22), no validan esta rpc-reply, debido a que los esquemas tienen como xmlns "urn:ietf:params:xml:ns:netconf:base:1.0" y no el indicado en la rpc-reply que envía el router Cisco y el cual es "urn:ietf:params:netconf:base:1.0".

En la Figura 36 se observa el resultado de la validación de la rpc-reply enviada por el router Juniper como respuesta a la rpc-request edit-config

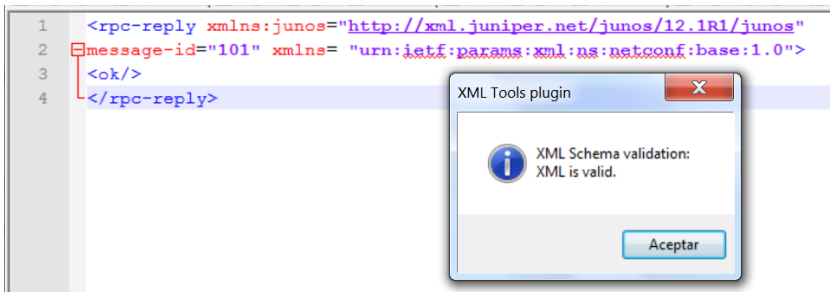


**Figura 36. Validación con notepad++ de la rpc-reply enviada por un router Juniper, como respuesta a la rpc-request edit-config proveniente del cliente.**

#### 5.1.4.1.5. Validación de la rpc-request closse-session

En la Figura 37 se muestra el resultado de la validación de la rpc close-session, y que es utilizada por el prototipo de la plataforma de gestión para realizar la operación “Cerrar sesion”.

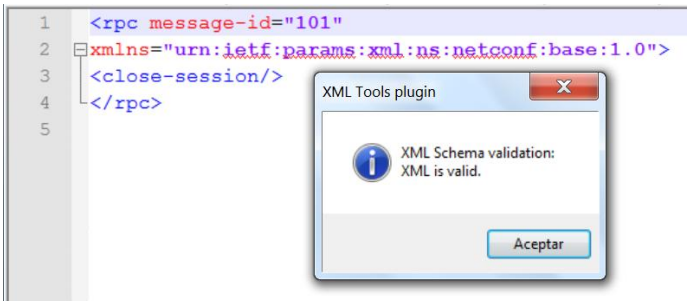


**Figura 37. Validación con notepad++ de la rpc close-session enviada a los servidores Cisco y Juniper.**

#### 5.1.4.1.6. Validación de la rpc-reply enviada por un servidor, como respuesta a una rpc-request close-session

En la Figura 38 se observa el resultado de la validación de la rpc-reply enviada por el router Cisco como respuesta a la rpc-request close-session.

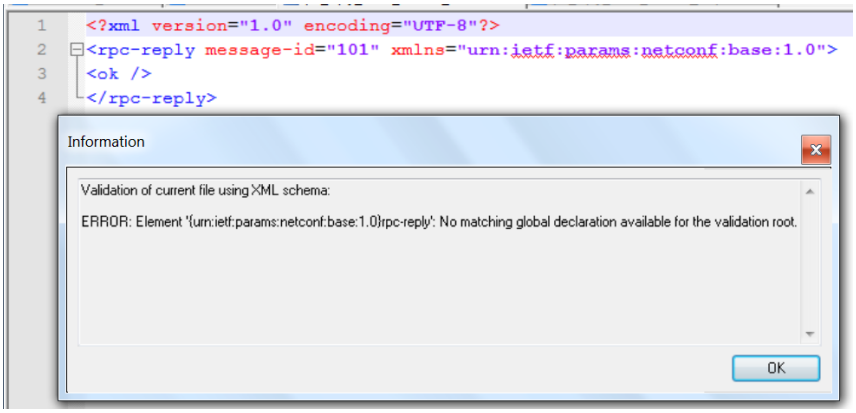


**Figura 38. Validación con notepad++ de la rpc-reply enviada por un router Cisco, como respuesta a la rpc-request close-session proveniente del cliente.**

Como ya se explicó para las rpc-reply de las operaciones get-config y edit-config, en esta Figura 38, se observa que los esquemas del modelo de datos de las capas de operación y de mensajes netconf (Ver Anexo 21 y Anexo 22), no validan esta rpc-reply, debido a que

los esquemas tienen como xmlns "urn:ietf:params:xml:ns:netconf:base:1.0" y no el indicado en la rpc-reply que envía el router Cisco y el cual es "urn:ietf:params:netconf:base:1.0".

En la Figura 39 se observa el resultado de la validación de la rpc-reply enviada por el router Juniper como respuesta a la rpc-request close-session.

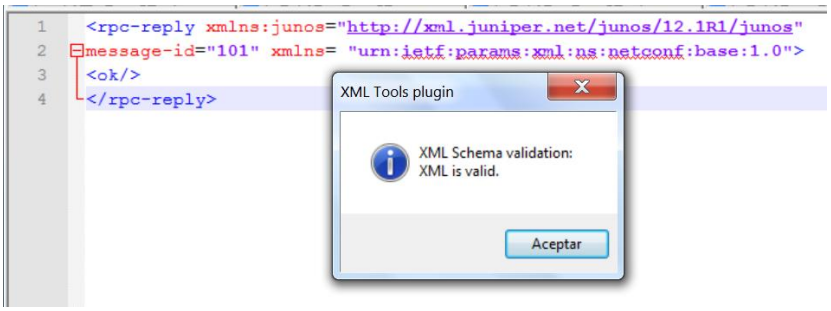


**Figura 39. Validación con notepad++ de la rpc-reply enviada por un router Juniper, como respuesta a la rpc-request close-session proveniente del cliente.**

#### 5.1.4.2. Validación RPCs utilizando módulo YANG Parser de java

La validación de las RPCs se hace utilizando el módulo Yang Parser del prototipo de gestión desarrollado. Este módulo realiza la validación de las distintas RPCs, teniendo en cuenta el esquema del modelo de datos de las operaciones del protocolo netconf (Ver Anexo 21). Dentro de este esquema, se realiza la importación del esquema del modelo de datos de la capa de operaciones netconf (Ver Anexo 22), dentro del cual se encuentran los distintos tipos válidos de datos.

##### 5.1.4.2.1. Validación del rpc-request get-config

En la Figura 40 se muestra el resultado de la validación de la rpc get-config utilizando el módulo YANG Parser de java, y que es utilizada por el prototipo de la plataforma de gestión para realizar la operación "Obtener toda la configuracion".

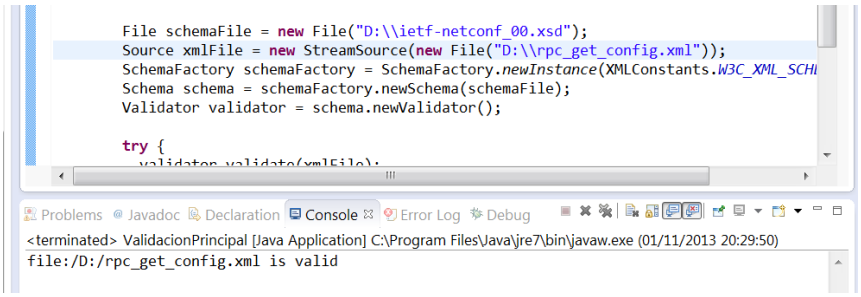


**Figura 40. Validación con eclipse de la rpc get-config.**

#### 5.1.4.2.2. Validación de la rpc-reply enviada por el servidor como respuesta a la rpc-request get-config

En la Figura 41 se observa el resultado de la validación de la rpc-reply enviada por el router Cisco como respuesta a la rpc-request get-config.

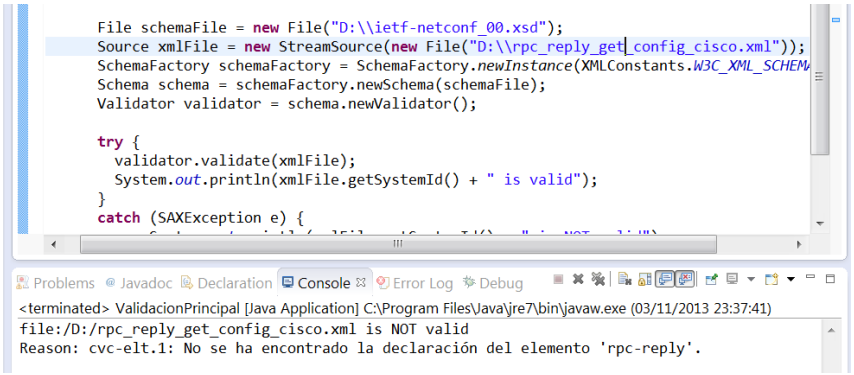


**Figura 41. Validación con eclipse de la rpc-reply enviada por el router Cisco, como respuesta de la rpc-request get-config enviada por el cliente.**

Las razones por las cuales esta rpc-reply no es validada, son las mismas razones que se expusieron en las validaciones de esta misma rpc-reply con la herramienta notepad ++ (Ver numeral 5.1.4.1.2 del presente documento).

En la Figura 42 se observa el resultado de la validación de la rpc-reply enviada por el router Juniper como respuesta a la rpc-request get-config.

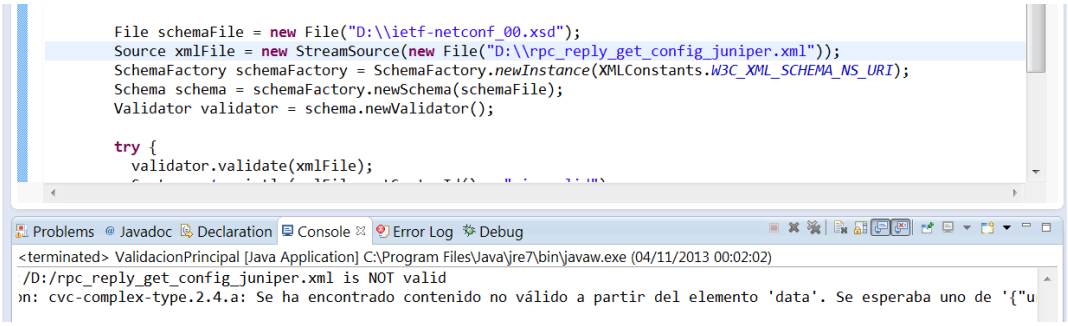


**Figura 42. Validación con eclipse de la rpc-reply enviada por el router Juniper, como respuesta de la rpc-request get-config enviada por el cliente.**

#### 5.1.4.2.3. Validación de la rpc-request edit-config

En la Figura 43 se muestra el resultado de la validación de la rpc edit-config, utilizada por el prototipo de la plataforma de gestión, para realizar la operación "Configurar interface" en un equipo Cisco.

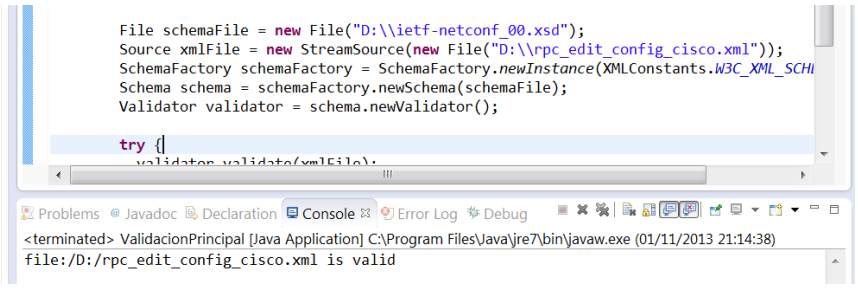


**Figura 43. Validación con eclipse de la rpc edit-config para un router Cisco.**

En la Figura 44 se muestra el resultado de la validación de la rpc edit-config, utilizada por el prototipo de la plataforma de gestión, para realizar la operación "Configurar interface" en un equipo Juniper.

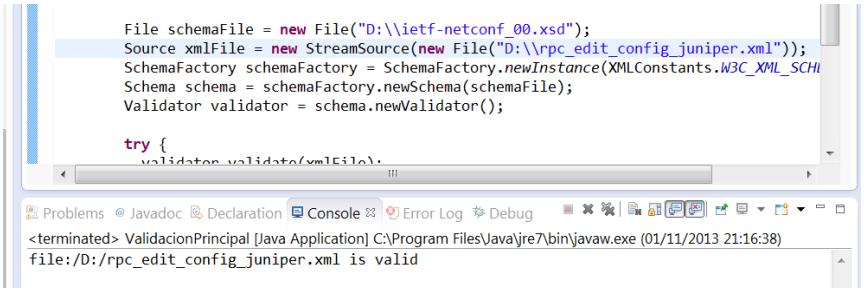


**Figura 44. Validación con eclipse de la rpc edit-config para un router Juniper.**

**5.1.4.2.4.** Validación de la rpc-reply enviada por el servidor como respuesta a la rpc-request edit-config

En la Figura 45 se observa el resultado de la validación de la rpc-reply enviada por el router Cisco como respuesta a la rpc-request edit-config.

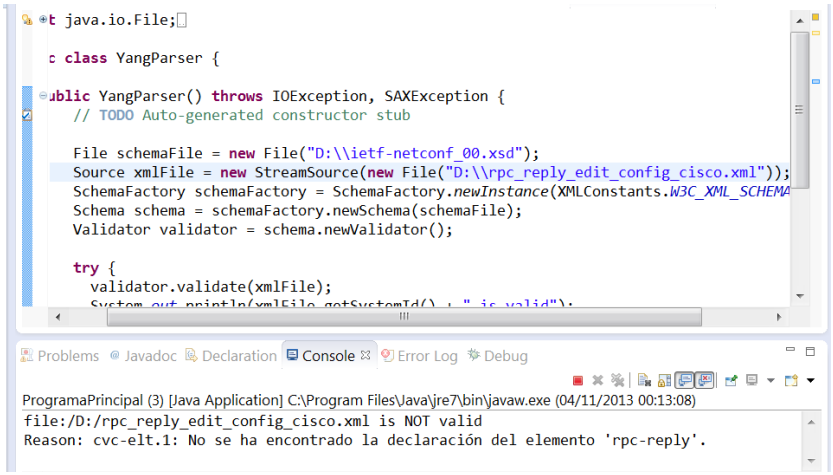


**Figura 45. Validación con eclipse de la rpc-reply enviada por el router Cisco, como respuesta de la rpc-request edit-config enviada por el cliente.**

Las razones por las cuales esta rpc-reply no es validada, son las mismas razones que se expusieron en las validaciones de esta misma rpc-reply con la herramienta notepad ++ (Ver el numeral 5.1.4.1.4 del presente documento).

En la Figura 46 se observa el resultado de la validación de la rpc-reply enviada por el router Juniper como respuesta a la rpc-request edit-config.

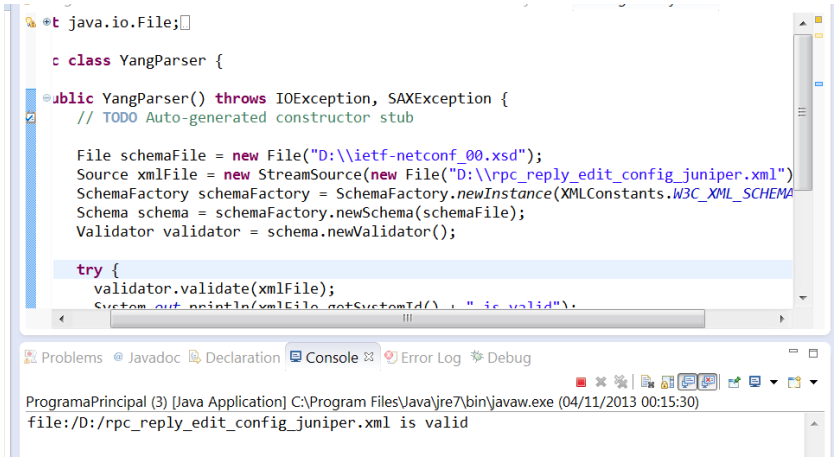


**Figura 46. Validación con eclipse de la rpc-reply enviada por el router Juniper, como respuesta de la rpc-request edit-config enviada por el cliente.**

#### 5.1.4.2.5. Validación de la rpc-request close-session

En la Figura 47se observa el resultado de la validación de la rpc close-session, utilizada por el prototipo de la plataforma de gestión para realizar la operación "Cerrar sesión" con los equipos Cisco y Juniper.

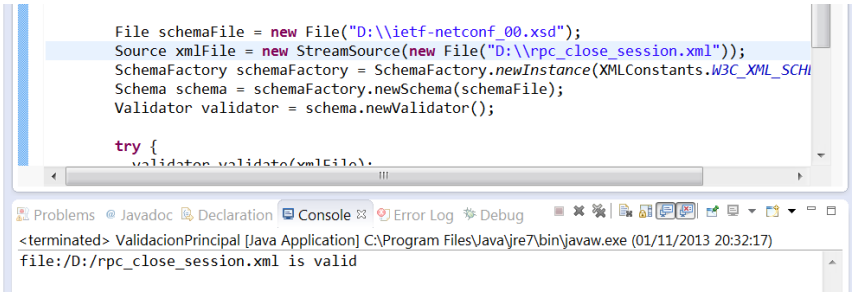


**Figura 47. Validación con eclipse de la rpc close-session.**

#### 5.1.4.2.6. Validación de la rpc-reply de una solicitud rpc-request closse-session

En la Figura 48 se observa el resultado de la validación de la rpc-reply enviada por el router Cisco como respuesta a la rpc-request close-session.

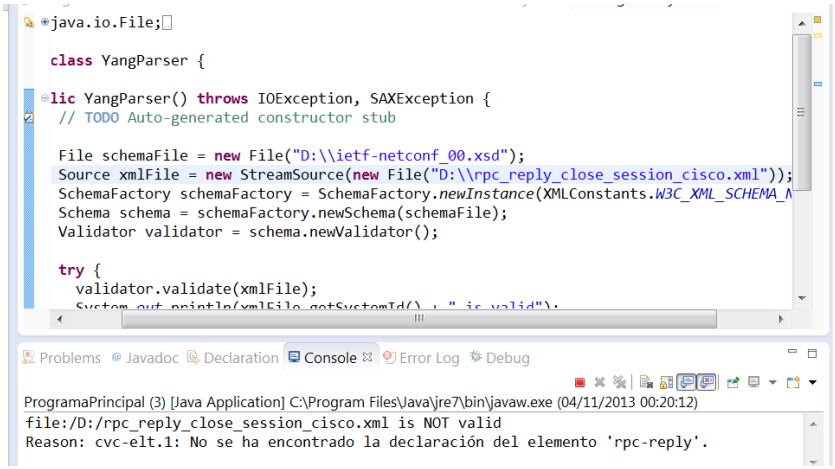


**Figura 48. Validación con eclipse de la rpc-reply enviada por el router Cisco, como respuesta de la rpc-request close-session enviada por el cliente.**

Las razones por las cuales esta rpc-reply no es validada, son las mismas razones que se expusieron en las validaciones de esta misma rpc-reply con la herramienta notepad ++ (Ver el numeral 5.1.4.1.6 del presente documento).

En la Figura 49 se observa el resultado de la validación de la rpc-reply enviada por el router Juniper como respuesta a la rpc-request close-session.

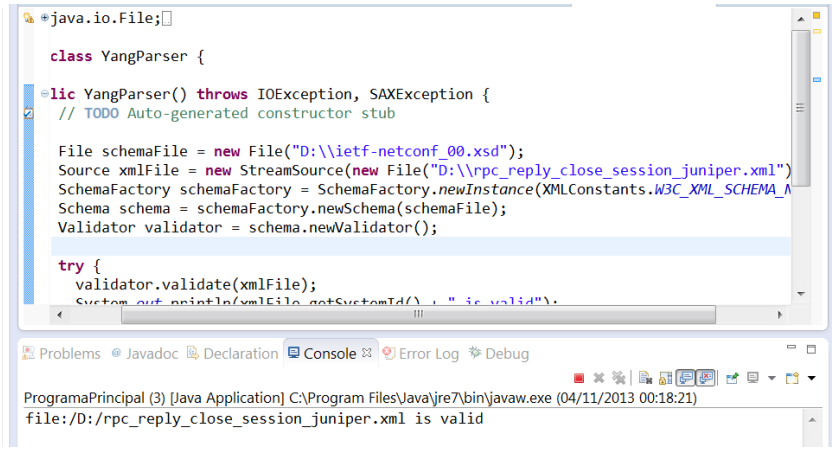


**Figura 49. Validación con eclipse de la rpc-reply enviada por el router Juniper, como respuesta de la rpc-request close-session enviada por el cliente.**

### 5.1.5. Envío de operación get-config para la recuperación de datos de configuración del equipo

La plataforma de gestión hace uso de la operación get-config (Ver Anexo 8), para realizar la operación "Obtener toda la configuración del equipo". Cuando hay una sesión netconf establecida, el usuario puede realizar las operaciones de configuración descritas en la sección 3 del presente documento, que hace referencia a las especificaciones del prototipo desarrollado. Así, en la **¡Error! No se encuentra el origen de la referencia.** se muestra el resultado de ejecutar la operación "Obtener toda la configuracion" en el router Cisco y en la **¡Error! No se encuentra el origen de la referencia.** se muestra el resultado de ejecutar la operación "Obtener toda la configuracion" en el router Juniper .

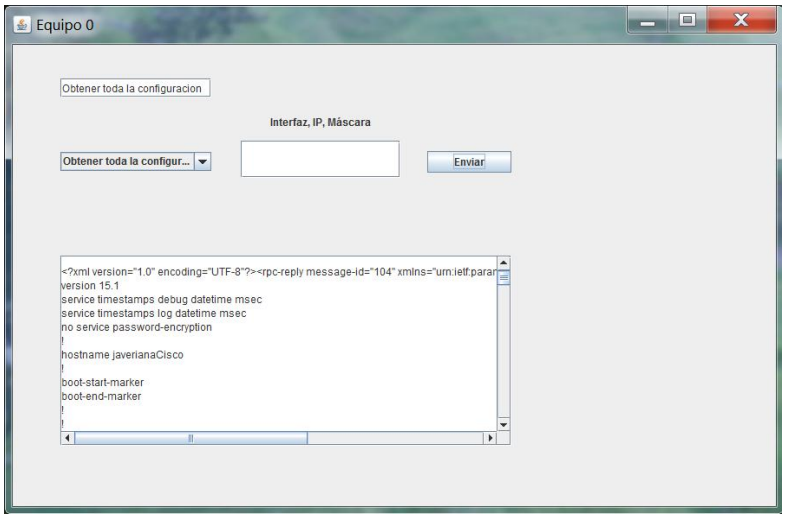


**Figura 50. Ventana de configuración del equipo con resultado de la ejecución de la operación "Obtener toda la configuración" en el equipo Cisco. Tomado de (17)**

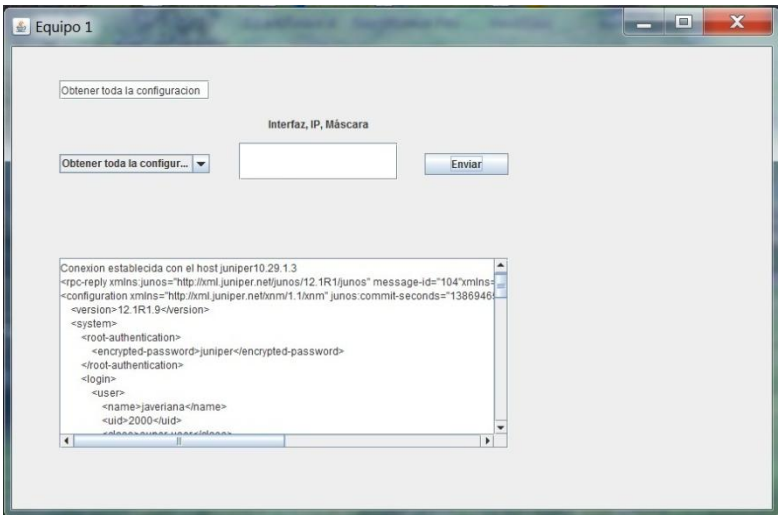


**Figura 51. Ventana de configuración del equipo con resultado de la ejecución de la operación "Obtener toda la configuración" en el equipo Juniper. Tomado de (17)**

### 5.1.6. Envío de operación edit-config para la configuración de una interfaz física del equipo

Con la configuración del equipo, el usuario puede observar el nombre de las interfaces del equipo, y si lo desea, puede configurar una de las interfaces físicas del equipo con IP y máscara de red. Para realizar esta configuración, se debe ingresar los parámetros en la caja de texto que se encuentra debajo de la inscripción “Interfaz,IP,Máscara” y pulsando el botón de enviar. Así, en la **¡Error! No se encuentra el origen de la referencia.** se muestra el resultado de ejecutar la operación “Configurar interface” en el router Cisco, y en la **¡Error! No se encuentra el origen de la referencia.** se muestra el resultado de ejecutar la operación “Obtener toda la configuracion” en el router juniper .

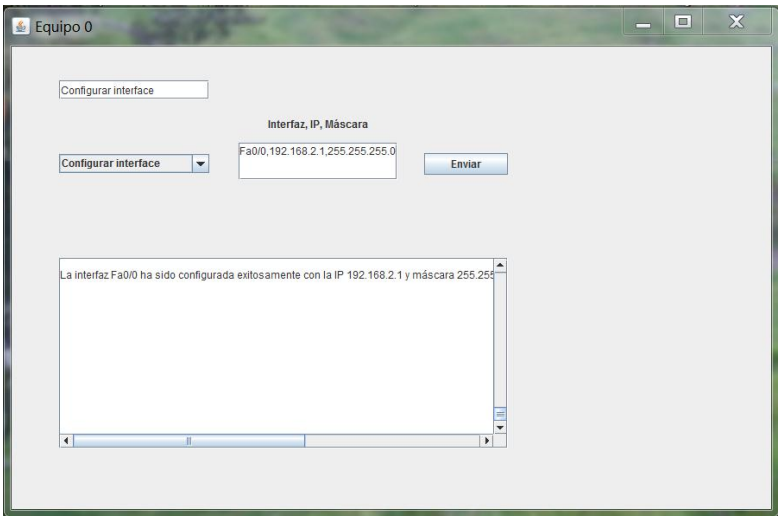


**Figura 52. Ventana de configuración del equipo con resultado de la ejecución de la operación "Configurar interface" en el equipo Cisco. Tomado de (17)**

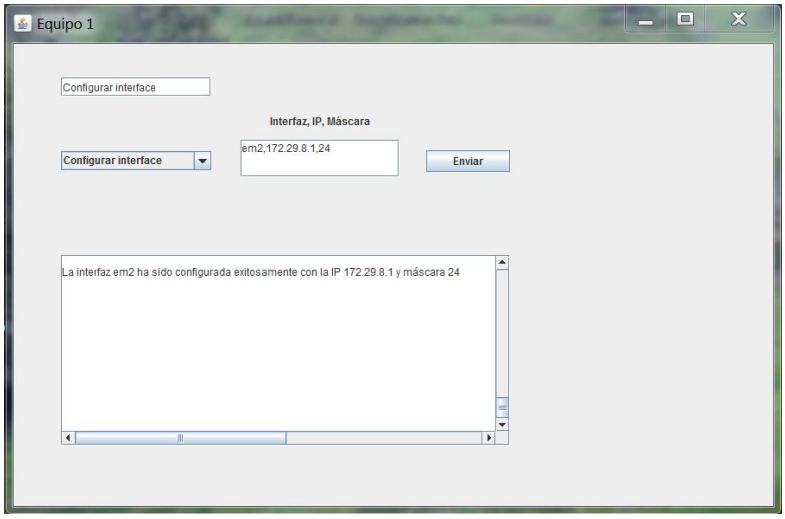


**Figura 53. Ventana de configuración del equipo con resultado de la ejecución de la operación "Configurar interface" en el equipo Juniper. Tomado de (17)**

Si se desea corroborar la información descrita en la consola de la ventana de configuración del equipo (cuadro de texto más grande de la ventana), se puede ejecutar nuevamente la operación "Obtener toda la configuracion" y se verificará si efectivamente la configuración fue almacenada por el dispositivo.

Para efectos de validación del funcionamiento de la plataforma de gestión, se accede directamente a las consolas de los routers y se verifica si ejecutaron la operación. En la Figura 54 se muestra la fracción de la interfaz configurada con el prototipo de gestión para el router Cisco, y en la Figura 55 se muestra

la fracción de la interfaz configurada con el prototipo de gestión para el router Juniper.

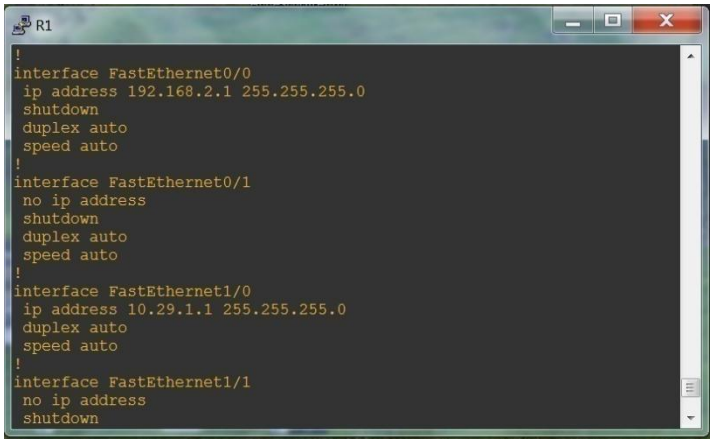


**Figura 54. Visualización de la configuración de la interfaz del Router Cisco configurada con el prototipo de gestion**

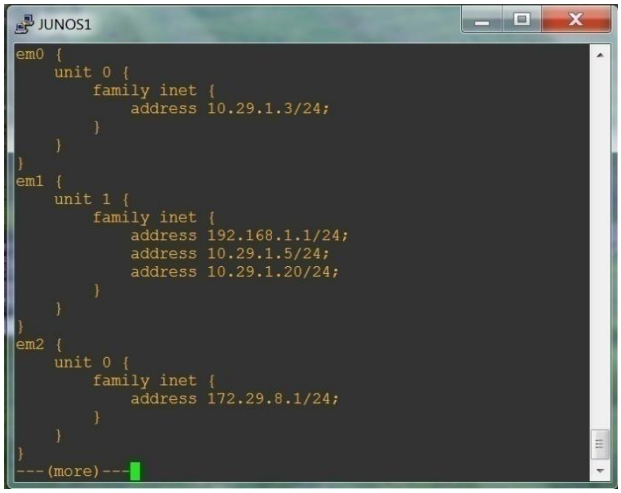


**Figura 55. Visualización de la configuración de la interfaz del Router Juniper configurada con el prototipo de gestion.**

### 5.1.7. Envío operación close-session para finalizar la sesión netconf entre el cliente y el servidor

Finalmente, cuando se desee finalizar la sesión netconf establecida entre el cliente (prototipo de gestión) y el servidor (router), debe escoger la opción "Cerrar sesion" de la lista desplegable que se encuentra en la ventana de configuración del equipo y pulsar el botón "Enviar". En la **¡Error! No se encuentra el origen de la referencia.** se muestra la ventana de

configuración del router Cisco en la que la consola indica que la sesión netconf con este dispositivo ha finalizado. Igualmente, en la **¡Error! No se encuentra el origen de la referencia.**, se muestra la ventana de configuración del router Juniper en la que la consola indica que la sesión netconf con este dispositivo ha finalizado.

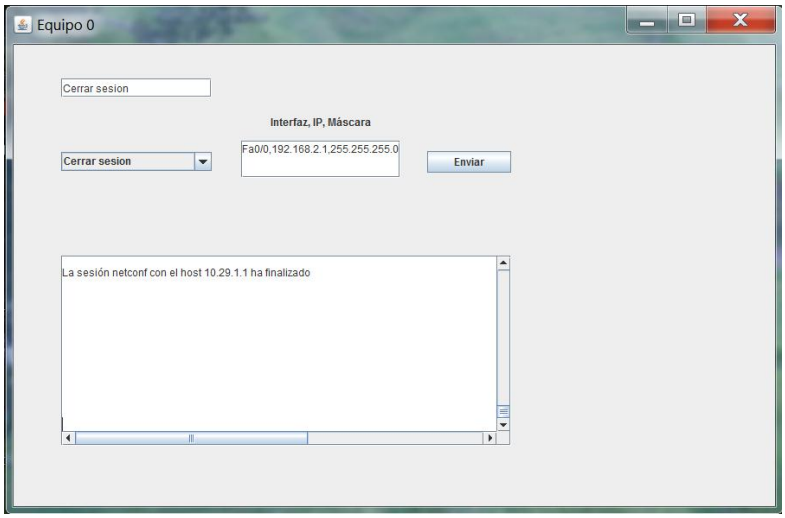


**Figura 56. Resultado de la ejecución de la operación Cerrar sesión en el router Cisco. Tomado de (17)**

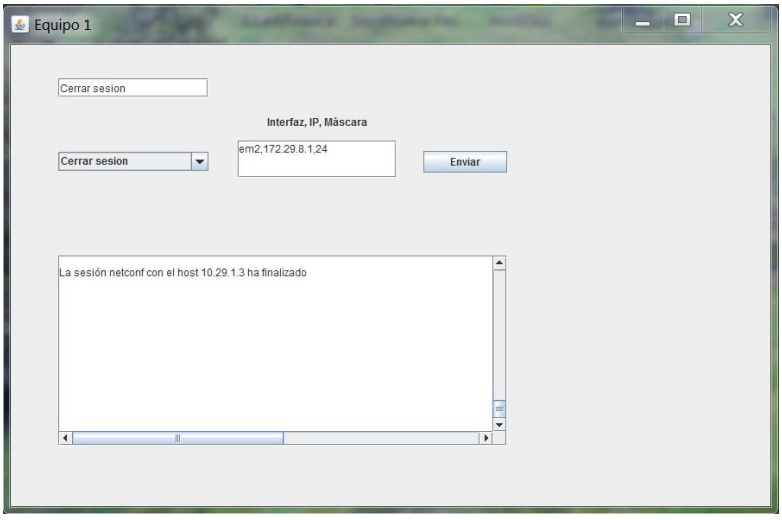


**Figura 57. Resultado de la ejecución de la operación Cerrar sesión en el router Juniper. Tomado de (17)**

# 6. CONCLUSIONES

Aunque ya han sido realizadas las especificaciones para la integración de la gestión de red de diferentes equipos de red de distintos proveedores, se observa que la completa integración de estas especificaciones en los equipos de red de datos de los proveedores Cisco y Juniper, aún está en desarrollo por parte de estos proveedores, ya que, además del soporte del protocolo de gestión de red NETCONF, es necesario que los equipos soporten un lenguaje de modelado de datos común, de tal forma que se asegure que los diferentes tipos de datos utilizados por el protocolo de gestión de red, cumplen con las mismas reglas, y en dado caso que no sea de esta manera, se pueda realizar una fácil corrección de los modelos. En el caso del router Cisco, utilizado en el desarrollo del presente proyecto, si hubiese soportado el lenguaje de modelado de datos YANG, la corrección del contenido del espacio de nombres del rpc-reply, se ubiera podido corregir con facilidad. La corrección de este fallo en los mensajes rpc-reply no se pudo ejecutar en el modelo de datos XSD que tiene el router, porque no se encontró en la bibliografía la implementación en el router de un nuevo esquema XSD.

Por otro lado, se tiene que los equipos de red de datos del proveedor Juniper que tengan el Junos 12.1R1.9, cumplen con el RFC6241 cuando realizan operaciones netconf. También, por el intercambio capacidades entre estos equipos y el cliente netconf desarrollado, se entiende que el equipo garantiza el cumplimiento con las especificaciones del protocolo Netconf, haciendo uso de su propio lenguaje de modelado de datos. También, en el caso de los equipos de red de datos del proveedor Cisco, se observa bajo este mismo análisis, que los equipos que tengan el IOS 15.1(4) M5, garantizan que las rpc request cumple con el RFC6241, haciendo uso de su propio lenguaje de modelado de datos XSD.

En cuanto al diseño de la arquitectura del prototipo de la plataforma de gestión de red, se observa que la integración de diferentes módulos de operación, facilita el desarrollo, entendimiento, y escalablilidad del prototipo de la plataforma de gestión de red, de tal modo que futuros desarrolladores, puedan fácilmente continuar con el trabajo y desarrollar una plataforma completa de gestión de red de equipos de datos de diferentes proveedores y que soporten NETCONF. Así, durante el desarrollo del prototipo fue un elemento clave la diferenciación de cada componente de software dentro de la plataforma, ya que un desarrollador que desee continuar con el desarrollo de la plataforma, entenderá rápidamente que el módulo “Netconf”, cumple con su rol de motor de implementación del protocolo

NETCONF, ya que es la entidad que se encarga de enviar las operaciones NETCONF al equipo de red que se esté gestionando y es quien verifica que la respuesta del equipo llegue en su totalidad al prototipo, y dependiendo de la misma, notificará el resultado de la operación.

Inicialmente se planteó la posibilidad de analizar los elementos rpc-reply provenientes de los equipos gestionados, pero esta idea se desechó por dos razones importantes: 1. Los rpc-reply de los equipos cisco con IOS 15.1(4) M5 no cumplen con el RFC6241, y 2. porque el análisis de la estructura del rpc-reply, no ofrece ningún valor al proceso de configuración del equipo, ya que lo realmente importante es verificar si este rpc-reply llega en su totalidad, y verificar los datos que contienen el resultado de la operación que ejecutó el dispositivo. Así, para efectos de comprobación y validación de los rpc enviados al servidor, y de los rpc-reply enviados por el servidor al cliente, se utilizó el módulo "YangParser" del prototipo de la plataforma de gestión. Este módulo, es quien le ayuda al desarrollador del prototipo de la plataforma de gestión, a validar previamente la sintaxis de las RPCs que envía el módulo "Netconf" al equipo gestionado, y saber si las mismas cumplen con el RFC6241 del protocolo NETCONF. En dado caso que la validación de estas rpcs indique que no cumplen con el modelado de datos de NETCONF, el desarrollador podrá consultar nuevamente la bibliografía y ajustarlas hasta que cumplan con las especificaciones del protocolo, logrando de esta manera, enseñar a la plataforma de gestión, la forma correcta de realizar las operaciones Netconf. También, en el caso de las rpc-reply, le indicará al desarrollador, los equipos que tienen una correcta implementación del protocolo NETCONF, y en dado caso de que no lo cumplan a cabalidad, puedan modificar el esquema de datos de estos equipos cuando los proveedores incorporen mecanismos que permitan realizar modificaciones a estos esquemas (esto teniendo en cuenta que el modelo de datos implementado en el equipo presente alguna desviación del modelo de datos descrito en el RFC6241). Si por el contrario, la falla se encuentra en que el dispositivo envía datos que no se rigen por el modelo de datos, se tendría que informar al proveedor para que realice la respectiva corrección en el firmware.

Debido a que no existe hasta el momento una impelementación del lenguaje de modelado de datos Yang en java, que permita realizar el análisis de la estructura y contenido de archivos en formato XML, con base a archivos en un formato del lenguaje de modelado de

datos Yang, el módulo Yang Parser del prototipo de la plataforma de gestión desarrollado, realiza el análisis de la estructura y contenido de las RPCs del protocolo netconf, haciendo uso de un esquema de datos XSD, el cual se obtiene de la conversión del esquema de datos del protocolo NETCONF de la capa de operaciones que viene en formato yang, a un esquema de datos en formato XML. Esta conversión se realiza haciendo uso de una de las herramientas yang gratuitas desarrolladas por la industria (Pyang).

El cumplimiento de los objetivos propuestos en el anteproyecto, se cumplieron a cabalidad, ya que el prototipo de la plataforma de gestión de red de datos que se desarrolló, implementó el protocolo de gestión de red NETCONF y el modelado de datos YANG, permitiendo la gestión de equipos de datos de los proveedores Cisco y Juniper que soportan el protocolo de gestión de red NETCONF. Así, dentro de las operaciones que envía el prototipo, y las cuales son ejecutadas exitosamente por los equipos, se encuentran las de iniciar sesión netconf, obtener toda la configuración de los equipos, configurar una interfaz física de los equipos y finalmente terminar la sesión netconf con los equipos.

El modelo de datos Yang para las operaciones del protocolo netconf (que importa el modelo de datos de la capa de mensajes del protocolo netconf), no valida los rpc-reply que contengan datos diferentes a los elementos ok ó rpc-error, debido a que en el modelo de datos de la capa de mensajes del protocolo netconf, no hay jerarquías que permitan la inclusión de datos, por lo que elementos diferentes a ok ó rpc-error, son desconocidos para el esquema de datos de la capa de mensajes del protocolo NETCONF.

Durante la contrucción de las rpc request para realizar la configuración de una interfaz física en los dispositivos Cisco y Juniper, se identificó que pueden ser diferentes los contenidos en la jerarquía de configuración (<config>) de la operación edit-config en distintos equipos de red de datos, ya que en el caso particular de los equipos de estos dos proveedores, los esquemas de las jerarquías de configuración de estos equipos son diferentes.

Fue preocupante encontrar durante la búsqueda bibliográfica del protocolo de red NETCONF, que la operación kill-session del protocolo NETCONF requiere un mayor control, dado que pese a que un usuario cumpla con el ciclo de configuración seguro para realizar cambios en un dispositivo (se descarten los cambios realizados en el almacenamiento de datos *candidate*, y se realice el bloqueo para que ningún usuario realice

cambios en el almacenamiento de datos), otro usuario puede obtener la sesión-id de quién está bloqueando la opción de realizar configuraciones en un almacenamiento de datos, enviando una petición de bloqueo a un datastore determinado, y como el primer usuario está manteniendo un bloqueo sobre este datastore, el dispositivo retornará un rpc-error con el id de la sesión que actualmente está sosteniendo el bloqueo. Mediante este id, el segundo usuario puede forzar la terminación de la sesión de quien estaba bloqueando la ejecución de cambios de configuración sobre un almacenamiento de datos , impidiendo que un usuario que esté realizando cambios en el equipo pueda finalizar las configuraciones que estaba realizando.

Finalmente, cabe resaltar que inicialmente la implementación de las especificaciones de NETCONF y YANG en el prototipo de la plataforma de gestión de red, fue complicada, ya que las fuentes bibliográficas que refieran a la implementación y/o uso del modelado de datos YANG en equipos Cisco y Juniper es nula.

## 7. Bibliografía


1. *NETCONF agent for link state monitoring.* **Diogo Loureiro, Pedro Gonçalves, António Nogueira.** 2012, 2nd IEEE International Workshop on Smart Communication Protocols and Algorithms, pp. 6565 - 6569.

2. *On Wide Area Network Optimization.* **Yan Zhang, Nirwan Ansari, Fellow, Mingquan Wu, Heather Yu.** 2012, IEEE COMMUNICATIONS SURVEYS & TUTORIALS, pp. 1090 - 1113.

3. *Ontology Mapping for the Interoperability.* **Alfred Ka Yiu Wong, Pradeep Ray, N. Parameswaran, John Strassner.** 2005, IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, pp. 2058 - 2068.

4. **Schoenwaelder, J.** Management of Networks with Constrained Devices: Problem Statement, Use Cases and Requirements. [Online] 02 14, 2013. [Cited: 03 25, 2013.] http://tools.ietf.org/html/draft-ersue-constrained-mgmt-03.

5. *Network configuration management using NETCONF and YANG.* **Jürgen Schönwälder, Martin Björklund, Phil Shafer.** 2010, Communications Magazine, IEEE , pp. 166 - 173.

6. **SNMP Research International, Inc.** Secure Your Network. [Online] 2013. [Cited: Abril 1, 2013.] http://www.snmp.com/protocol/snmp_rfcs.shtml.

7. **Network Working Group, R. Frye.** Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework. [Online] 2003. [Cited: Abril 1, 2013.] http://tools.ietf.org/html/rfc3584.

8. *Integrating SNMP agents and CLI with NETCONF-based network management systems.* **Wu, Baozhen and Chang, Yanan.** 2010, IEEE CONFERENCE PUBLICATIONS, pp. 81 - 84.

9. **IETF Network Working Group R. Enns, Ed.** NETCONF Configuration Protocol. *RFC4741*. [Online] December 2006. http://tools.ietf.org/html/rfc4741.

10. **Internet Engineering Task Force (IETF) R. Enns, Ed.** Network Configuration Protocol (NETCONF). *RFC6241*. [Online] 2011 June. http://tools.ietf.org/html/rfc6241.
11. **Internet Engineering Task Force (IETF).** YANG - A Data Modeling Language for. [Online] Octubre 2010. http://tools.ietf.org/html/rfc6020.
12. *Research and Implement on Compatibility and Scalability of Agent of Platform for Network Management Based on NETCONF Protocol.* **Shisong, Xiao, Ran, Jia and Junying, Guo.** 2010, IEEE CONFERENCE PUBLICATIONS, pp. 1035 - 1038.
13. **Internet Engineering Task Force (IETF).** Using the NETCONF Protocol over Secure Shell . [Online] Junio 2011. http://www.rfc-editor.org/rfc/rfc6242.txt.
14. **W3C.** Extensible Markup Language (XML) 1.0 (Second Edition). [Online] Octubre 2000. http://www.w3.org/TR/2000/REC-xml-20001006.
15. **Siemens.** RUGGEDCOM ROX II NETCONF. [Online] Julio 2013. http://www.ruggedcom.com/pdfs/roxII-netconf/roxii_netconf_reference_guide.pdf.
16. Brocade MLX Series and Brocade NetIron Family YANG Guide. [Online] Junio 2011. http://www.brocade.com/downloads/documents/product_manuals/B_NetIron/Brocade_05200_YangGuide.pdf.
17. **Rojas, Adriana Fernanda Amarillo.** Ingeniera Electrónica. Bogotá : s.n., 2013.
18. *Customizing Rational Unified Process in a Systems.* **Lice, Erida and Biba, Marenglen.** 2012, IEEE CONFERENCE PUBLICATIONS, pp. 76 - 83.
19. **The Eclipse Foundation.** Eclipse. [Online] 2013. http://www.eclipse.org/windowbuilder/download.php.
20. An extensible YANG validator and converter in python. [Online] 2013. https://code.google.com/p/pyang/.
21. **CISCO.** NETCONF over SSHv2. [Online] Juinio 2006. http://www.cisco.com/en/US/docs/ios/12_2sr/12_2sra/feature/guide/srnetcon.pdf.
22. —. Cisco Feature Navigator. [Online] Enero 2012. http://tools.cisco.com/ITDIT/CFN/jsp/by-feature-technology.jsp.
23. **JCraft.** Code the Craft, Craft the Code. [Online] 2012. http://www.jcraft.com/jsch/.

## 8. ANEXOS

**Anexo 1. Terminología NETCONF**

**Anexo 2. Módulo YANG para la capa de operaciones de NETCONF**

**Anexo 3. Typedefs importados por el módulo YANG**

**Anexo 4. Terminología YANG**

**Anexo 5. Mensaje Hello**

**Anexo 6. Mensaje Hello enviado por el servidor Cisco**

**Anexo 7. Mensaje Hello enviado por el servidor Juniper**

**Anexo 8. Operación get-config**

**Anexo 9. Operación edit-config para router Cisco**

**Anexo 10. Operación edit-config para router Juniper**

**Anexo 11. Esquema netconf del router cisco**

**Anexo 12. Esquema jerárquico de las interfaces del router Juniper**

**Anexo 13. Mensaje rpc-reply**

**Anexo 14. Operación close-session**

**Anexo 15. Clase principal del proyecto**

**Anexo 16. Código fuente en java de la clase User Interface**

**Anexo 17. Código fuente en java de la clase Canal SSH**

**Anexo 18. Código fuente en java de la clase Comandos**

**Anexo 19. Código fuente en java de la clase Netconf**

**Anexo 20. Código fuente en java de la clase YANG Parser**

**Anexo 21. Esquema XSD para las operaciones del protocolo de gestión de red NETCONF**

**Anexo 22. Esquema XSD para la capa de mensajes del protocolo NETCONF**