(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 824 023**

(51) Int. CI.:

H04L 29/06 (2006.01)
H04L 12/24 (2006.01)
H04L 12/46 (2006.01)
G06F 9/54 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **26.02.2015 PCT/CN2015/073307**

(87) Fecha y número de publicación internacional: **03.09.2015 WO15127894**

(96) Fecha de presentación y número de la solicitud europea: **26.02.2015 E 15755432 (0)**

(97) Fecha y número de publicación de la concesión europea: **19.08.2020 EP 3103244**

(54) Título: **Planteamiento declarativo para la creación y explotación de redes virtuales**

(30) Prioridad:

28.02.2014 US 201414193034

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
11.05.2021

(73) Titular/es:

HUAWEI TECHNOLOGIES CO., LTD. (100.0%)
Huawei Administration Building, Bantian, Longgang District
Shenzhen, Guangdong 518129, CN

(72) Inventor/es:

ZHANG, SHUJIN

(74) Agente/Representante:

ELZABURU, S.L.P

DESCRIPCIÓN

Planteamiento declarativo para la creación y explotación de redes virtuales

Campo técnico

La presente invención se refiere, en general, a una red virtual (VN), y en realizaciones particulares, a técnicas para la creación y explotación de redes virtuales.

Antecedentes

Una red virtual (VN) es una red de ordenadores que comprende nodos y/o enlaces de red virtuales. El concepto y la adopción de redes virtuales (VNs) se han popularizado ampliamente debido a diversas razones, tales como la necesidad de una introducción de servicios rápidos y ahorros de costes operativos en la informática en la nube. Una VN se puede formar con o sin una topología de red física correspondiente. Para crear y explotar una VN, las aplicaciones de red pueden llamar a una interfaz de programación de aplicaciones (API) proporcionada por un controlador de red que controla la VN. Por ejemplo, se pueden transmitir solicitudes de una aplicación al controlador por medio de la API de una en una, y una VN se puede construir o actualizar en función de la solicitud para adaptarse a las necesidades de la aplicación.

En la actualidad, los desarrolladores de aplicaciones pueden usar un planteamiento procedimental para la explotación de una VN. En este planteamiento, una aplicación puede usar una API de controlador para construir un procedimiento de tipo paso a paso que represente una lógica de aplicación destinada a la explotación de la VN. La construcción de los pasos procedimentales puede requerir que los desarrolladores de aplicaciones tengan un conocimiento profundo de la VN (por ejemplo, cómo está construida la VN). A medida que la escala y la complejidad de una VN crecen, puede resultar cada vez más difícil para los desarrolladores de aplicaciones gestionar la VN.

HEWLETT-PACKARD ET AL: “Annex A. Relationship to SDN; NFVSWA (14) 000078r2_Annex_A_Relationship_to_SDN-A-1-A-4” BORRADOR DEL ETSI; 650, ROUTE DES LUCIOLES; F-06921 SOPHIA-ANTIPOLIS; FRANCIA, vol. ISG-NFV, 24 de febrero de 2014, páginas 1 a 7, XP014229721, da a conocer una visión general de una SDN, por ejemplo, el principio de la SDN es que el controlador de la SDN carga tablas de reenvío de paquetes dinámicamente en los elementos de red de la SDN por medio de una interfaz abierta. El controlador de la SDN gestiona su topología subyacente y la consistencia de estas tablas de reenvío de paquetes, y ofrece APIs hacia el norte (en inglés, *northbound*).

JESUS WANDERSON PAIM DE ET AL: “ProViNet – An Open Platform for Programmable Virtual Network Management”, 2013 37ª Conferencia de Aplicaciones y Software Informático Anual (en inglés, Computer Software and Applications Conference, COMPSAC) del IEEE, 22-07-2013, páginas 329 a 338, XP032517864, proporciona una plataforma ProViNet que fusiona los conceptos de SDN y NV (en inglés, network virtualization, virtualización de red) para proporcionar un despliegue en la infraestructura de redes existente.

El documento “Meridian: An SDN Platform for Cloud Network Services” de M. Banikazemi et al., IEEE Communications Magazine, febrero de 2013, XP11493792, describe la arquitectura de Meridian, una plataforma de controladores de SDN que admite un modelo de nivel de servicio para redes de aplicaciones en nubes.

Compendio

La presente invención es tal como se define en las reivindicaciones independientes adjuntas. Realizaciones de ejemplo dadas a conocer pueden ayudar a mejorar la gestión de redes virtuales (VN) usando códigos de programa escritos en un lenguaje de programación declarativo en lugar de un lenguaje de programación procedimental con el fin de llevar a cabo diversas tareas relacionadas con las VN. En una realización de ejemplo, un controlador de red lleva a cabo un método según se reivindica en la Reivindicación independiente 1.

En una realización de ejemplo, se proporciona un producto de programa de ordenador almacenado en un controlador de red para la gestión de redes virtuales, VN, según se reivindica en la Reivindicación independiente 5.

En otra realización, se proporciona un controlador de red para la gestión de redes virtuales, VN, según se reivindica en la Reivindicación independiente 7.

Estas y otras características se entenderán más claramente a partir de la siguiente descripción detallada considerada en combinación con los dibujos y las reivindicaciones adjuntos.

Breve descripción de los dibujos

Para entender de manera más completa a esta exposición, se hace referencia a continuación a la breve descripción sucesiva, considerada en relación con los dibujos y la descripción detallada adjuntos, en donde números de referencia iguales representan partes iguales.

La FIG. 1 es un diagrama esquemático que muestra una realización de ejemplo de un modelo de red.

La FIG. 2 es un diagrama esquemático que muestra una realización de ejemplo de la arquitectura de una solución de red virtual (VN).

La FIG. 3A es un diagrama esquemático que muestra una realización de ejemplo de una arquitectura de comunicación de VN.

La FIG. 3B ilustra un fragmento ilustrativo de seudocódigos escritos en un planteamiento procedimental.

La FIG. 4 es un diagrama esquemático que muestra una realización de ejemplo de una arquitectura de comunicación entre una aplicación y un controlador de red.

Las FIGS. 5A y 5B ilustran un esquema ilustrativo de lenguaje de marcado extensible (XML) que describe por lo menos algunos aspectos de una VN.

La FIG. 6A ilustra un fragmento ilustrativo de códigos de programa que describe una VN ilustrativa.

La FIG. 6B ilustra la VN creada con el uso de los códigos de programa mostrados en la FIG. 6A.

La FIG. 7 es un diagrama de flujo de una realización de ejemplo de un método de gestión de VN.

La FIG. 8 es un diagrama esquemático de una realización de ejemplo de un dispositivo de red.

Descripción detallada

Debe entenderse desde el principio que, aunque a continuación se proporciona una implementación ilustrativa de una o más realizaciones de ejemplo, los sistemas y/o métodos dados a conocer se pueden implementar usando el número de técnicas que se desee, ya sean conocidas o existentes actualmente. La exposición no debe limitarse en modo alguno a las implementaciones, dibujos y técnicas ilustrativos que se muestran a continuación, incluidos los diseños e implementaciones ilustrativos que se ilustran y describen en la presente, sino que puede modificarse dentro del alcance de las reivindicaciones adjuntas.

Para gestionar una red virtual (VN), un controlador de red puede proporcionar una interfaz de programación de aplicaciones (API) para interaccionar con aplicaciones que son explotadas por desarrolladores de aplicaciones. Se han producido esfuerzos por normalizar los formatos de las API. Por ejemplo, puede considerarse que Elastic Compute Cloud (EC2) propuesta por AMAZON es una norma de facto para las API, y la API Quantum de OpenStack puede considerarse una variante de la API de EC2. Hay además en marcha propuestas nuevas que intentan ampliar la API de manera que incorpore nuevas características o escenarios de aplicación.

Algunos de los planteamientos de API existentes comparten unos cuantos rasgos comunes. En primer lugar, las APIs pueden ser descriptivas. Una API puede representar una acción o tarea que las aplicaciones ordenan llevar a cabo al controlador de red, y los desarrolladores de aplicaciones pueden proporcionar parámetros de entrada para la tarea y esperar un resultado después de que se haya completado la llamada de la API. En segundo lugar, las APIs pueden ser procedimentales. Aplicaciones pueden usar una API para construir un procedimiento que represente la lógica de la aplicación, por lo que puede que sea necesario que los desarrolladores de aplicaciones tengan un conocimiento profundo de cómo se construye una red virtual por medio de los pasos procedimentales. A medida que crecen la escala y la complejidad de una VN, los desarrolladores de aplicaciones pueden escribir códigos de programa largos y complicados para crear y explotar la VN. Además, el mantenimiento de la VN y la prevención de potenciales fallos de programación o errores pueden resultar costosos.

La presente exposición revela realizaciones de ejemplo para hacer frente a las debilidades de las APIs actuales usando un planteamiento declarativo en lugar de un planteamiento procedimental en la creación y la explotación de una VN. Específicamente, se pueden generar solicitudes u órdenes que comprenden códigos de programa usando un lenguaje de programación declarativo en lugar de un lenguaje de programación procedimental. El lenguaje declarativo, al que se hace referencia, en ocasiones, en la presente, como lenguaje de definición de red virtual (VNDL), se puede escribir en diversos formatos o esquemas, tales como un formato de lenguaje de marcado extensible (XML). Un desarrollador de aplicaciones puede usar el VNDL para describir una VN en términos de nombres definidos, tales como nodo, puerto, enlace, ancho de banda, política y gestión de eventos. A diferencia de los lenguajes de programación procedimentales, el VNDL puede especificar lógicas u objetivos de una tarea sin tener que especificar explícitamente cada paso detallado con el fin de llevar a cabo una tarea.

Los códigos de programa escritos en VNDL pueden ser generados por una aplicación y, a continuación, se pueden enviar a un controlador de red por medio de una API admitida por el controlador de red. El controlador de red puede analizar sintácticamente el código VNDL obteniendo objetos internos y llevar a cabo una ejecución en consecuencia usando una API interna del controlador. El controlador de red puede llevar a cabo varias tareas para explotar una VN, que incluyen la creación de la VN (por ejemplo, definición de nodos, puertos y enlaces), la actualización de la VN, la imposición de políticas de red de la VN, y la gestión de eventos que se producen en la VN.

La FIG. 1 es un diagrama esquemático que muestra una realización de ejemplo de un modelo 100 de red, que comprende una o más VNs configuradas sobre una capa de red física. Tal como se muestra en la FIG. 1, la capa de

red física comprende una red 110 y una red 120, que pueden considerarse redes diferentes o dominios diferentes de la misma red. Las redes 110 y 120 comprenden nodos de red físicos, los cuales se pueden interconectar entre sí por medio de enlaces físicos. Los nodos de red físicos en las redes 110 y 120 se pueden interconectar con otros nodos de red tanto dentro como fuera de la red doméstica.

Para ampliar funcionalidades de la estructura en red, pueden formarse una o más VNs usando algunos de los nodos de red en la capa de red física. Por ejemplo, se puede configurar una primera VN 130 usando los nodos 111, 113, 115 y 121 a 125 de red, y se puede configurar una segunda VN 140 usando los nodos 112, 114, 116, 121 a 122 y 124 a 125 de red. En la práctica, parte de las funcionalidades de los nodos físicos se puede usar para formar nodos de red virtuales, que, a su vez, pueden constituir las VNs. Por lo tanto, un nodo físico se puede usar para construir uno o más nodos virtuales. Tal como se muestra en la FIG. 1, dos nodos virtuales 141 y 142, que pertenecen ambos a la VN 140, se forman compartiendo el mismo nodo físico 116. Enlaces virtuales pueden interconectar nodos virtuales entre sí. No obstante, también es posible disponer de nodos virtuales sin ninguna conexión. Debe entenderse que los nodos y enlaces de red físicos/virtuales descritos en la presente se pueden implementar usando cualesquiera técnicas adecuadas. En función de la aplicación, a una VN se le puede hacer referencia, en ocasiones, con nombres equivalentes o similares. Por ejemplo, una VN se puede implementar en forma de una red privada virtual (VPN), una red definida por *software* (SDN), una red centrada en servicios (SCN), una red centrada en información (ICN), una red centrada en el contenido (CCN), una red de operador virtual, una red empresarial virtual, una red específica de aplicación, y así sucesivamente. De este modo, el término VN que se da a conocer en la presente puede abarcar estas equivalentes y variantes.

Para gestionar una red que comprenda múltiples nodos de red, un controlador de red se puede establecer como una unidad central que supervise la red. Esta arquitectura puede usarse en redes tanto físicas como virtuales. La FIG. 2 es un diagrama esquemático que muestra una realización de ejemplo de una arquitectura 200 de solución VN, que abarca capas de red tanto físicas como virtuales. En redes o dominios de red grandes, se pueden desplegar múltiples controladores de red que supervisen, cada uno de ellos, una parte de la(s) red(es) completa(s). Tal como se muestra en la FIG. 2, un primer controlador de red física (PNC) 220 se puede configurar para gestionar una primera pluralidad de nodos físicos 211, 212 y 213 de red, mientras que un segundo PNC 230 se puede configurar para gestionar una segunda pluralidad de nodos físicos 214, 215 y 216 de red. Los nodos 211 a 216 de red se pueden implementar de manera que tengan cualquier función, tal como plataformas de encaminamiento versátiles (VRPs). Cada PNC comprende diversos módulos o unidades. Por ejemplo, el PNC 220 puede comprender un servidor web 221 y una pluralidad de módulos de servicio que realicen servicios de PNC y otros servicios.

En la práctica, una red relativamente grande se puede dividir en múltiples fragmentos de redes o dominios más pequeños, cada uno de los cuales puede ser gestionado por un PNC. Múltiples PNCs, tales como los PNCs 220 y 230, pueden ser gestionados además por un controlador de red de nivel superior, por ejemplo, un controlador de VN (VNC) 240 mostrado en la FIG. 2, para crear una jerarquía de controladores de dos capas. Obsérvese que, si se desea, se pueden construir más capas o niveles. Una aplicación (app) 250 puede acceder a recursos de la red y controlarlos por medio del PNC 220, del PNC 230 y/o del VNC 240, que se pueden implementar usando un servidor de plataforma abierta (OPS). El VNC 240 se puede comunicar con controladores de nivel inferior, tales como el PNC 230 por medio de una API 246, por ejemplo, transmitiendo solicitudes u órdenes 247 al PNC 230. Obsérvese que los controladores 220, 230 y 240 pueden estar situados en la misma ubicación física (escenario centralizado) o en ubicaciones diferentes (escenario distribuido). La aplicación 250 se puede implementar como cualquier forma adecuada de programa de *software* que se ejecute en el dispositivo o nodo de red que sea explotado por un desarrollador de aplicaciones.

El VNC 240 puede ser una aplicación que se coordine con múltiples PNCs. El VNC 240 puede ofrecer muchos tipos diferentes de servicios que beneficien a los desarrolladores de aplicaciones, tales como la provisión de una interfaz de nivel superior para que las aplicaciones creen, configuren, actualicen y exploten VNs. En una realización de ejemplo, el VNC 240 comprende diversos módulos para llevar a cabo diversas tareas, y los módulos pueden incluir un servidor web 241, un módulo 242 de servicio de VNC, un módulo 243 de servicio de capa de abstracción de topologías (TAL), uno o más módulos 244 de servicio de cálculo de rutas (RC), un módulo 245 de expresión de VNC y otros módulos de servicio. El servidor web 241 puede interaccionar con la aplicación 250 por medio de una API 252. Específicamente, el servidor web 241 se puede integrar en el VNC 240 para recibir solicitudes 254 (por ejemplo, solicitudes de API RESTful), y a continuación despachar las solicitudes 254 a otros módulos en el VNC 240. Las solicitudes 254 pueden comprender códigos de programa escritos en VNDL para explotar la red. El módulo 242 de servicio de VNC puede facilitar la gestión de los PNCs 220 y 230. Adicionalmente, el módulo 243 de servicio de TAL puede recopilar la topología de cada PNC y, a continuación, combinar múltiples topologías en una única topología de la red completa. Desarrolladores de aplicaciones pueden usar la topología como herramienta de ayuda para construir VNs de gran tamaño, por ejemplo, aquellas que se extienden sobre múltiples redes físicas.

Los módulos 244 de servicio de RC pueden proporcionar opciones de implementación para la API 252, que pueden tener parámetros para la elección de la implementación a usar para la gestión de la API. El módulo 245 de expresión de VNC puede permitir que el desarrollador de aplicaciones escriba códigos de programa usando un lenguaje de programación declarativo, y que use los códigos de programa para facilitar la creación y la explotación de la VN. Desde una perspectiva práctica, los desarrolladores de aplicaciones pueden interpretar un VNC o un PNC como un controlador de SDN. Por ejemplo, el VNC 240 puede considerarse un controlador de SDN para acceder a recursos de

red físicos. El VNC 240 puede proporcionar cualquier tipo adecuado de API para comunicarse con desarrolladores de aplicaciones y otros dispositivos de red. Por ejemplo, la API 252 y/o la API 246 pueden ser una API REST o RESTful, tal como la EC2 propuesta por AMAZON y OpenStack, que es un tipo de API sin estados. Obsérvese que las APIs 246 y 252 pueden ser del mismo tipo o de tipos diferentes. Además, los formatos de las API pueden ser diferentes incluso si estas son del mismo tipo ya que cada proveedor puede elegir la adición de funciones nuevas.

En la práctica, controladores diferentes pueden tener APIs diferentes. Aunque la presente exposición se menciona en su mayor parte en el contexto de una API RESTful para controladores tales como el VNC 240 y los PNCs 220 y 230, puede usarse cualquier otro tipo adecuado de API siempre que la aplicación 250 pueda enviar códigos de programa escritos en el VNDL a los controladores, tales como el VNC 240.

La FIG. 3A es un diagrama esquemático que muestra una realización de ejemplo de una arquitectura 300 de comunicación de VN. Alguien con conocimientos habituales en la técnica dilucidará las similitudes entre la arquitectura 200 y 300, por lo que, en interés de su concisión, las descripciones adicionales se centrarán en aspectos que son diferentes o que todavía no han sido tratados (se aplica el mismo principio a otras figuras). Una o más aplicaciones 310 pueden comunicarse con un controlador 320 por medio de una API 322. El controlador 320 puede ser un VNC (por ejemplo, el VNC 240) que gestiona una VN 330 superpuesta sobre una red física 332. En la práctica, las aplicaciones 310 se pueden implementar usando cualquier tipo adecuado de *software* y se pueden ejecutar sobre nodos físicos y/o virtuales.

En una realización de ejemplo, los desarrolladores que usan las aplicaciones 310 pueden escribir códigos 312 de programa en un lenguaje de programación declarativo. Los códigos 312 de programa se pueden enviar por medio de la API 322 al controlador 320, que puede comprender un módulo o unidad 324 de expresión de VNC. Después de recibir un código de programa escrito en el lenguaje de programación declarativo, el módulo 324 de expresión de VNC puede llevar a cabo varias tareas sobre los códigos de programa. El lenguaje de programación declarativo, al que se hace referencia en la presente, en ocasiones, como VNDL, puede adoptar la forma de cualquier formato adecuado, tal como un formato XML, de modo que los términos o nombres lícitos se pueden definir con un esquema XML. El VNDL permite que los desarrolladores de aplicaciones describan una VN.

El planteamiento declarativo que se da a conocer en la presente para la explotación de una VN puede resultar ventajoso con respecto a un planteamiento procedimental convencional. La FIG. 3B ilustra un fragmento ilustrativo de seudocódigos 350 escrito en un planteamiento procedimental. Es posible que los códigos de programa escritos en un lenguaje de programación procedimental necesiten especificar explícitamente cada paso detallado con el fin de llevar a cabo una tarea. Tal como se muestra en la FIG. 3B, para crear una red denominada “roja”, los seudocódigos 350 especifican cada paso de adición de nodos, de puertos en los nodos, y de enlaces entre los puertos. Además, para gestionar un evento (por ejemplo, si un enlace se cae) que se produce en la red “roja”, es necesario que los seudocódigos 350 especifiquen pasos exactos sobre cómo gestionar el evento. De este modo, el planteamiento procedimental convencional puede requerir que un desarrollador de una aplicación entienda a fondo los detalles de una VN. Por contraposición, la explotación de una VN que usa un lenguaje de programación declarativo de los revelados en la presente puede evitar dichos problemas al no requerir especificación de cada paso detallado.

En las ciencias informáticas, la programación declarativa puede considerarse un paradigma de programación o un estilo de construcción de programas de ordenador, en los que los códigos de programa expresan la lógica de una computación sin tener que describir el flujo de control detallado. Este planteamiento declarativo puede minimizar o eliminar efectos colaterales al describir los que debería lograr el programa en términos de lógicas y objetivos, más que describiendo cómo se debe lograr en forma de una secuencia de primitivas del lenguaje de programación. Los detalles sobre cómo lograrlo se pueden dejar para que sean determinados por otro dispositivo (por ejemplo, el controlador VNC).

Debe entenderse que hay una multitud de lenguajes de programación que se pueden clasificar o categorizar como lenguajes de programación declarativos. La categoría de lenguajes de programación declarativos comprende subcategorías que incluyen lenguajes de marcado declarativos, lenguajes funcionales, lenguajes de programación lógica y lenguajes de programación lógica funcional. Los ejemplos específicos de lenguajes de programación declarativos incluyen, aunque sin carácter limitativo, Alpha, Atom, Sistema de tipo aplicado (ATS, por sus siglas en inglés), Curl, Lenguaje de Especificación de Aplicaciones Distribuidas, el lenguaje de programación centrado en datos conocido como ECL, Erlang, el Lenguaje de Marcado de Aplicación Extensible (XAML, por sus siglas en inglés), lógica de trama (F-logic, abreviado en inglés), FXML, el Lenguaje Declarativo de Propósito General (GenDL, por sus siglas en inglés), Glowe, el lenguaje de programación de agentes GOAL, el lenguaje Bueno para razonamiento el ecuacional (Gofer, por sus siglas en inglés), el lenguaje de programación Conocimientos de Maquina (KM, por sus siglas en inglés), Lithe, Lucid, Lustre, Miranda, Pan, Qi, XML, QUILL, SequenceL, el Lenguaje Estructurado de Consultas (SQL, por sus siglas en inglés), el Lenguaje de Integración Multimedia Sincronizado (SMIL, por sus siglas en inglés), y el Lenguaje Ontológico de Web (OWL, por sus siglas en inglés), y combinaciones de los mismos. Por otro lado, los ejemplos de lenguaje de programación procedimental, al que se hace referencia en ocasiones como lenguaje de programación imperativo, pueden incluir Visual Basic, C, C++, el Lenguaje Común Orientado a Negocios (COBOL, por sus siglas en inglés), Java, Perl, Python, y un lenguaje de guion de instrucciones del lado del servidor, de fuente abierta, conocido como PHP. Además, debe entenderse que el planteamiento declarativo dado a conocer en la presente puede ser compatible con planteamientos procedimentales existentes y se puede usar como una tecnología

complementaria en ciertas situaciones.

La FIG. 4 es un diagrama esquemático que muestra una realización de ejemplo de una arquitectura 400 de comunicaciones entre una aplicación 410 y un controlador 420 de red. El controlador 420 de red se puede implementar en forma del VNC 240 ó del controlador 320, y sus partes ilustradas en la FIG. 4 se pueden implementar con un módulo de expresión de VNC y otros módulos pertinentes. En una realización de ejemplo, la aplicación 410 puede generar órdenes o solicitudes escritas en VNDL para diversas tareas de la VN. Por ejemplo, un desarrollador de aplicaciones puede escribir solicitudes usando la aplicación 410 con el fin de crear y/o explotar una VN. La aplicación 410 puede llamar al controlador 420 enviando una orden o una solicitud 412 que comprende códigos de programa escritos en VNDL. La solicitud 412 se puede encapsular o formatear usando cualquier forma adecuada, y los códigos de programa de la misma pueden tener cualquier formato o longitud adecuado (por ejemplo, que contengan guiones de instrucciones, cadenas o códigos). La aplicación 410 puede transmitir la solicitud 412 por medio de una API 421 hacia el norte (en inglés, *northbound)*, que puede ser una API sin estados o RESTful.

Un servidor 422 de Protocolo de Transferencia Hipertexto (HTTP) integrado en el controlador 420 puede recibir la solicitud 412 de la aplicación 410. A continuación, la solicitud 412 que contiene códigos VNDL se puede trasladar a un módulo principal 424, el cual puede iniciar y/o coordinar el proceso de traducción, por ejemplo, trasladando los códigos VNDL a un módulo 426 de ejecución. A continuación, el módulo 426 de ejecución puede trasladar los códigos de VN a un analizador sintáctico 428 para analizar sintácticamente los códigos VNDL. Durante el análisis sintáctico, los códigos VNDL se pueden traducir o convertir en objetos internos 430, que sean entendibles por una API interna 432 de VNC en el controlador 420 de red. Los objetos internos 430 pueden representar la VN descrita en los códigos VNDL y acciones para describir la gestión de eventos de la red. Los objetos internos 430 se pueden guardar en un directorio de objetos del controlador 420 de red. Los objetos internos 430 se pueden implementar en forma de cualesquiera códigos, estructuras de datos, guiones de instrucciones, órdenes, u otra información, siempre que el controlador de red pueda entender los objetos internos 430 suficientemente como para llevar a cabo tareas relacionadas con la VN.

El módulo 426 de ejecución puede entender cómo gestionar las tareas definidas por códigos VNDL. Específicamente, el módulo 426 de ejecución puede llamar a la API interna 432 del VNC (puede considerarse como una API hacia el sur (en inglés, *southbound*)) para llevar a cabo tareas relacionadas con la VN de acuerdo con o sobre la base de los objetos internos 430 traducidos a partir de los códigos VNDL. La API interna 432 del VNC puede llevar a cabo varias tareas, tales como crear, suprimir, actualizar, consultar estados y gestionar aspectos de la VN, incluidos nodos, puertos, enlaces, políticas y eventos.

Sí se produce un cierto evento en la VN controlada por el controlador 420 de red, un manejador 434 de eventos puede gestionar de manera correspondiente el evento. Por ejemplo, si un nodo de red, un puerto o un enlace entre dos nodos se cae, el manipulador 434 de eventos puede trasladar información del evento al módulo 426 de ejecución, el cual, a continuación, puede gestionar el evento usando políticas predefinidas. La plataforma del controlador 420 de red se puede implementar usando cualquier lenguaje o entorno adecuado. Por ejemplo, puede usarse un módulo 436 de tiempo de ejecución Python para implementar módulos, según se muestra en la FIG. 4.

Las FIGS. 5A y 5B ilustran un esquema 500 de XML ilustrativo que describe por lo menos partes o algunos aspectos de una VN. Alguien con conocimientos habituales en la materia entenderá la sintaxis del esquema 500 de XML, por lo que las descripciones adicionales pueden centrarse en características novedosas que se dan a conocer en la presente. Obsérvese que los nombres y los escenarios usados en el esquema 500 de XML son ejemplificativos y no limitativos. El esquema 500 de XML usa elementos para definir una pluralidad de nombres lícitos para describir una VN. Por ejemplo, el esquema 500 de XML comprende un elemento 510 denominado “red” (obsérvese que elemento 510 cruza múltiples líneas de código). Por lo tanto, cuando un fragmento de códigos VNDL contiene el término “red”, el mismo puede ser reconocido por una entidad de procesado (por ejemplo, un VNC) como nombre lícito válido para una VN y se puede procesar en consecuencia. Términos no definidos, tales como “redes”, se pueden identificar como erróneos.

En la práctica, una red se puede identificar con una identificación (id) y puede comprender nodos y enlaces de red en la misma. Para describir de forma más detallada una VN, el elemento 510 puede comprender subelementos o elementos hijos que definen otros nombres lícitos. Por ejemplo, un elemento hijo 512 denominado “id” puede especificar una identificación de la VN, un elemento hijo 514 denominado “nodo” puede especificar un nodo dentro de la VN, y un elemento 522 denominado “enlace” puede especificar un enlace o conexión de red dentro de la VN.

En la medida en la que un nodo se puede identificar con una id, una dirección, un nombre de nodo y/o puerto(s), el elemento 514 comprende, además, elementos hijo que describen el nodo. Específicamente, los elementos hijo 515, 516, 517 y 518 describen una id, una dirección, un nombre, y un puerto del nodo, respectivamente. Además, el elemento 518 comprende un elemento hijo 519 que describe un nombre del puerto.

De manera similar, un enlace puede tener diversos atributos, tales como un nodo de origen, un nodo de destino, y un ancho de banda. Para describir un enlace representado por el elemento 522, un elemento hijo 523 denominado “de” (en inglés, “from”) especifica un nodo de origen del enlace, un elemento hijo 524 denominado “a” (en inglés, “to”) especifica un nodo de destino del enlace, y un elemento hijo 525 denominado “ancho de banda” (en inglés, “bandwidth”) especifica un ancho de banda de la red asignado al enlace.

La FIG. 6A ilustra un fragmento ilustrativo de códigos 600 de programa que describe una VN ejemplificativa 650, y la FIG. 6B ilustra la VN 650, que se puede crear usando los códigos 600 de programa. Usando un planteamiento declarativo dado a conocer en la presente, un desarrollador de aplicaciones puede seguir un esquema de XML (por ejemplo, el esquema 500 de XML) para escribir los códigos 600 de programa. Los códigos 600 de programa pueden ser enviados por una aplicación (por ejemplo, la aplicación 410) a un controlador de red (por ejemplo, el controlador 420), el cual, a continuación, puede crear la VN 650 de acuerdo con los códigos 600 de programa.

Los nombres lícitos definidos en el esquema de XML se pueden poner en mayúsculas en los códigos de programa. Tal como se muestra en la FIG. 6A, los nombres lícitos que incluyen nodo, puerto, enlace y ancho de banda (BW) se escriben en mayúsculas. Basándose en los códigos 600 de programa, la red que se va a crear tiene un nombre ilustrativo “Prueba” (en inglés, “Test”) y comprende siete nodos de red denominados del 1 a 7. Por ejemplo, la línea 610 de los códigos 600 de programa especifica que el Nodo 1 comprende dos puertos denominados Puerto 1 y Puerto 2, que se muestran en la FIG. 6B con nodos representados con círculos de color oscuro y puertos representados con recuadros ovalados unidos a los círculos.

Además, pueden establecerse enlaces entre puertos. Por ejemplo, la línea 620 de los códigos 600 de programa comprende “Enlace 1.2=3.1” (en inglés, “Link 1.2=3.1”), lo cual establece un enlace (mostrado como enlace 652 en la FIG. 6B) del Nodo 1: Puerto 2 al Nodo 3: Puerto 1. Obsérvese que la línea 620 especifica además que debe asignarse al enlace 652 un ancho de banda de 1 gigabyte (G). Pueden establecerse otros nodos, puertos y enlaces de la misma manera usando los códigos 600 de programa.

Alguien con conocimientos habituales en la materia entenderá que los códigos de programa escritos en VNDL se pueden ampliar en su totalidad para incluir otra información, tal como descripciones para gestionar eventos e imponer políticas fijadas por las aplicaciones. Puesto que, en una VN, pueden producirse diversos eventos, sus políticas de gestión pueden estar configuradas por desarrolladores de aplicaciones y se pueden especificar en códigos VNDL. Por ejemplo, cuando un nodo de red funciona defectuosamente, códigos VNDL pueden definir una política para restablecer el nodo o encontrar un nodo alternativo. Un puerto o un enlace entre puertos se puede gestionar de manera similar. Como ejemplo alternativo, cuando la utilización de un enlace físico supera un cierto porcentaje, una de las políticas puede trasladar un enlace de VN relacionado desde el enlace físico a otro enlace físico. Todavía como ejemplo alternativo, cuando un recuento de caída de una interfaz llega a un cierto umbral, una de las políticas puede seleccionar el uso de otra interfaz. En general, los principios dados a conocer en la presente se pueden aplicar a un sin número de otros eventos y políticas de ejemplo.

Usando un lenguaje de programación declarativo en lugar de uno procedimental, los desarrolladores de aplicaciones pueden llevar a cabo tareas de la VN sin conocer o entender los detalles de las redes. Un controlador de red que traduce los códigos VNDL puede garantizar la red y los comportamientos de la red deseados. Los desarrolladores de aplicaciones pueden usar una API RESTful para comunicarse con un VNC, y explotar una VN a través del VNC. Por ejemplo, pueden usarse códigos VNDL para consultar el estado de un nodo, un puerto o un enlace. También pueden usarse códigos VNDL para efectuar operaciones, tales como la adición o eliminación de un nodo, puerto o enlace específico.

El planteamiento declarativo puede aportar varias ventajas. La productividad del programador puede verse incrementada (por ejemplo, en experimentos la longitud de códigos de programa se ha reducido en un 90%), puede mejorarse la precisión de la programación ya que los programadores pueden centrarse en las sentencias declarativas en lugar de en los pasos detallados, y se puede mejorar la visibilidad con una descripción y un comportamiento concisos de la red ya que los desarrolladores pueden usar la sentencia declarativa para construir diagramas visuales de redes.

La FIG. 7 es un diagrama de flujo de una realización de ejemplo de un método 700 de gestión de VN, que se puede implementar mediante un controlador de red (por ejemplo, el PNC 220, el PNC 230, el VNC 240, el controlador 320 o el controlador 420). El método 700 comienza a partir del paso 710, donde el controlador de red puede recibir una solicitud de una aplicación por medio de una API. La solicitud puede comprender códigos de programa escritos en un lenguaje de programación declarativo, y los códigos de programa pueden describir por lo menos algunos aspectos de una VN. En función de la necesidad de la aplicación, los códigos de programa pueden contener contenidos diferentes. Por ejemplo, los códigos de programa pueden comprender instrucciones para crear la VN, una o más políticas para la explotación de la VN, o la definición de eventos que se producen en la VN. En una realización de ejemplo, el lenguaje de programación declarativo puede usar un esquema de XML con nombres lícitos definidos, y los nombres lícitos usados en la descripción de la VN pueden comprender uno o más de nodo, puerto, enlace, ancho de banda, política y evento. Además, la aplicación se puede acoplar de manera remota al controlador de red o se puede ubicar conjuntamente en un mismo lugar con dicho controlador, con lo cual la recepción puede ser remota o local.

En el paso 720, el controlador de red puede analizar sintácticamente los códigos de programa obteniendo objetos internos del controlador de red. Los objetos internos representan los aspectos de la VN descritos por los códigos de programa. El análisis sintáctico se puede implementar de cualquier manera adecuada para traducir o convertir los códigos VNDL en los objetos internos.

En el paso 730, el controlador de red puede gestionar la VN usando una API interna del controlador de red de acuerdo

con los objetos internos, que son entendibles por la API interna. Obsérvese que la gestión de la VN puede abarcar la creación de la VN, la actualización de la VN, la gestión de políticas de la VN, y la gestión de eventos que se produzcan en la VN. Debe entenderse que el método 700 sirve como realización de ejemplo, por lo que pueden usarse alternativas para modificar el método 700 y se pueden incorporar pasos adicionales según sea necesario.

La FIG. 8 es un diagrama esquemático de una realización de ejemplo de un sistema de ordenador o dispositivo 800 de red. El dispositivo 800 de red se puede implementar en forma de cualquier dispositivo adecuado, tal como un nodo de red que ejecute aplicaciones o un controlador de red (por ejemplo, el PNC 220, el PNC 230, el VNC 240, el controlador 320 o el controlador 420) dado a conocer en la presente. El dispositivo 800 de red puede ser capaz de recibir, procesar y transmitir mensajes portadores de información, tales como solicitudes relacionadas con la VN, hacia y desde una red. El dispositivo 800 de red puede comprender uno o más puertos 810 de entrada acoplados a un receptor (Rx) 812, que se puede configurar para recibir solicitudes, respuestas, códigos de programa y otra información de otros componentes de red. El dispositivo 800 de red puede comprender, además, uno o más puertos 830 de salida acoplados a un transmisor (Tx) 832, que se puede configurar para transmitir solicitudes, respuestas, códigos de programa y otra información a otros componentes de la red.

El dispositivo 800 de red puede comprender una unidad lógica o procesador 820 que esté en comunicación con el receptor 812 y el transmisor 832. El procesador 820 se puede implementar usando *hardware* o una combinación de *hardware* y *software*. El procesador 820 se puede implementar en forma en forma de uno o más chips de unidad de procesador central (CPU), núcleos (por ejemplo, un procesador multinúcleo), matrices de puertas programables in situ (FPGAs), circuitos integrados de aplicación específica (ASICs) y/o procesadores de señal digitales (DSPs). El procesador 820 se puede configurar para implementar cualquiera de los módulos o unidades funcionales descritos en la presente, tales como el servidor web 221, el servidor web 241, el módulo 242 de servicio de VNC, el módulo 243 de servicio de TAL, los módulos 244 de servicio de RC, el módulo 245 de expresión de VNC, el servidor HTTP 422, el módulo principal 424, el módulo 426 de ejecución, el analizador sintáctico 428, la API interna 432 de VNC, el manejador 434 de eventos, y el módulo 436 de tiempo de ejecución de Python, o cualquier otro componente funcional conocido por alguien con conocimientos habituales en la materia, o combinaciones de ellos. Por ejemplo, un módulo 821 de expresión de VNC se puede incorporar en el procesador 820 y se puede configurar para procesar códigos VNDL para tareas de la VN.

El dispositivo 800 de red puede comprender, además, por lo menos una memoria 822, que se puede configurar para almacenar datos, tales como códigos de programa y objetos internos de una VNC. Obsérvese que, en la práctica, puede existir tráfico bidireccional procesado por el dispositivo 800 de red, con lo cual algunos puertos pueden tanto recibir como transmitir paquetes. En este sentido, los puertos 810 de entrada y los puertos 830 de salida pueden estar ubicados conjuntamente o se pueden considerar funcionalidades diferentes de los mismos puertos que están acoplados a transceptores (Rx/Tx). El procesador 820, la memoria 822, el receptor 812 y el transmisor 832 también se pueden configurar para implementar o admitir cualquiera de las realizaciones de ejemplo antes descritas, tales como el método 700 de gestión de VN.

Se entiende que, al programar y/o cargar instrucciones ejecutables en el dispositivo 800 de red, cambia por lo menos uno de entre el procesador 820 y la memoria 822, transformando el dispositivo 800 de red, en parte, en una máquina o aparato particular (por ejemplo, un nodo de red que tiene la funcionalidad revelada por la presente exposición). Las instrucciones ejecutables se pueden almacenar en la memoria 822 y se pueden cargar en el procesador 820 para su ejecución. Es fundamental para las técnicas de la ingeniería eléctrica y la ingeniería informática que la funcionalidad que se puede implementar cargando *software* ejecutable en un ordenador se pueda convertir a una implementación de *hardware* mediante reglas de diseño bien conocidas. Las decisiones entre la implementación de un concepto en *software* con respecto a *hardware* giran, típicamente, en torno a consideraciones de estabilidad del diseño y de los números de unidades a producir más que de cualesquiera problemas involucrados en pasar del dominio del *software* al dominio del *hardware*. En general, puede ser preferible implementar en *software* un diseño que siga estando sujeto a cambios frecuentes, ya que rehacer una implementación de *hardware* es más costoso económicamente que rehacer un diseño de *software*. En general, puede que sea preferible implementar en *hardware* un diseño que sea estable y que se producirá en un volumen elevado, por ejemplo en un ASIC, ya que, para grandes tiradas de producción, la implementación en *hardware* puede resultar menos costosa económicamente que la implementación en *software*. Frecuentemente, un diseño se puede desarrollar y someter a prueba en forma de *software* y posteriormente se puede transformar, mediante reglas de diseño bien conocidas, a una implementación de *hardware* equivalente en un circuito integrado de aplicación específica que realice un conexionado permanente de las instrucciones del *software*. De la misma manera, en la medida en la que una máquina controlada por un ASIC nuevo es una máquina o aparato particular, asimismo un ordenador que se haya programado y/o cargado con instrucciones ejecutables puede considerarse como una máquina o aparato particular.

Cualquier procesado de la presente exposición se puede implementar consiguiendo que un procesador (por ejemplo, una CPU de propósito general) ejecute un programa de ordenador. En este caso, se puede proporcionar un producto de programa de ordenador a un ordenador o un dispositivo de red usando cualquier tipo de soporte legible por ordenador, no transitorio. El producto de programa de ordenador se puede almacenar en un soporte no transitorio legible por ordenador en el ordenador o el dispositivo de red. Los soportes no transitorios legibles por ordenador incluyen cualquier tipo de soportes de almacenamiento tangibles. Los ejemplos de soportes no transitorios legibles por ordenador incluyen soportes de almacenamiento magnético (por ejemplo, discos flexibles, cintas magnéticas,

unidades de disco duro, etcétera), soportes de almacenamiento magnético óptico (por ejemplo, magneto-ópticos), disco compacto con memoria de solo lectura (CD-ROM), disco compacto grabable (CD-R), disco compacto regrabable (CD-R/W), disco digital versátil (DVD), disco Blu-ray (marca comercial registrada) (BD) y memorias de semiconductores, tales como ROM de máscara, ROM programable (PROM), PROM borrable, ROM *flash*, y memoria de acceso aleatorio (RAM). El producto de programa de ordenador también se puede proporcionar a un ordenador o un dispositivo de red usando cualquier tipo de soporte transitorio legible por ordenador. Los ejemplos de soportes transitorios legibles por ordenador incluyen señales eléctricas, señales ópticas y ondas electromagnéticas. Los medios transitorios legibles por ordenador pueden proporcionar programa a un ordenador por medio de una línea de comunicaciones por cable (por ejemplo, hilos eléctricos y fibras ópticas) o una línea de comunicaciones inalámbrica.

Por consiguiente, el alcance de protección no queda limitado por la descripción antes expuesta sino que se define por las reivindicaciones que se ofrecen seguidamente.

La descripción de una referencia en la exposición no significa admitir que la misma sea técnica anterior, especialmente cualquier referencia que tenga una fecha de publicación posterior a la fecha de prioridad de esta solicitud.

REIVINDICACIONES

1. Un método implementado por un controlador de red para la gestión de redes virtuales, VN, en donde el controlador de red comprende un controlador (240) de VN y uno o más controladores de red físicos, PNCs, (220, 230), estando configurado el VNC para gestionar y coordinarse con los múltiples PNCs, y estando configurado cada PNC para gestionar una pluralidad de nodos físicos (211 a 213, ó 214 a 216) de red; comprendiendo el método:

recibir una solicitud por medio de una interfaz de programación de aplicaciones, API, en donde la solicitud comprende códigos de programa escritos en un lenguaje de programación declarativo, y en donde los códigos de programa describen por lo menos algunos aspectos de una VN; y

analizar sintácticamente los códigos de programa obteniendo objetos internos del controlador de red, en donde los objetos internos representan los aspectos de la VN descritos por los códigos de programa;

en donde la VN comprende nodos de red, puertos en los nodos de red, y enlaces entre los puertos, y

en donde los códigos de programa comprenden instrucciones para crear la VN, en donde el método comprende, además, crear la VN usando una API interna del controlador de red de acuerdo con los objetos internos, que son entendibles por la API interna, y

en donde la creación de la VN comprende establecer por lo menos uno de los nodos de red, los puertos, los enlaces y anchos de banda asignados a los enlaces, y

en donde el lenguaje de programación declarativo usa un esquema de lenguaje de marcado extensible, XML, con nombres lícitos definidos, en donde los nombres lícitos usados en la descripción de la VN comprenden uno o más de nodo, puerto, enlace, ancho de banda, política y evento.

2. El método de la reivindicación 1, en donde los códigos de programa comprenden una o más políticas para la explotación de la VN, y en donde el método comprende, además, imponer la política o políticas usando una API interna del controlador de red de acuerdo con los objetos internos.

3. El método de la reivindicación 1, que comprende, además, gestionar uno o más eventos que se producen en la VN de acuerdo con los objetos internos, en donde la gestión de los eventos comprende usar una API interna del controlador de red.

4. El método de la reivindicación 1, que comprende, además, comunicación con controladores de red adicionales gestionados por el controlador de red para la explotación de la VN, en donde la solicitud recibida por el controlador de red se genera mediante una aplicación explotada por un desarrollador de aplicaciones y acoplada de forma remota al controlador de red por medio de la API, y en donde la API es una API RESTful.

5. Un producto de programa de ordenador almacenado en un controlador de red para la gestión de redes virtuales, VN, en donde el controlador de red comprende un controlador (240) de VN y uno o más controladores de red físicos, PNCs, (220, 230), estando configurado el VNC para gestionar y coordinarse con los múltiples PNCs, y estando configurado cada PNC para gestionar una pluralidad de nodos físicos (211 a 213, ó 214 a 216) de red; comprendiendo el producto de programa de ordenador instrucciones ejecutables por ordenador almacenadas en un soporte no transitorio legible por ordenador de tal manera que, cuando son ejecutadas por un procesador, provocan que el controlador de red:

obtenga una solicitud de una aplicación por medio de una interfaz de programación de aplicaciones, API, en donde la solicitud comprende códigos de programa escritos en un lenguaje de programación declarativo, y en donde los códigos de programa describen por lo menos algunos aspectos de una red virtual, VN; y

analice sintácticamente los códigos de programa obteniendo objetos internos del controlador de red, en donde los objetos internos representan los aspectos de la VN descritos por los códigos de programa;

en donde la VN comprende nodos de red, puertos en los nodos de red, y enlaces entre los puertos, y

en donde los códigos de programa comprenden instrucciones para crear la VN, en donde el método comprende, además, crear la VN usando una API interna del controlador de red de acuerdo con los objetos internos, que son entendibles por la API interna, y en donde la creación de la VN comprende establecer por lo menos uno de los nodos de red, los puertos, los enlaces y anchos de banda asignados a los enlaces;

en donde el lenguaje de programación declarativo usa un esquema de lenguaje de marcado extensible, XML, con nombres lícitos definidos, en donde los nombres lícitos usados en la descripción de la VN comprenden uno o más de nodo, puerto, enlace, ancho de banda, política y evento.

6. El producto de programa de ordenador de la reivindicación 5, en donde la API es una API RESTful, y en donde el controlador de red comprende un módulo de expresión de controlador de VN, VNC, implementado usando Python y configurado para analizar sintácticamente los códigos de programa obteniendo los objetos internos.

7. Un aparato implementado como controlador de red para la gestión de redes virtuales, VN, en donde el controlador de red comprende un controlador (240) de VN y uno o más controladores de red físicos, PNCs, (220, 230), estando configurado el VNC para gestionar y coordinarse con los múltiples PNCs, y estando configurado cada PNC para gestionar una pluralidad de nodos físicos (211 a 213, ó 214 a 216) de red; comprendiendo el aparato:

un receptor (420) configurado para recibir una solicitud (412) por medio de una interfaz de programación de aplicaciones, API (421), en donde la solicitud comprende códigos de programa escritos en un lenguaje de programación declarativo, y en donde los códigos de programa describen por lo menos algunos aspectos de una red virtual, VN; y

un procesador (420) acoplado al receptor y configurado para analizar sintácticamente los códigos de programa obteniendo objetos internos del controlador de red, en donde los objetos internos representan los aspectos de la VN descritos por los códigos de programa;

en donde la VN comprende nodos de red, puertos en los nodos de red, y enlaces entre los puertos, y

en donde los códigos de programa comprenden instrucciones para crear la VN, en donde las instrucciones comprenden, además, crear la VN usando una API interna del controlador de red de acuerdo con los objetos internos, que son entendibles por la API interna, y en donde la creación de la VN comprende establecer por lo menos uno de los nodos de red, los puertos, los enlaces y anchos de banda asignados a los enlaces;

en donde el lenguaje de programación declarativo usa un esquema de lenguaje de marcado extensible, XML, con nombres lícitos definidos, en donde los nombres lícitos usados en la descripción de la VN comprenden uno o más de nodo, puerto, enlace, ancho de banda, política y evento.

8. El aparato de la reivindicación 7, que comprende, además, un transmisor acoplado al procesador y configurado para transmitir órdenes a uno o más controladores de red adicionales con vistas a la explotación de la VN, en donde las órdenes son generadas por el procesador sobre la base de los objetos internos, en donde los controladores de red adicionales están acoplados al aparato y son gestionados por el mismo, el cual sirve como controlador de VN, VNC, de capa superior.

9. El aparato de la reivindicación 7, en donde el procesador está configurado, además, para gestionar la VN de acuerdo con los objetos internos, en donde la gestión de la VN comprende crear la VN, actualizar la VN, gestionar políticas de la VN, y gestionar eventos que se producen en la VN.

10. El aparato de la reivindicación 7, en donde la API es una API RESTful, y en donde el procesador comprende un módulo de expresión de controlador de VN, VNC, implementado usando Python y configurado para analizar sintácticamente los códigos de programa obteniendo los objetos internos.

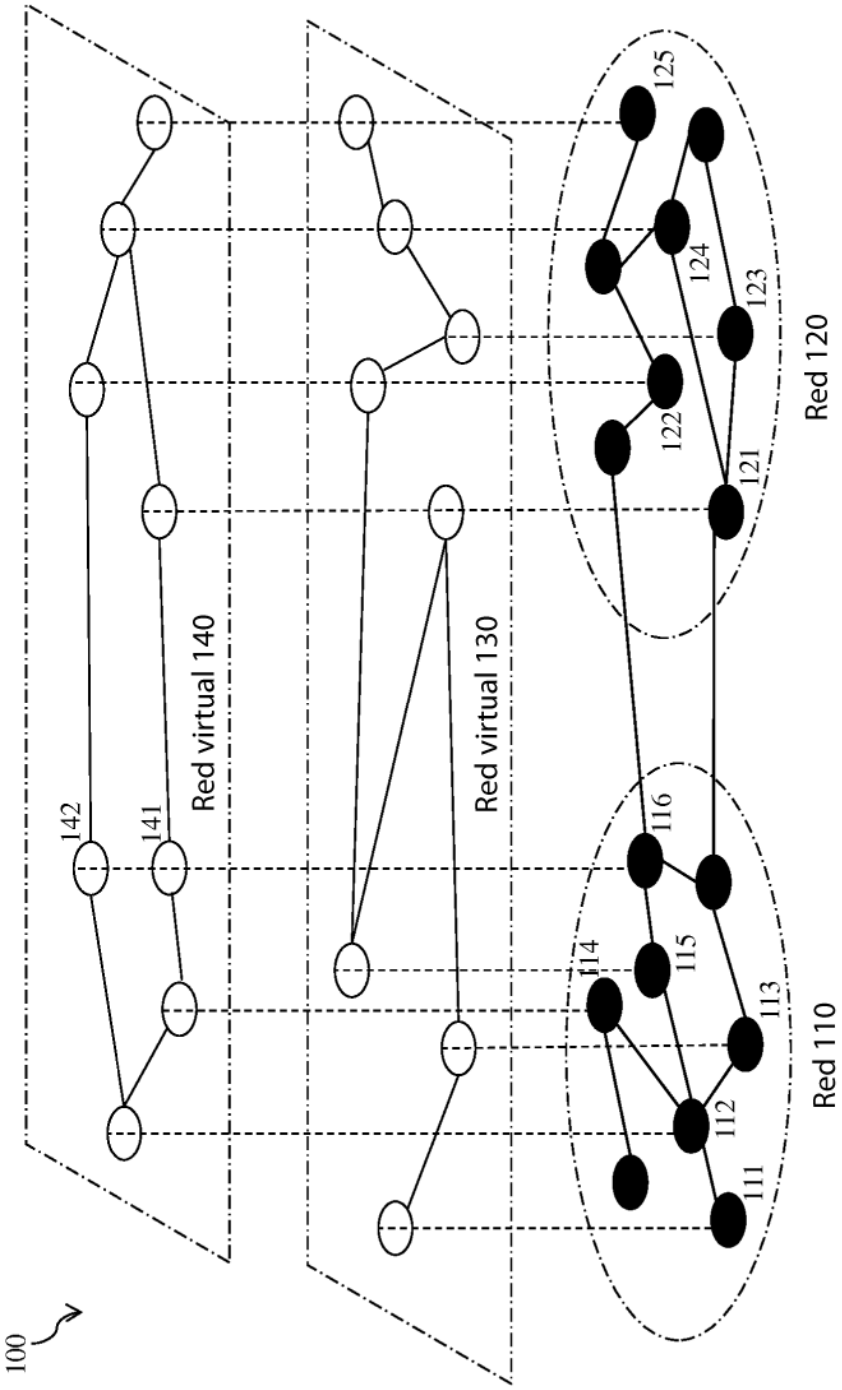
100
Red virtual 140
Red virtual 130
Red 120
Red 110
142
141
125
124
123
122
121
116
115
114
113
112
111

FIG. 1

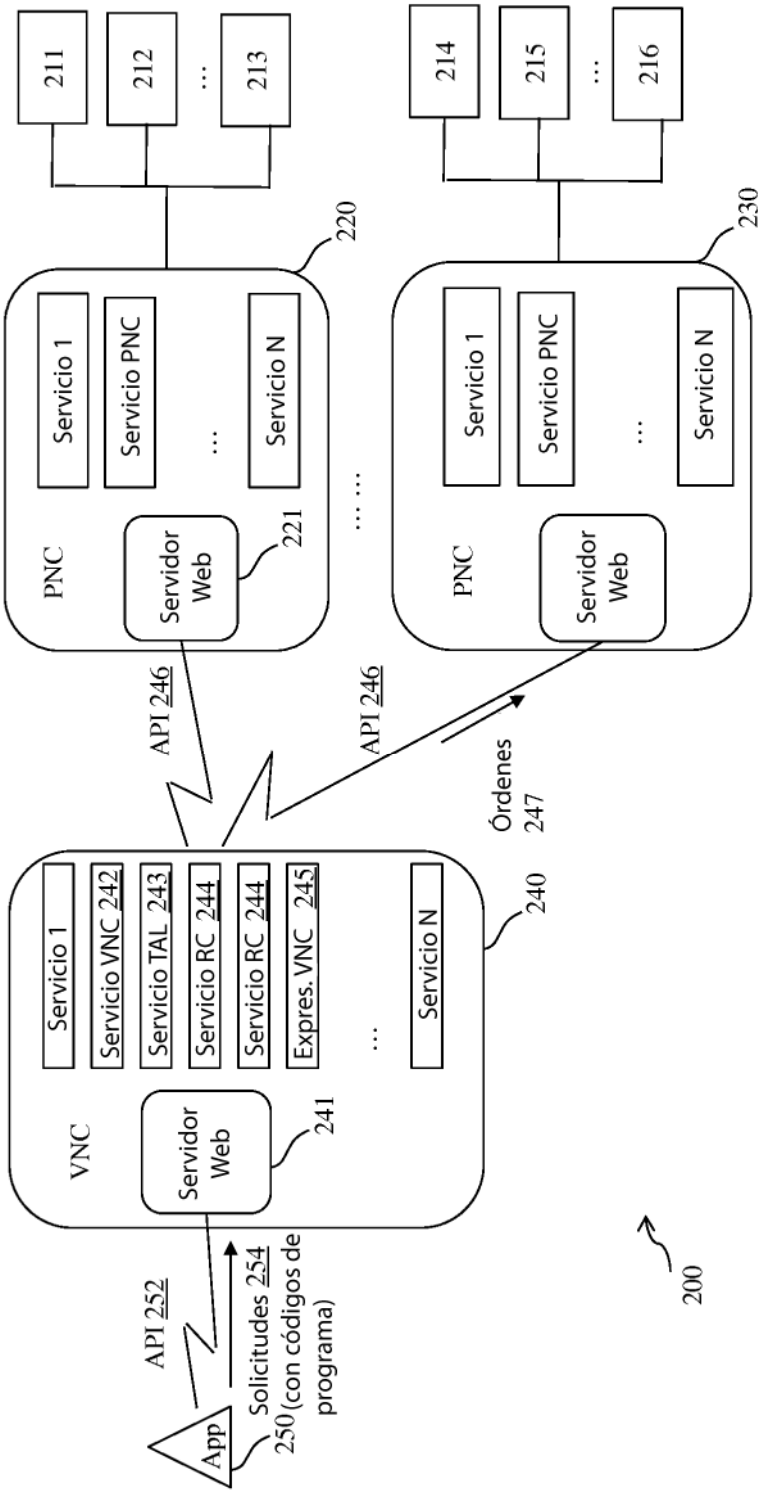
211
212
213
214
215
216
220
230
Servicio 1
Servicio PNC
Servicio N
PNC
Servidor Web
221
API 246
API 246
Órdenes 247
Servicio 1
Servicio VNC 242
Servicio TAL 243
Servicio RC 244
Servicio RC 244
Expres. VNC 245
Servicio N
240
VNC
Servidor Web
241
API 252
Solicitudes 254 (con códigos de programa)
App
250
200

FIG. 2

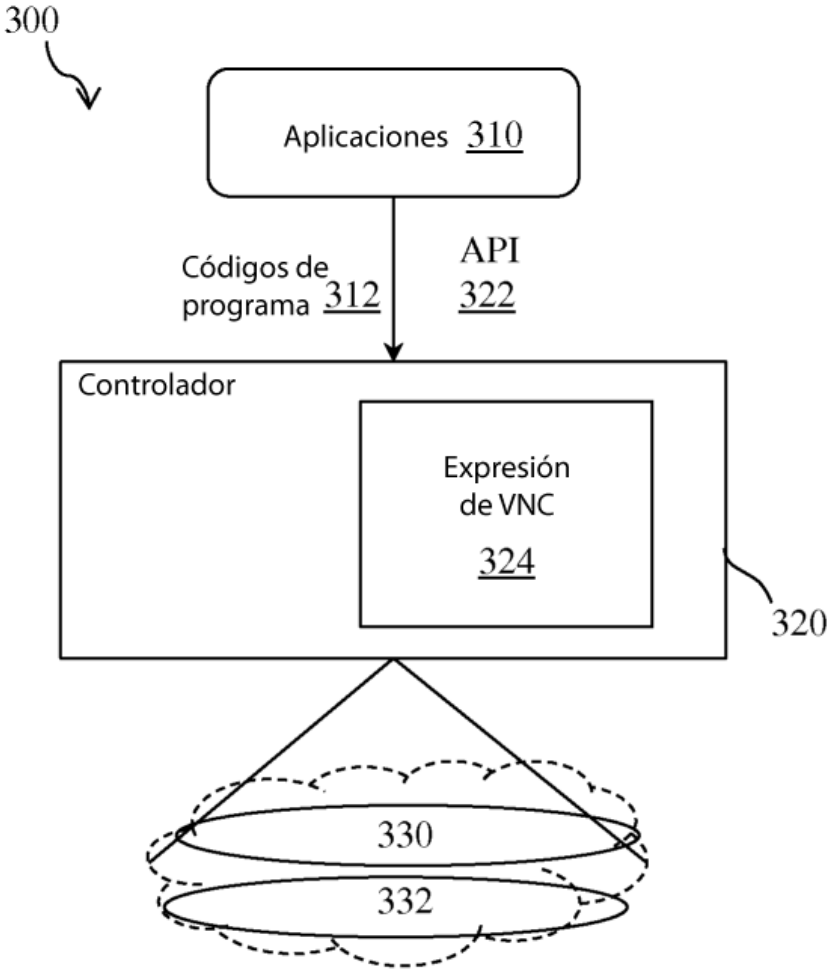
300
Aplicaciones 310
Códigos de programa 312
API 322
Controlador
Expresión de VNC 324
320
330
332

FIG. 3A

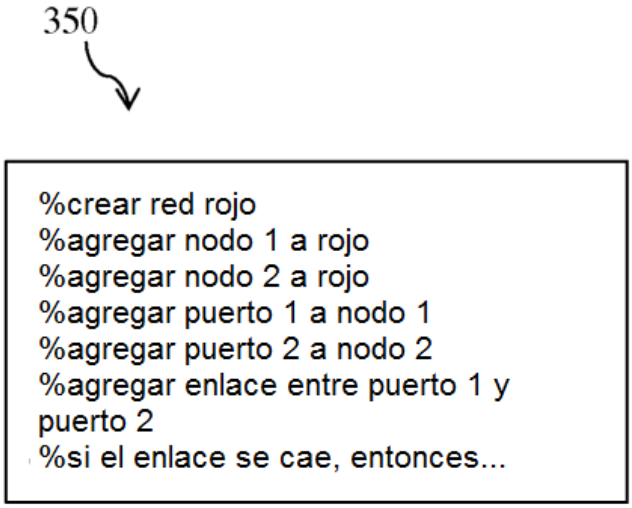
350
%crear red rojo
%agregar nodo 1 a rojo
%agregar nodo 2 a rojo
%agregar puerto 1 a nodo 1
%agregar puerto 2 a nodo 2
%agregar enlace entre puerto 1 y puerto 2
%si el enlace se cae, entonces...

FIG. 3B

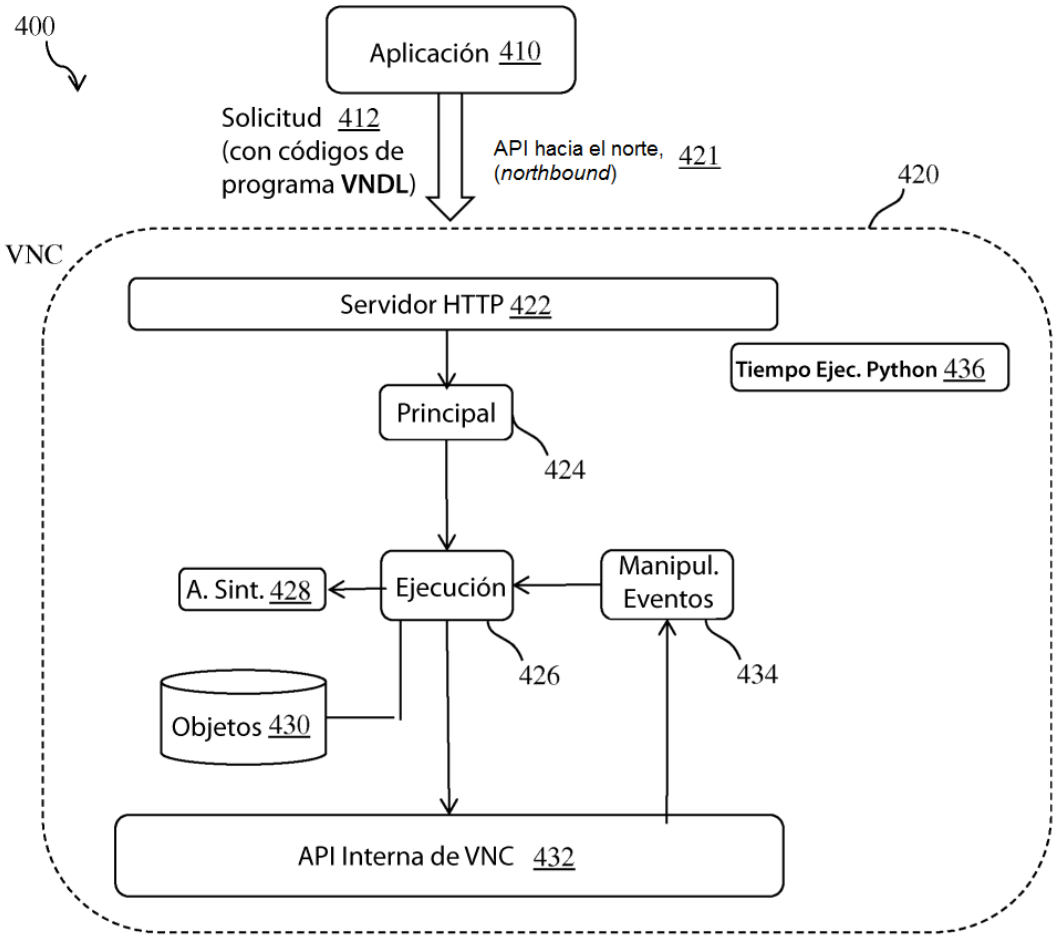
400
Aplicación 410
Solicitud 412
(con códigos de
programa VNDL)
API hacia el norte, 421
(northbound)
420
VNC
Servidor HTTP 422
Tiempo Ejec. Python 436
Principal
424
A. Sint. 428
Ejecución
Manipul.
Eventos
426
434
Objetos 430
API Interna de VNC 432

FIG. 4

500

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
< xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
< xs:element name="network"> ◄—— 510
        <xs:complexType>
        <xs:sequence>
                <xs:element name=“id" type="xs:string"/>◄—— 512
                <xs:element name="node"> ◄—— 514
                        <xs:complexType>
                        <xs:sequence>
                                <xs:element name= "id" type="xs:string"/> ◄—— 515
                                <xs:element name="address" type="xs:string"/> ◄—— 516
                                <xs:element name= "name" type="xs:string"/> ◄—— 517
                                        518 ——► <xs:element name=“port" type="xs:string"/>
                                                <xs:complexType>
                                                519 ——► <xs:element name=“" type="xs:string"/>
                                                </xs:complexType>
                                        </xs:element>
                                </xs:sequence>
                        </xs:complexType>
                </xs:element>
        <xs:complexType>
        <xs:sequence>
        … …
< /xs:element>
< /xs:schema>
```

FIG. 5A

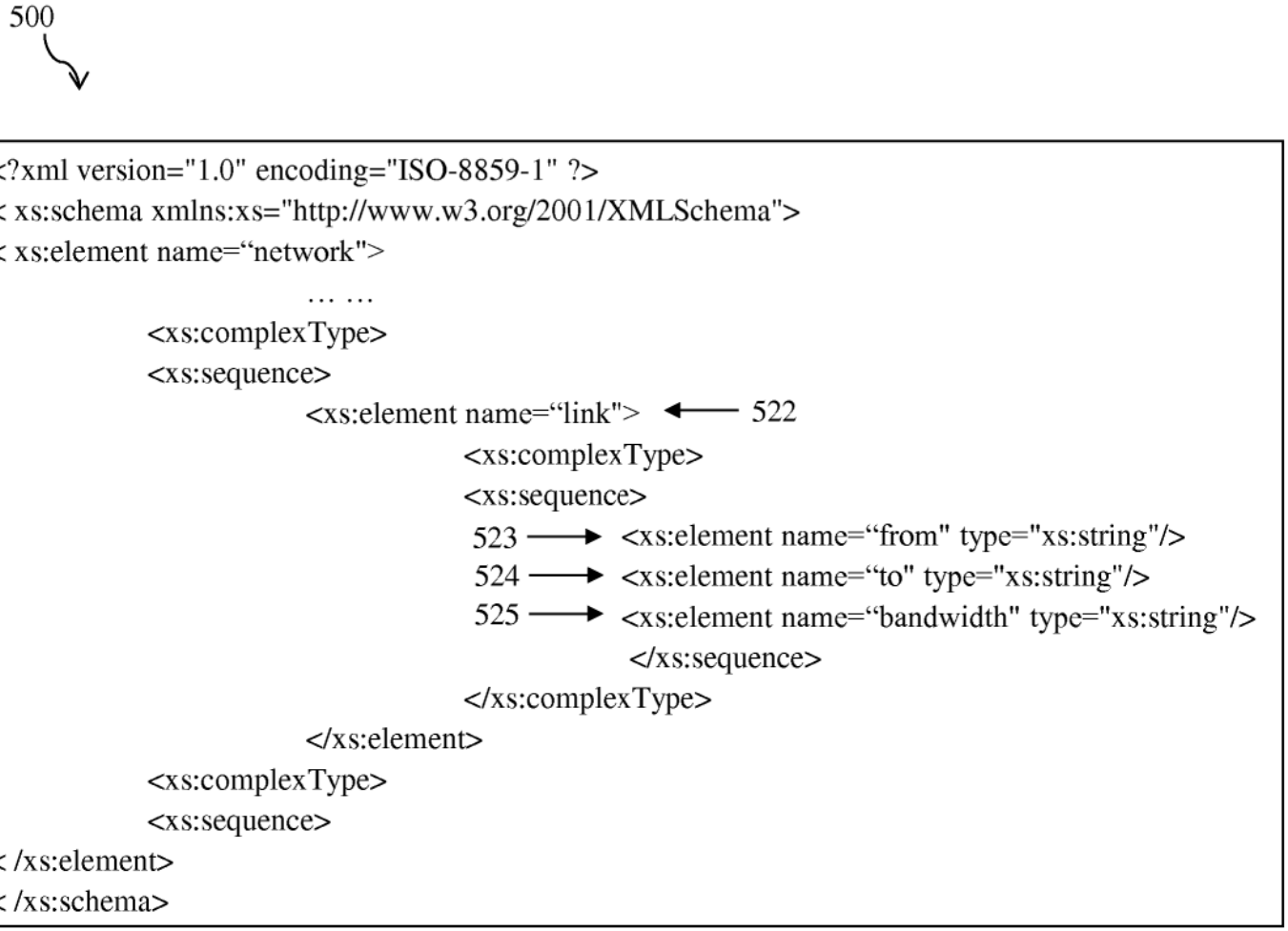
500
<?xml version="1.0" encoding="ISO-8859-1" ?>
< xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
< xs:element name="network">
… …
<xs:complexType>
<xs:sequence>
<xs:element name="link"> ← 522
<xs:complexType>
<xs:sequence>
523 → <xs:element name="from" type="xs:string"/>
524 → <xs:element name="to" type="xs:string"/>
525 → <xs:element name="bandwidth" type="xs:string"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:complexType>
<xs:sequence>
< /xs:element>
< /xs:schema>

FIG. 5B

600

```
<NETWORK name="Test">
    <NODE name="1"><PORT name="1"></PORT><PORT name="2"></PORT></NODE>   ← 610
    <NODE name="2"><PORT name="1"></PORT><PORT name="2"></PORT></NODE>
    <NODE name="3"><PORT name="1"></PORT><PORT name="2"></PORT><PORT name="3"></PORT>
                    <PORT name="4"></PORT<PORT name="5"></PORT</NODE>
    <NODE name="4"><PORT name="1"></PORT><PORT name="2"></PORT><PORT name="3"></PORT></NODE>
    <NODE name="5"><PORT name="1"></PORT><PORT name="2"></PORT>
                    <PORT name="3"></PORT<PORT name="4"></PORT</NODE>
    <NODE name="6"><PORT name="1"></PORT><PORT name="2"></PORT> <PORT name="3"></PORT></NODE>
    <NODE name="7"><PORT name="1"></PORT><PORT name="2"></PORT>
                     <PORT name="3"></PORT<PORT name="4"></PORT</NODE>
    <LINK 1.2=3.1><BW>1G</BW></LINK>  ← 620
    <LINK 2.2=3.2></LINK>
    <LINK 3.3=4.1></LINK>
    <LINK 3.5=5.1><BW>10G</BW></LINK>
    <LINK 3.4=6.1></LINK>
    <LINK 4.2=5.2></LINK>
    <LINK  5.3=6.2></LINK>
    <LINK 4.3=7.2></LINK>
    <LINK 5.4=7.1></LINK>
    <LINK 6.3=7.3></LINK>
</NETWORK>
```

FIG. 6A

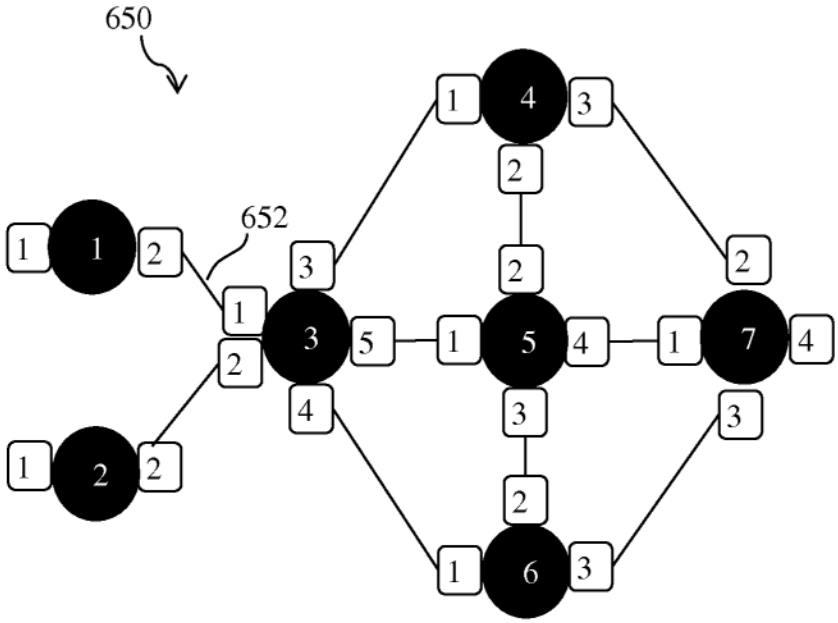
650
652
1
2
3
4
5
6
7

FIG. 6B

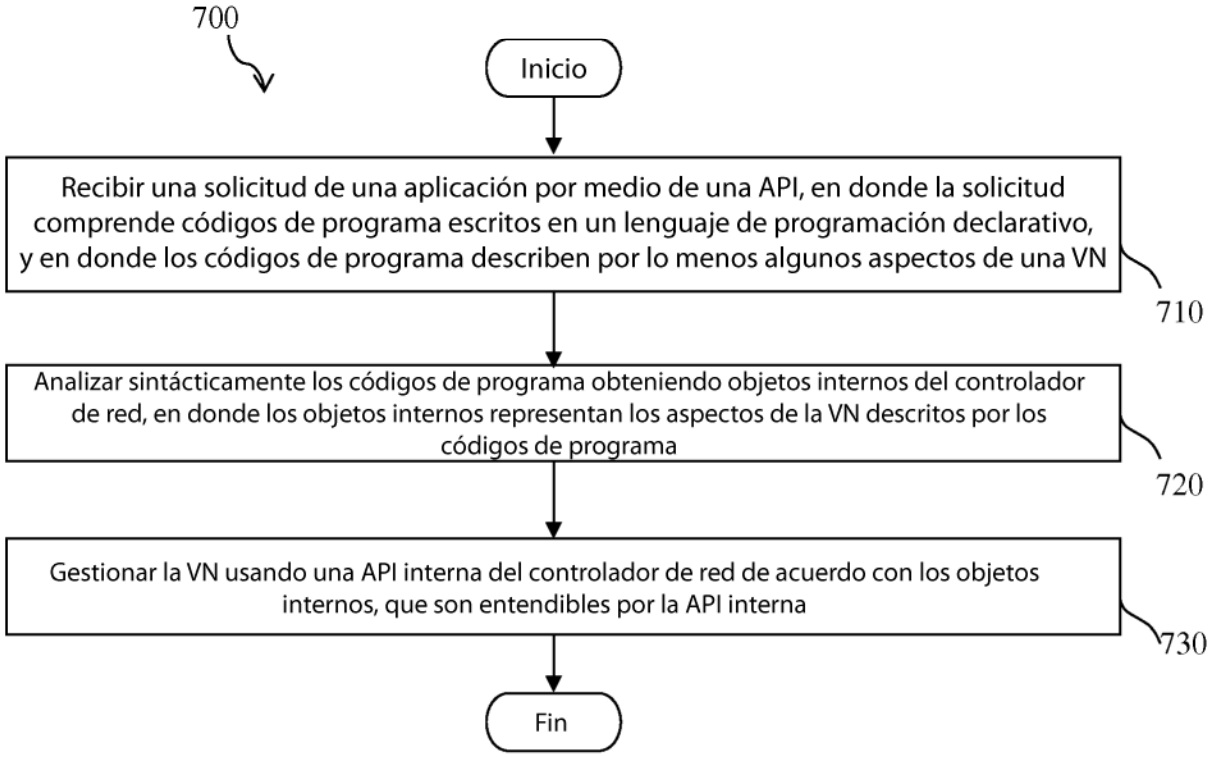
700
Inicio
Recibir una solicitud de una aplicación por medio de una API, en donde la solicitud comprende códigos de programa escritos en un lenguaje de programación declarativo, y en donde los códigos de programa describen por lo menos algunos aspectos de una VN
710
Analizar sintácticamente los códigos de programa obteniendo objetos internos del controlador de red, en donde los objetos internos representan los aspectos de la VN descritos por los códigos de programa
720
Gestionar la VN usando una API interna del controlador de red de acuerdo con los objetos internos, que son entendibles por la API interna
730
Fin

FIG. 7

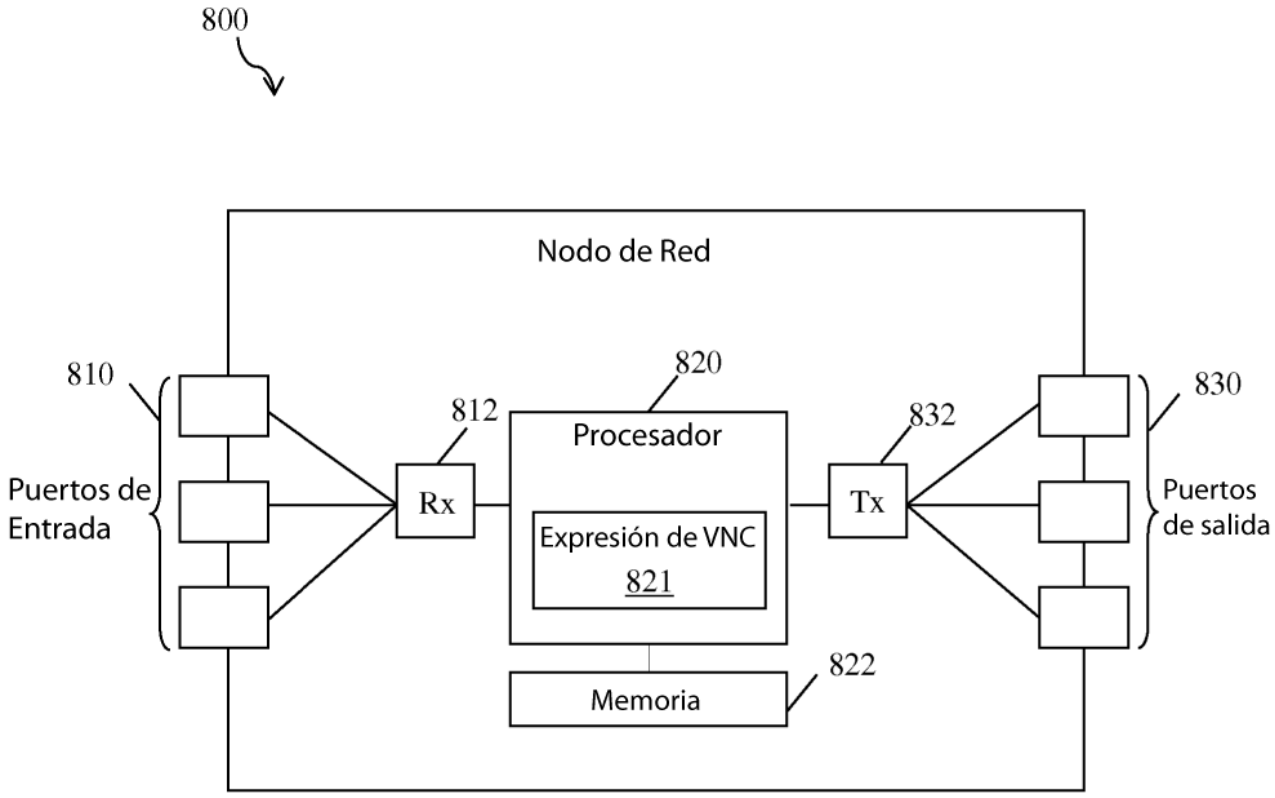
800
Nodo de Red
810
Puertos de Entrada
812
Rx
820
Procesador
Expresión de VNC
821
822
Memoria
832
Tx
830
Puertos de salida

FIG. 8