(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 659 396**

(51) Int. CI.:

H04L 29/06 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **10.02.2009 PCT/CN2009/070391**

(87) Fecha y número de publicación internacional: **07.01.2010 WO10000139**

(96) Fecha de presentación y número de la solicitud europea: **10.02.2009 E 09771909 (0)**

(97) Fecha y número de publicación de la concesión europea: **15.11.2017 EP 2302864**

(54) Título: **Método para procesar formato TLV de datos de comunicación**

(30) Prioridad:

02.07.2008 CN 200810137647

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
15.03.2018

(73) Titular/es:

ZTE CORPORATION (100.0%)
ZTE Plaza, Keji Road South, Hi-Tech Industrial Park, Nanshan District
Shenzhen, Guangdong 518057, CN

(72) Inventor/es:

ZHANG, XINPING

(74) Agente/Representante:

VALLEJO LÓPEZ, Juan Pedro

DESCRIPCIÓN

Método para procesar formato TLV de datos de comunicación

Campo de la invención

La presente invención se refiere al campo de la comunicación y, en particular, a un método para procesar un formato TLV de datos de comunicación.

Antecedentes de la invención

Notación de Sintaxis Abstracta 1 (ASN.1 para abreviar) es un método normalizado para describir un formato de mensaje transmitido en una red definida por la Unión Internacional de Telecomunicaciones (ITU para abreviar), que se usa para proporcionar un formato normalizado para conversión de datos entre nodos. Cada nodo únicamente necesita saber un formato de datos que se traduce desde la ASN.1 o traduce en la ASN.1, pero es innecesario un formato en el que existen datos en otros lugares de la red.

La ASN.1 tiene dos partes: la primera parte (ISO 8824/ITU X.208) describiendo datos, un tipo de datos y un formato de secuencia en un mensaje, que es una sintaxis de datos; y la segunda parte (ISO 8825/ITU X.209) describiendo una regla sobre cómo constituir un mensaje mediante datos de respectivas partes, que es una regla básica de codificación de datos. La ASN.1 se desarrolló en un principio como una parte de la X.409 y se convirtió en una norma independiente después. La norma de Infraestructura de Clave Pública de primera generación (PKI para abreviar) se basa principalmente en la ASN.1. En un Protocolo de Gestión de Red Simple (SNMP para abreviar), la ASN.1 se usa para definir una unidad de datos de SNMP y un formato de un objeto.

La ASN.1 se aplica ampliamente en el campo de comunicación y campo informático y otros campos. El mensaje ASN.1 tiene las siguientes propiedades especiales.

1. El mensaje es de una estructura de árbol que puede definir mensaje anidado multicapa como se desee.

2. El tipo de datos de un nodo en el mensaje podría ser un tipo de datos simples, por ejemplo, INTEGER, GRAPHICSTRING, ENUM, etc., o un tipo de datos compuestos que contiene un sub-nodo, por ejemplo, SET, SEQUENCE, SET OF, SEQUENCE OF.

3. En el tipo SET o el tipo SEQUENCE, un cierto nodo podría estar ausente ya que está permitido que el nodo de mensaje correspondiente se establezca como OPTIONAL, que significa que no es esencial. Cuando se usa un fichero ASN.1, primero se define un tipo de ASN.1 en el fichero ASN.1 y a continuación se compila mediante un compilador, de tal forma que el tipo de ASN.1 se convierte en un tipo de lenguaje de programación intermediario, tales como un lenguaje java o C++, y a continuación se usa un tipo de lenguaje de programación correspondiente para lograr el propósito de comunicación.

4. La ASN.1 soporta codificación BER, codificación DER, codificación VAL, etc., que pueden codificar un ejemplo de un tipo de lenguaje intermedio en un flujo de código y decodificar el flujo de código en un ejemplo del tipo de lenguaje intermedio correspondiente.

5. El formato de definición usado por la ASN.1 es: "NewStructName ::=defining a message", por ejemplo UserNameList-T::= SEQUENCE OF UserName-T, que indica que el tipo UserNameList-T se define como una matriz con UserName-T como un elemento.

En una arquitectura de Cliente/Servidor (C/S) de un software, el Cliente y el Servidor necesitan comunicarse entre sí. El mensaje de comunicación generalmente tiene dos formatos: un formato privado y un formato normalizado, por ejemplo, protocolos de ASN.1, SNMP, etc.

Un método común de usar el mensaje ASN.1 en un software de tipo C/S incluye:

1. establecer una conexión entre un terminal cliente y un terminal servidor;

2. el terminal cliente interactúa con el terminal servidor a través de un protocolo, que incluye:

1) un lado de envío de mensaje rellenando un encabezamiento de mensaje y un cuerpo de mensaje del mensaje ASN.1, en el que el encabezamiento de mensaje incluye un código de comandos, y el cuerpo de mensaje usa un tipo AsnAny con su tipo de datos real variando dependiendo del código de comandos;

2) el lado de envío de mensaje codificando el mensaje en un flujo de código y enviando el mismo a un lado de recepción de mensaje;

3) el lado de recepción de mensaje recibiendo el flujo de código del mensaje, decodificando el encabezamiento de mensaje y a continuación decodificando el cuerpo de mensaje usando un tipo de datos específicos de acuerdo con el código común en el mismo; y

4) el lado de recepción de mensaje haciendo una correspondiente respuesta de acuerdo con los contenidos del mensaje;

3. finalizar la interacción y desconectar la conexión.

En comunicación, datos que se constituyen de acuerdo con el formato de "tipo de datos, longitud de datos, cuerpo de datos" se representa como "Tipo, Longitud, Valor", que se llama como un formato TLV para abreviar. Es muy conveniente y altamente eficiente usar un formato TLV para constituir datos, especialmente adecuado para unos datos de longitud-variable. La constitución de los contenidos de un cuerpo de datos en una capa de aplicación generalmente usa un formato de este tipo. La codificación BER en la ASN.1 es realmente una codificación TLV.

En la técnica relacionada, un usuario no puede personalizar el tipo de datos T en la codificación TLV. Si el mismo nombre de tipo no se define de una misma manera en diferentes versiones, el Cliente y el Servidor pueden no comunicarse. Es decir, las versiones de protocolos de mensaje en el terminal cliente y terminal servidor serán consistentes. Si no son consistentes, los dos lados no pueden interactuar entre sí.

NEUFELD G ET AL: "An overview of ASN.1", COMPUTER NETWORKS AND ISDN SYSTEMS, NORTH HOLLAND PUBLISHING. AMSTERDAM, NL, vol. 23, n.º 5, 1 de febrero de 1992 (01-02-1992), páginas 393-415, XP026635867, ISSN: 0169-7552, DOI: 10.1016/0169-7752(92) 90014-H [recogido el 02-02-1992]* párrafo [0001]-párrafo [0003] proporciona respectivas soluciones técnicas; sin embargo, el problema mencionado anteriormente aún permanece sin resolverse.

Sumario de la invención

En vista de que el problema presente en las técnicas relacionadas que el Cliente y el Servidor fallan en comunicarse entre sí cuando las versiones de protocolos de mensaje de los dos lados no son consistentes, se propone la presente invención. Por lo tanto, la presente invención tiene por objeto proporcionar un método mejorado para procesar un formato TLV de datos de comunicación para resolver el problema anterior en las técnicas relacionadas.

En las realizaciones de la presente invención, se proporciona un método para procesar un formato TLV de formato de comunicación de acuerdo con el método de la reivindicación 1 (como se modifica).

Preferentemente, definir el tipo de entero en el formato ASN.1 particularmente comprende: establecer el formato ASN.1 para permitir definir el tipo de entero; y definir una estructura del tipo de entero en un formato de "string type name [X]:: =defining a structure", en el que la X se refiere al tipo de entero.

Preferentemente, definir la estructura del tipo de entero en el formato de "string type name [X]:: =defining a structure" particularmente comprende: establecer la estructura del tipo de entero como un tipo SEQUENCE, SET o CHOICE con el mismo nombre de tipo y especificar que si el miembro del mismo necesita modificarse, únicamente está permitido un aumento neto o una disminución neta en los miembros y únicamente está permitido cambiar un miembro de cola pero no está permitido modificar el tipo.

Preferentemente, el método para procesar el formato TLV comprende además: establecer un compilador de formato ASN.1 con una función de procesamiento del tipo de entero.

Preferentemente, el método para procesar el formato TLV comprende además: obtener, a través de una función, el tipo de entero en un tipo C++ en el formato ASN.1.

Preferentemente, la etapa del lado de envío decodificando el mensaje en el flujo de código comprende: llamar a una función de codificación de valor, una función de codificación de longitud y una función de codificación de tipo de entero mediante una función de codificación de integración, o llamar únicamente a la función de codificación de valor mediante la función de codificación de integración, para codificar el mensaje en el flujo de código; siendo la función de codificación de tipo de entero usada para obtener el tipo de entero del flujo de código de acuerdo la con codificación de datos en la instancia de estructura de datos; siendo la función de codificación de valor usada para codificar datos simples en la instancia de estructura de datos en la V del flujo de código, así como para la instancia de estructura de datos de un tipo de datos simples; siendo la función de codificación de valor usada para llamar secuencialmente a la función de codificación de integración de miembros individuales en la instancia de estructura de datos, así como para la instancia de estructura de datos del tipo SEQUENSE o SET; siendo la función de codificación de valor usada para llamar secuencialmente a la función de codificación de integración de elementos individuales de una matriz en la instancia de estructura de datos, así como para la instancia de estructura de datos del tipo SEQUENSE OF o SET OF; y siendo la función de codificación de valor usada para codificar un miembro seleccionado en la instancia de estructura de datos en la V del flujo de código, así como para la instancia de

estructura de datos del tipo CHOICE, en la que la V comprende el número de secuencia del miembro seleccionado en el CHOICE y el miembro seleccionado.

Preferentemente, cuando el tipo de entero es 0 o está ausente, la función de codificación de integración llama únicamente a la función de codificación de valor; y cuando el tipo de entero no es 0, el flujo de código comprende la V, la longitud L de la V y un tipo de entero.

Preferentemente, el método para procesar el formato TLV comprende además: un lado de recepción que decodifica el flujo de código.

Preferentemente, decodificar el flujo de código particularmente comprende: llamar a una función de decodificación de valor, una función de decodificación de longitud y una función de decodificación de tipo de entero mediante una función de decodificación de integración, o llamar únicamente a la función de decodificación de valor mediante la función de decodificación de integración, para decodificar el flujo de código; siendo la función de decodificación de tipo de entero usada para decodificar el flujo de código de acuerdo con el tipo de entero en el flujo de código; siendo la función de decodificación de valor usada para decodificar la V en el flujo de código en datos simples en la instancia de estructura de datos, así como para la instancia de estructura de datos de un tipo de datos simples; siendo la función de decodificación de valor usada para decodificar secuencialmente la V en el flujo de código en datos de miembros individuales en la instancia de estructura de datos, así como para la instancia de estructura de datos del tipo SEQUENSE o SET; siendo la función de decodificación de valor usada para decodificar secuencialmente la V en el flujo de código en elementos individuales de una matriz en la instancia de estructura de datos, así como para la instancia de estructura de datos del tipo SEQUENSE OF o SET OF, durante la que se analiza si la longitud de un sistema de decodificación en la estructura es más pequeña que L, y si es más pequeña que la L, la decodificación se continúa, de otra manera, la decodificación en la presente estructura se detiene; y siendo la función de decodificación de valor usada para decodificar un número de secuencia en el flujo de código y un miembro que corresponde al número de secuencia, así como para la instancia de estructura de datos del tipo CHOICE, en la que si el miembro que corresponde al número de secuencia no existe en la presente estructura, la función de decodificación de valor se finaliza.

Preferentemente, la etapa de llamar a la función de decodificación de valor, la función de decodificación de longitud y la función de decodificación de tipo de entero mediante la función de decodificación de integración o llamar únicamente a la función de decodificación de valor mediante la función de decodificación de integración para decodificar el flujo de código particularmente comprende: determinar si el tipo de entero es 0 o está ausente mediante la función de decodificación de integración; si no es 0 o no está ausente, llamar a la función de decodificación de valor, la función de decodificación de longitud y la función de decodificación de tipo de entero, en la que la función de decodificación de valor se llama para decodificar el tipo de entero en el flujo de código para verificar si el tipo de entero obtenido es consistente con el mismo tipo de entero, y si no es consistente, se considera como decodificación anormal, la función de decodificación de longitud se llama para decodificar la L en el flujo de código y a continuación la función de decodificación de valor se llama para decodificar la V en el flujo de código; si la longitud vL obtenida decodificando la V es más pequeña que la L, se saltan L-vL bytes; y si es 0 o está ausente, llamar únicamente a la función de decodificación de valor.

Los métodos para procesar el formato TLV en las realizaciones anteriores de la presente invención pueden superar el defecto de que diferentes versiones de protocolos no pueden intercomunicarse debido a la adición de un método para extender la T en el formato ASN.1 para realizar intercomunicación entre diferentes versiones de la ASN.1 y realización de un método de TLV orientado a objetos.

Otras características y ventajas de la presente invención se describen en las partes posteriores de la descripción, y además son parcialmente evidentes a partir de la descripción, o se entienden mediante la implementación de la presente invención. El objetivo y otras ventajas de la presente invención pueden realizarse y obtenerse mediante una estructura particularmente indicada en la descripción, las reivindicaciones y los dibujos.

Breve descripción de los dibujos

Los dibujos ilustrados en este documento proporcionan una comprensión adicional de la presente invención y forman una parte de la presente solicitud. Las realizaciones ilustrativas de la presente invención y la descripción de las mismas se usan para explicar la presente invención sin limitar excesivamente la presente invención, en los que:

la Figura 1 es un diagrama de flujo de un método para procesar un formato TLV de datos de comunicación de acuerdo con una realización de la presente invención;

la Figura 2 es un diagrama esquemático de un proceso de desarrollo en el que un lado de cliente y un servidor comparten un mismo entorno de desarrollo de lenguajes de acuerdo con una realización preferible de la presente invención;

la Figura 3 es una vista de un proceso de desarrollo en diferentes entornos de desarrollo de lenguajes de

acuerdo con una realización preferible de la presente invención;

la Figura 4 es un diagrama esquemático de un proceso de desarrollo en el que un lado de cliente y un servidor usan diferentes entornos de desarrollo de lenguajes y diferentes versiones de protocolos de acuerdo con una realización preferible de la presente invención;

la Figura 5 es un diagrama esquemático de un sistema heredado de tipos de clase base de C++ de ESNACC de acuerdo con una realización preferible de la presente invención; y

la Figura 6 es un diagrama de flujo de una función de decodificación integral de TLV de acuerdo con una realización preferible de la presente invención.

Descripción detallada de las realizaciones preferibles

Sumario de función

En consideración del problema presente en la técnica relacionada que el Cliente y el Servidor fallan en comunicarse entre sí cuando las versiones de los protocolos de mensaje de los dos lados no son consistentes, se proporciona un método para procesar un formato TLV de datos de comunicación mediante las realizaciones de la presente invención. En la presente realización, un método de extensión de la T se añade en el formato ASN.1 y se rellena un mensaje en el formato ASN.1 extendido, superando de este modo el defecto que diferentes versiones de protocolos no pueden comunicarse y realizando intercomunicación entre diferentes versiones de la ASN.1 y un método TLV orientado a objetos.

La presente invención se describirá en detalle en lo sucesivo con referencia a los dibujos y en combinación con las realizaciones. Si no surge ningún conflicto, las realizaciones de la presente invención y las características de las realizaciones pueden combinarse entre sí. Lo que necesita explicarse es que, las etapas mostradas en los diagramas de flujo de los dibujos pueden implementarse en un sistema informático tales como un conjunto de instrucciones ejecutables por ordenador. Además, aunque en los diagramas de flujo se muestra un orden lógico, las etapas mostradas o descritas pueden implementarse en el orden diferentes en algunas ciertas situaciones.

La Figura 1 es un diagrama de flujo de un método para procesar un formato TLV de datos de comunicación de acuerdo con una realización de la presente invención, incluyendo las siguientes etapas:

etapa S10, definir un tipo T de un tipo de entero en un formato ASN.1;

etapa S20, un lado de envío rellenando en un mensaje en el formato ASN.1 incluyendo el mensaje una instancia de estructura de datos; y

etapa S30, el lado de envío decodificando el mensaje en un flujo de código, en el que el flujo de código de la instancia de estructura de datos en el mensaje incluye un valor V, o incluye la V, el tipo de entero de la V y una longitud L de la V.

El método para procesar el formato TLV supera el defecto de que diferentes versiones de protocolos no pueden intercomunicarse y realiza intercomunicación entre diferentes versiones de la ASN.1 y un método TLV orientado a objetos ya que el método de extensión de la T se usa en el formato ASN.1.

El anterior proceso de procesamiento se describe en detalle en lo sucesivo.

(I) Etapa S10

La operación de definir el tipo de entero en el formato ASN.1 puede comprender específicamente: establecer el formato ASN.1 para permitir definir el tipo de entero; y definir una estructura del tipo de entero en un formato de "string type name [X] :: =defining a structure", en el que la X se refiere al tipo de entero. En concreto, se añade una sintaxis extendida a la sintaxis ASN.1 para soportar un tipo X autodefinido (a saber, la T en el TLV), que se define de una manera opcional entre el nombre de tipo y el símbolo ":: =". Si no se proporciona X, X se considera como 0, por lo tanto, tales datos únicamente tienen V pero no X y L en un flujo de código correspondiente.

Específicamente, la operación de definir la estructura del tipo de entero en el formato de "string type name [X]:: =defining a structure" particularmente comprende: establecer la estructura del tipo de entero como un tipo de SEQUENCE, SET o CHOICE con el mismo nombre de tipo, y especificar que si el miembro del mismo necesita modificarse, únicamente está permitido un aumento neto o una disminución neta en los miembros y únicamente está permitido cambiar un miembro de cola pero no está permitido modificar el tipo. Siguiendo las reglas anteriores, puede realizarse la compatibilidad de un mensaje de protocolo nuevo con un mensaje de protocolo antiguo. Para facilitar la gestión del tipo de mensaje de protocolo, puede requerirse que una versión del protocolo pueda únicamente extenderse sobre la base del tipo de una antigua versión del protocolo y los miembros extendidos

únicamente pueden estar en la cola de un tipo progenitor, pero no puede reducirse.

En la etapa S10, ya que el formato ASN.1 se establece para permitir definir el tipo de entero, de tal forma que el compilador de formato ASN.1 puede proporcionarse con una función de procesamiento del tipo de entero. Preferentemente, el tipo de entero puede obtenerse a través de una función en un tipo C++ en el formato ASN.1.

(II) Etapa S20

La operación del lado de envío decodificando el mensaje en el flujo de código específicamente comprende: llamar a una función de codificación de valor, una función de codificación de longitud y una función de codificación de tipo de entero mediante una función de codificación de integración, o llamar únicamente a la función de codificación de valor mediante la función de codificación de integración, para codificar el mensaje en el flujo de código. La función de codificación de tipo de entero se usa para obtener el tipo de entero del flujo de código de acuerdo la con codificación de datos en la instancia de estructura de datos. Específicamente, la función de codificación de valor se usa para codificar datos simples en la instancia de estructura de datos en la V del flujo de código, así como para la instancia de estructura de datos de un tipo de datos simples. La función de codificación de tipo de valor se usa para llamar secuencialmente a la función de codificación de integración de miembros individuales en la instancia de estructura de datos, así como para la instancia de estructura de datos del tipo SEQUENSE o SET. La función de codificación de tipo de valor se usa para llamar secuencialmente a la función de codificación de integración de elementos individuales de una matriz en la instancia de estructura de datos, así como para la instancia de estructura de datos del tipo SEQUENSE OF o SET OF. La función de codificación de tipo de valor se usa para codificar un miembro seleccionado en la instancia de estructura de datos en la V del flujo de código, así como para la instancia de estructura de datos del tipo CHOICE, en la que la V comprende el número de secuencia del miembro seleccionado en el CHOICE y el miembro seleccionado.

Preferentemente, cuando el tipo de entero es 0 o está ausente, la función de codificación de integración llama únicamente a la función de codificación de valor, a saber, el flujo de código únicamente comprende V o comprende V y el tipo de entero = 0. Cuando el tipo de entero no es 0, la función de codificación de integración llama a la función de codificación de valor, la función de codificación de longitud y la función de codificación de tipo de entero, a saber, el flujo de código comprende la V, la longitud L de la V y un tipo de entero.

El lado de envío que procesa el formato TLV se describe como anteriormente, así como para un lado de recepción, el método para procesar el formato TLV comprende además: el lado de recepción decodificando el flujo de código.

Preferentemente, la operación de decodificar el flujo de código particularmente incluye:

llamar a una función de decodificación de valor, una función de decodificación de longitud y una función de decodificación de tipo de entero mediante una función de decodificación de integración, o llamar únicamente a la función de decodificación de valor mediante la función de decodificación de integración, para decodificar el flujo de código; usar la función de decodificación de tipo de entero para decodificar el flujo de código de acuerdo con el tipo de entero en el flujo de código; usar la función de decodificación de valor para decodificar la V en el flujo de código en datos simples en la instancia de estructura de datos, así como para la instancia de estructura de datos de un tipo de datos simples; usar la función de decodificación de valor para decodificar secuencialmente la V en el flujo de código en datos de miembros individuales en la instancia de estructura de datos, así como para la instancia de estructura de datos del tipo SEQUENSE o SET; usar la función de decodificación de valor para decodificar secuencialmente la V en el flujo de código en elementos individuales de una matriz en la instancia de estructura de datos, así como para la instancia de estructura de datos del tipo SEQUENSE OF o SET OF, durante la que se analiza si la longitud de un sistema de decodificación en la estructura es más pequeña que L, y si es más pequeña que la L, la decodificación se continúa, de otra manera, la decodificación en la presente estructura se detiene; y usar la función de decodificación de valor para decodificar un número de secuencia en el flujo de código y un miembro que corresponde al número de secuencia, así como para la instancia de estructura de datos del tipo CHOICE, en la que si el miembro que corresponde al número de secuencia no existe en la presente estructura, la función de decodificación de valor se finaliza.

Preferentemente, la operación de llamar a la función de decodificación de valor, la función de decodificación de longitud y la función de decodificación de tipo de entero mediante la función de decodificación de integración o llamar únicamente a la función de decodificación de valor mediante la función de decodificación de integración para decodificar el flujo de código particularmente comprende: determinar si el tipo de entero es 0 o está ausente mediante la función de decodificación de integración en la que si no es 0 o no está ausente, llama a la función de decodificación de valor, la función de decodificación de longitud y la función de decodificación de tipo de entero, en la que la función de decodificación de valor se llama para decodificar el tipo de entero en el flujo de código para verificar si el tipo de entero obtenido es consistente con su propio tipo de entero, y si no es consistente, se considera como decodificación anormal, a continuación la función de decodificación de longitud se llama para decodificar la L en el flujo de código y a continuación la función de decodificación de valor se llama para decodificar la V en el flujo de código, y si la longitud vL obtenida decodificando la V es más pequeña que la L, se saltan L-vL bytes; y si es 0 o está ausente, llama únicamente a la función de decodificación de valor.

La función de decodificación de integración primero determina si su propia T es 0. Si no es 0, la T se decodifica y se verifica si la T decodificada es consistente con su T, en la que si no es consistente, se considera como decodificación anormal, a continuación se decodifica L, y a continuación se decodifica V, en la que si la longitud vL de la V decodificada es más pequeña que L, se saltan L-vL bytes para equilibrar la longitud del tipo presente; si es 0, se decodifica V directamente.

Funciones virtuales requeridas por las funciones de codificación y funciones de decodificación anteriores se añaden a las clases base de ASN.1 en un módulo básico ASN.1 C++ y las clases y las funciones de los miembros de las mismas requeridas por las funciones de codificación y funciones de decodificación anteriores se añaden a un compilador ASN.1 C++.

La realización preferible anterior incluye dos módulos funcionales: un módulo básico y un compilador. El módulo base realiza soporte de las clases base de TLV y algunas funciones de soporte. El compilador realiza análisis de un fichero ASN.1 y genera clases que corresponden a diversos tipos de ASN.1 y funciones de miembros de las clases. En un programa de aplicación, el mensaje definido mediante un protocolo TLV de este tipo se usa no únicamente para realizar interacción con mensajes de protocolo de la misma versión, sino también para realizar interacción entre el Cliente con el Servidor con los nuevos y antiguos mensajes de protocolo para proporcionar garantía de las reglas de codificación para la interacción C/S de los protocolos de versiones diferentes. Las realizaciones de la presente invención pueden realizarse o bien independientemente o extendidamente en cualquier plataforma de compilador ASN.1 existente.

La Figura 2 muestra un procedimiento de desarrollo en el que el Cliente y el Servidor comparten un mismo entorno de desarrollo de lenguajes de acuerdo con una realización preferible de la presente invención, en el que, tanto el Cliente como el Servidor se desarrollan usando C++. El procedimiento de desarrollo incluye: A. un ingeniero de especificación de interfaz definiendo un fichero de interfaz de ASN.1 y generando un fichero.h de interfaz y un fichero.cpp de interfaz usando un compilador ASN.1 C++; B. un desarrollador de servidor escribiendo un código de servidor y compilando el mismo con el fichero.h de interfaz y el fichero.cpp de interfaz en un programa de servidor ejecutable; C. un desarrollador de Cliente escribiendo un código de Cliente y compilando el mismo con el fichero.h de interfaz y el fichero.cpp de interfaz en un programa de cliente ejecutable; y D. el programa de cliente ejecutable y el programa de servidor ejecutable, cuando se están ejecutando, interactuando y comunicando usando el módulo básico ASN.1 C++.

La Figura 3 muestra un procedimiento de desarrollo en diferentes entornos de desarrollo de lenguajes de acuerdo con una realización preferible de la presente invención, en el que el Cliente usa java para desarrollar y el Servidor usa C++ para desarrollar. El procedimiento de desarrollo incluye: A. un ingeniero de especificación de interfaz definiendo un fichero de interfaz de ASN.1, generando un fichero.h de interfaz y un fichero.cpp de interfaz usando un compilador ASN.1 C++ para un desarrollador C++, y generando un fichero.java de interfaz usando un compilador java ASN.1 para un desarrollador java; B. un desarrollador de servidor escribiendo un código de servidor y compilando el mismo con el fichero.h de interfaz y el fichero.cpp de interfaz en un programa de servidor ejecutable; C. un desarrollador de Cliente escribiendo un código de Cliente y compilando el mismo con el fichero.java de interfaz en un programa de cliente ejecutable; y D. el programa de cliente ejecutable y el programa de servidor ejecutable, cuando se están ejecutando, interactuando y comunicando usando los módulos básicos ASN.1 de su respectivo lenguaje.

La Figura 4 muestra un procedimiento de desarrollo en el que el Cliente y el Servidor usan diferentes entornos de desarrollo de lenguajes y diferentes versiones de protocolos de acuerdo con una realización preferible de la presente invención, en el que el Servidor usa una versión antigua, el Cliente desarrolla con java y el Servidor desarrolla con C++, que es procedimiento de desarrollo relativamente complejo. En las etapas de realización, tanto el Cliente como el Servidor desarrollan con C++. El procedimiento incluye:

A. un ingeniero de especificación de interfaz definiendo un fichero versión 2.asn de interfaz cuando desarrolla una nueva versión basándose en un fichero versión 1.asn de interfaz definido para la versión antigua, y compilando dos versiones respectivamente para generar ficheros de realización de interfaz C++ y Java;

B. un desarrollador de servidor escribiendo un código de servidor en la versión antigua y compilando el mismo con un fichero versión 1.h de interfaz y un fichero versión 1.cpp de interfaz en un fichero de ejecución de servidor ejecutable;

C. un desarrollador de Cliente escribiendo un código de Cliente en una nueva versión y compilando el mismo con el fichero 2.java de interfaz en un fichero de ejecución de cliente ejecutable; y

D. el programa de cliente ejecutable y el programa de servidor ejecutable, cuando se están ejecutando, interactuando y comunicando usando los módulo básicos ASN.1 de su respectivo lenguaje.

La Figura 5 muestra un sistema heredado de tipos de clase base de C++ de ESNACC de acuerdo con una realización preferible de la presente invención, diagrama jerárquico de relación de herencia de clase base de

ESNACC, en el que la ASNType es la clase base de máximo nivel. AsnType es una clase base ABSTRACT. Sobre esta base, algunos tipos simples se heredan para soportar tipos de datos ASN.1 base tales como AsnInt, AsnReal, AsnRelativeOid, AsnAny, AsnOcts, AsnBool, AsnNull y asnString, en el que AsnInt se obtiene a partir de PERGgeneral, AsnEnum se obtiene a partir de AsnInt; AsnOid se obtiene a partir de AsnRelativeOid; AsnList es un tipo intermedio para realizar otros tipos de ASN.1; AsnSetOf y AsnSeqOf se obtienen a partir de AsnList; los std::string y std::list de una librería de plantillas normalizada se usan para soportar adicionalmente SEQUENCE OF, SET OF, etc. Algunos tipos de cadena aplicados tales como VisibleString, GraphicString, Ia5String, PrintableString, NumericString se obtienen a partir de un tipo de cadena básico AsnString, en el que GeneralizedTime y UTCTime se obtienen a partir de VisibleString.

La Figura 6 es un diagrama de flujo de una función de decodificación integral de TLV de acuerdo con una realización preferible de la presente invención.

En lo sucesivo, de acuerdo con demanda de una entidad de gestión de red, se realizan reglas de codificación y decodificación de mensajes TLV basándose en ESNACC, que es plataforma de compilación de código abierto de ASN.1. La presente invención se realiza usando cualquier lenguaje de programación orientado a objetos. La presente solicitud se describe principalmente en conjunción con C++. También puede usarse otro lenguaje de programación orientado a objetos, tal como lenguaje JAVA.

(I) Fuente de la demanda

así como para entidades de gestión de red, una entidad de gestión de red a menudo necesita gestionar diversos dispositivos de elementos de red. Diferentes dispositivos tienen diferentes versiones, y la propia entidad de gestión de red también tiene diferentes versiones. Las entidades de gestión de red y el software de dispositivo necesitan interacción de mensajes. Para debilitar la dependencia de versión de las entidades de gestión de red y el software de dispositivo, las entidades de gestión de red necesitan gestionar dispositivos de elementos de red de una manera compatible con la versión, de este modo se necesita una regla de protocolo para cumplir la demanda.

(II) Plataforma de compilación

El ESNACC es un compilador de código abierto de ASN.1 y soporta codificación BER y codificación PER. Un módulo base ASN.1 proporcionado por el compilador. El compilador ASN.1 generalmente proporciona un compilador y un módulo de soporte básico y el compilador compila un fichero ASN.1 para generar un código de lenguaje de programación, tales como C++ y java. Los tipos de la ASN.1 se compilan en las clases de un lenguaje de programación. El módulo de soporte base proporciona soporte para la ejecución de los códigos generados. Las reglas de codificación y decodificación de TLV se realizan modificando el compilador y el módulo de soporte básico.

(III) Realización de reglas de codificación de TLV específicas

En un procedimiento de realización, se requiere que el flujo de código se depure y se requiere que se lea el flujo de código TLV, por lo tanto, T usa dos bytes y L usa cuatro bytes, y el formato es de orden de red. Pueden usarse otros formatos para almacenar T y L en la práctica, tales como el formato de representación en la BER con respecto a la longitud.

1. La sintaxis se extiende en la ASN.1 para soportar una autodefinida T, por ejemplo:

```
QxString32 [10]::= GraphicString (32)
NameAndStringValue-E [30] ::= SEQUENCE
{
    name QxString32
    ,value QxString32
}
```

T se rellena en los corchetes cuadrados antes de "::=", y puede representarse en diversos formatos, tales como decimal y hexadecimal. La sintaxis es un artículo opcional, por lo tanto, la situación es imposible que T se representa como 0.

2. Los mensajes definidos de las siguientes formas pueden intercomunicar entre una versión antigua y una nueva versión.

(1) Definir la versión antigua

```
ANode
[0xF001]::= SEQUENCE
{
    a1 QxUInt8,
```

```
        a2 QxUInt32,
    }
    BNode [0xF003]::= SEQUENCE
    {
        b1 QxUInt32,
        b2 QxUInt16,
        c3 QxUInt32
    }
    CNode [0xF005]::= SEQUENCE
    {
        a ANode,
        b BNode
    }
```

(2) Definir la versión nueva

```
    ANode [0xF001]::= SEQUENCE
    {
        a1 QxUInt8,
        a2 QxUInt32,
        d3 QxUInt32,
        d4 QxUInt32
    }
    BNode [0xF003]::= SEQUENCE
    {
        b1 QxUInt32,
        b2 QxUInt16,
    }
    CNode [0xF005]::= SEQUENCE
    {
        a ANode,
        b BNode
    }
```

En las versiones nuevas y antiguas, miembros d3 y d4 se añaden en la cola del ANode de la nueva versión, mientras un miembro c3 se borra en el BNode. Ellos y CNode pueden lograr una compatibilidad de versión e interacción con el método de la presente invención. Desde el flujo de código de mensaje de versión antigua hasta el mensaje de nueva versión, d3 y d4 usan valores por defecto, mientras el flujo de código de c3 se omite. Desde el flujo de código de mensaje de versión antigua hasta el mensaje de nueva versión, el flujo de códigos de d3 y d4 se omiten, mientras el valor de c3 es un valor por defecto. Otros datos son consistentes entre las versiones nuevas y antiguas.

3. El fichero de sintaxis ASN.1 del ESNACC se modifica para soportar que el compilador analice y explore la T. Durante la realización, la T se sitúa en la información de descripción de la clase META de las clases.

4. Funciones de codificación y decodificación se añaden a la clase base básica del ESNACC, que son principalmente una función de codificación de integración, una función de codificación de valor, una función de decodificación de integración y una función de codificación de valor. Los prototipos de las funciones pueden ser como se indica a continuación:

la función de codificación de integración: virtual AsnLen TlvEnc (AsnBuf &_b);

la función de codificación de V: virtual AsnLen TlvEncContent (AsnBuf &_b);

la función de decodificación de integración: virtual AsnLen TlvDec (AsnBuf &_b, AsnLen&bytesDecoded); y

la función de decodificación de valor: virtual AsnLen TlvDecContent (AsnBuf &_b, short iT, AsnLen iL, AsnLen & bytesDecoded);

En lo anterior, b es un parámetro de flujo de código, byteDecoded es el número acumulado de bytes de un flujo de código de mensaje decodificado, iT es T del flujo de código, iL es L del flujo de código y valores de retorno TlvEnc y TlvEncContent son longitudes de flujos de código de codificación. Valores de retorno TlvDec y TlvDecContext son los números de bytes decodificados en las funciones. Basándose en la manera de realización, el prototipo de las funciones varía según se requiera.

5. El compilador se modifica en el ESNACC para realizar códigos de generación de diversas funciones de codificación y decodificación, que principalmente son una función de codificación de valor y una función de decodificación de valor. Ya que las funciones de codificación y decodificación de integración pueden ser universales para diversos tipos, los procedimientos de codificación y decodificación se describieron haciendo referencia a la Figura 5 y en el anterior proceso de operación de codificación y decodificación del flujo de código.

6. De acuerdo con la descripción detallada de la anterior etapa S10, se hace un fichero.asn de interfaz de versión nueva/antigua y se generan un mensaje.h de interfaz y un mensaje.cpp de interfaz, y tanto el Cliente como el Servidor usan C++ para editar. Por lo tanto, de manera similar a como se muestra en la Figura 3, puede realizarse interacción y comunicación entre diferentes versiones del cliente y el Servidor.

Se apreciará que, para el experto en la materia, diversas posibles modificaciones o sustituciones pueden hacerse de acuerdo con la descripción de la solución técnica y las realizaciones específicas de la presente invención, tales como la posición y el formato de la T, el formato de la L, los formatos de compresión y encriptación posiblemente usados y la funciones para realizar la presente invención podrían ser diferentes, y además podría realizarse incluso usando la manera de operador de codificación en lugar de las funciones de miembro.

A partir de la anterior descripción, puede observarse que, la presente invención realiza los siguientes efectos técnicos:

1. convertir un tipo en un flujo de código mediante un método de codificación;

2. decodificar el flujo de código en un ejemplo de mediante un método de decodificación;

3. intercomunicar entre protocolos de versiones diferentes;

4. autodefinir un tipo T;

5. intercomunicar entre programas de diferentes lenguajes; y

6. realizar capacidad de procesamiento alta abstractamente debido al uso de un mecanismo de función virtual de un lenguaje orientado a objetos.

El presente método para procesar un propio formato TLV es universal y no depende un compilador ASN.1 específico y un lenguaje orientado a objetos específico. Incluso en un lenguaje estructurado, puede realizarse tal codificación y decodificación de TLV.

Obviamente, el experto en la materia apreciará que, los anteriores módulos respectivos o etapas respectivas de la presente invención pueden realizarse mediante un dispositivo informático universal. Pueden integrarse en un único dispositivo informático o distribuirse por una red compuesta de múltiples dispositivos informáticos. Opcionalmente, pueden realizarse usando un código de programa ejecutable mediante el dispositivo informático, por lo tanto, pueden almacenarse en un dispositivo de almacenamiento para ejecutarse mediante el dispositivo informático o fabricarse en respectivos módulos de circuito integrado o sus múltiples módulos o etapas se fabrican en un único módulo de circuito integrado a realizar. Por lo tanto, la presente invención no se limita a ninguna combinación específica de hardware y software.

Las descripciones anteriores son únicamente realizaciones preferibles de la presente invención, que no se usan para restringir la presente invención. La presente invención puede tener diversos modificaciones y cambios. Cualquier modificación, sustituciones equivalentes, mejoras se incluyen todas en el alcance de protección de la presente invención.

REIVINDICACIONES

1. Un método para procesar un formato Tipo Longitud Valor, TLV, de datos de comunicación, **caracterizado por** comprender las siguientes etapas:

definir un tipo, T, de un tipo de entero en un formato de Notación de Sintaxis Abstracta 1, ASN.1;
un lado de envío rellenando un mensaje en el formato ASN.1, incluyendo el mensaje una instancia de estructura de datos; y
el lado de envío decodificando el mensaje en un flujo de código, en donde el flujo de código de la instancia de estructura de datos en el mensaje incluye un valor, V, o incluye la V, el tipo de entero de la V y una longitud L de la V;
en donde, cuando el tipo de entero es 0 o está ausente, el flujo de código únicamente comprende V, y cuando el tipo de entero no es 0, el flujo de código comprende la V, la longitud L de la V y el tipo de entero.

2. El método para procesar el formato TLV de acuerdo con la reivindicación 1, **caracterizado por que**, al definir el tipo de entero en el formato ASN.1, particularmente comprende:

establecer el formato ASN.1 para permitir definir el tipo de entero; y
definir una estructura del tipo de entero en un formato de "string type name [X]:: =defining a structure", en donde la X se refiere al tipo de entero.

3. El método para procesar el formato TLV de acuerdo con la reivindicación 2, **caracterizado por que**, al definir la estructura del tipo de entero en el formato de "string type name [X]:: =defining a structure", particularmente comprende:

establecer la estructura del tipo de entero como un tipo de SEQUENCE, SET o CHOICE con el mismo nombre de tipo y especificar que, si el miembro del mismo necesita modificarse, únicamente está permitido un aumento neto o una disminución neta en los miembros y únicamente está permitido cambiar un miembro de cola, pero no está permitido modificar el tipo.

4. El método para procesar el formato TLV de acuerdo con la reivindicación 3, **caracterizado por** que además comprende:

establecer un compilador de formato ASN.1 con una función de procesamiento del tipo de entero.

5. El método para procesar el formato TLV de acuerdo con la reivindicación 4, **caracterizado por que** además comprende:

obtener, a través de una función, el tipo de entero en un tipo C++ en el formato ASN.1.

6. El método para procesar el formato TLV de acuerdo con la reivindicación 1, **caracterizado por que** la etapa del lado de envío decodificando el mensaje en el flujo de código comprende:

llamar a una función de codificación de valor, a una función de codificación de longitud y a una función de codificación de tipo de entero, mediante una función de codificación de integración, o llamar únicamente a la función de codificación de valor mediante la función de codificación de integración para codificar el mensaje en el flujo de código;
siendo la función de codificación de tipo de entero usada para obtener el tipo de entero del flujo de código de acuerdo con la codificación de datos en la instancia de estructura de datos;
siendo la función de codificación de valor usada para codificar datos simples en la instancia de estructura de datos en la V del flujo de código, así como para la instancia de estructura de datos de un tipo de datos simples;
siendo la función de codificación de valor usada para llamar secuencialmente a la función de codificación de integración de miembros individuales en la instancia de estructura de datos, así como para la instancia de estructura de datos del tipo SEQUENSE o SET;
siendo la función de codificación de valor usada para llamar secuencialmente a la función de codificación de integración de elementos individuales de una matriz en la instancia de estructura de datos, así como para la instancia de estructura de datos del tipo SEQUENSE OF o SET OF; y
siendo la función de codificación de valor usada para codificar un miembro seleccionado en la instancia de estructura de datos en la V del flujo de código, así como para la instancia de estructura de datos del tipo CHOICE, en la que la V comprende el número de secuencia del miembro seleccionado en el CHOICE y el miembro seleccionado.

7. El método para procesar el formato TLV de acuerdo con la reivindicación 6, **caracterizado por que**, cuando el tipo de entero es 0 o está ausente, la función de codificación de integración llama únicamente a la función de codificación de valor; y cuando el tipo de entero no es 0, el flujo de código comprende la V, la longitud L de la V y un tipo de entero.

8. El método para procesar el formato TLV de acuerdo con la reivindicación 1, **caracterizado por que** además comprende: un lado de recepción, decodificando el flujo de código.

9. El método para procesar el formato TLV de acuerdo con la reivindicación 8, **caracterizado por que** decodificar el flujo de código particularmente comprende:

llamar a una función de decodificación de valor, a una función de decodificación de longitud y a una función de decodificación de tipo de entero mediante una función de decodificación de integración o llamar únicamente a la función de decodificación de valor mediante la función de decodificación de integración para decodificar el flujo de código;
siendo la función de decodificación de tipo de entero usada para decodificar el flujo de código de acuerdo con el tipo de entero en el flujo de código;
siendo la función de decodificación de valor usada para decodificar la V en el flujo de código en datos simples en la instancia de estructura de datos, así como para la instancia de estructura de datos de un tipo de datos simples;
siendo la función de decodificación de valor usada para decodificar secuencialmente la V en el flujo de código en datos de miembros individuales en la instancia de estructura de datos, así como para la instancia de estructura de datos del tipo SEQUENSE o SET;
siendo la función de decodificación de valor usada para decodificar secuencialmente la V en el flujo de código en elementos individuales de una matriz en la instancia de estructura de datos, así como para la instancia de estructura de datos del tipo SEQUENSE OF o SET OF, durante la que se analiza si la longitud de un flujo de decodificación en la estructura es más pequeña que L, y si es más pequeña que la L, la decodificación se continúa, de otra manera, la decodificación en la presente estructura se detiene; y
siendo la función de decodificación de valor usada para decodificar un número de secuencia en el flujo de código y un miembro que corresponde al número de secuencia, así como para la instancia de estructura de datos del tipo CHOICE, en donde si el miembro que corresponde al número de secuencia no existe en la presente estructura, la función de decodificación de valor se finaliza.

10. El método para procesar el formato TLV de acuerdo con la reivindicación 9, **caracterizado por que**, la etapa de llamar a la función de decodificación de valor, a la función de decodificación de longitud y a la función de decodificación de tipo de entero mediante la función de decodificación de integración o llamar únicamente a la función de decodificación de valor mediante la función de decodificación de integración para decodificar el flujo de código particularmente comprende:

determinar si el tipo de entero es 0 o está ausente mediante la función de decodificación de integración; si no es 0 o no está ausente, llamar a la función de decodificación de valor, a la función de decodificación de longitud y a la función de decodificación de tipo de entero, en donde se llama a la función de decodificación de valor para decodificar el tipo de entero en el flujo de código para verificar si el tipo de entero obtenido es consistente con el mismo tipo de entero, y si no es consistente, se considera como decodificación anormal, se llama a la función de decodificación de longitud para decodificar la L en el flujo de código y a continuación se llama a la función de decodificación de valor para decodificar la V en el flujo de código; si la longitud vL obtenida decodificando la V es más pequeña que la L, se saltan L-vL bytes; y
si es 0 o está ausente, llamar únicamente a la función de decodificación de valor.

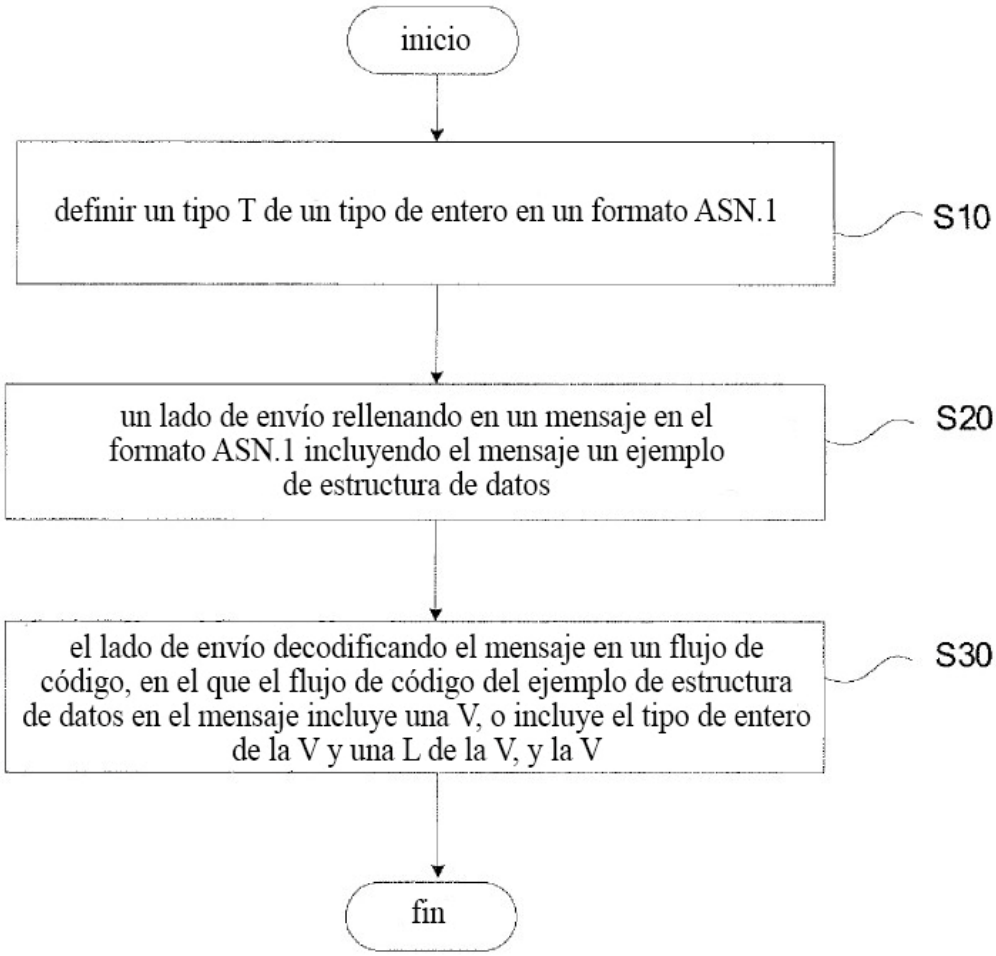
inicio
definir un tipo T de un tipo de entero en un formato ASN.1
S10
un lado de envío rellenando en un mensaje en el formato ASN.1 incluyendo el mensaje un ejemplo de estructura de datos
S20
el lado de envío decodificando el mensaje en un flujo de código, en el que el flujo de código del ejemplo de estructura de datos en el mensaje incluye una V, o incluye el tipo de entero de la V y una L de la V, y la V
S30
fin

Fig. 1

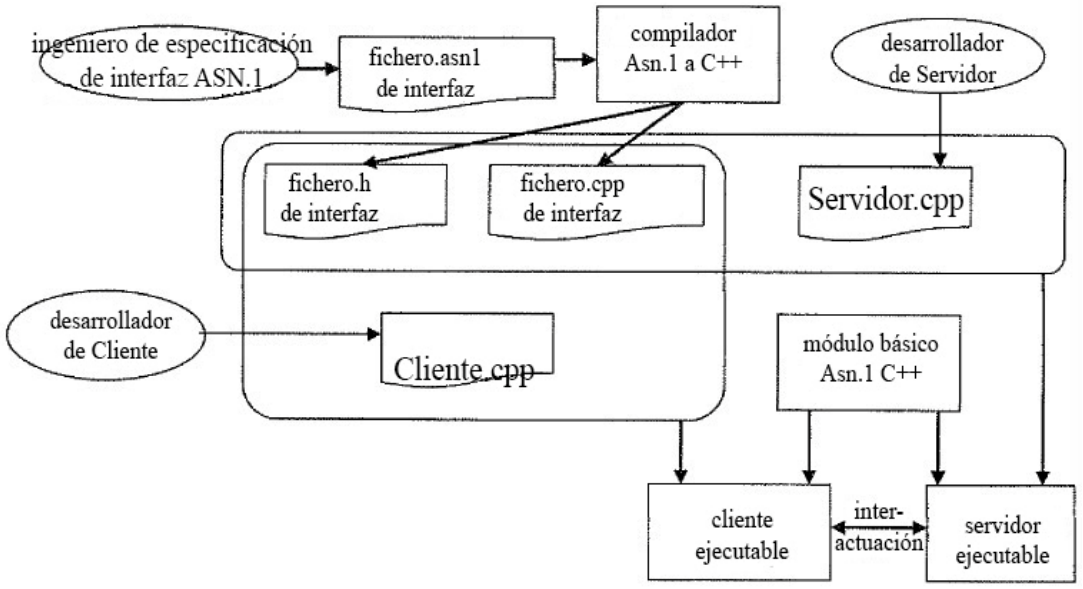
ingeniero de especificación de interfaz ASN.1
fichero.asn1 de interfaz
compilador Asn.1 a C++
desarrollador de Servidor
fichero.h de interfaz
fichero.cpp de interfaz
Servidor.cpp
desarrollador de Cliente
Cliente.cpp
módulo básico Asn.1 C++
cliente ejecutable
inter-actuación
servidor ejecutable

Fig. 2

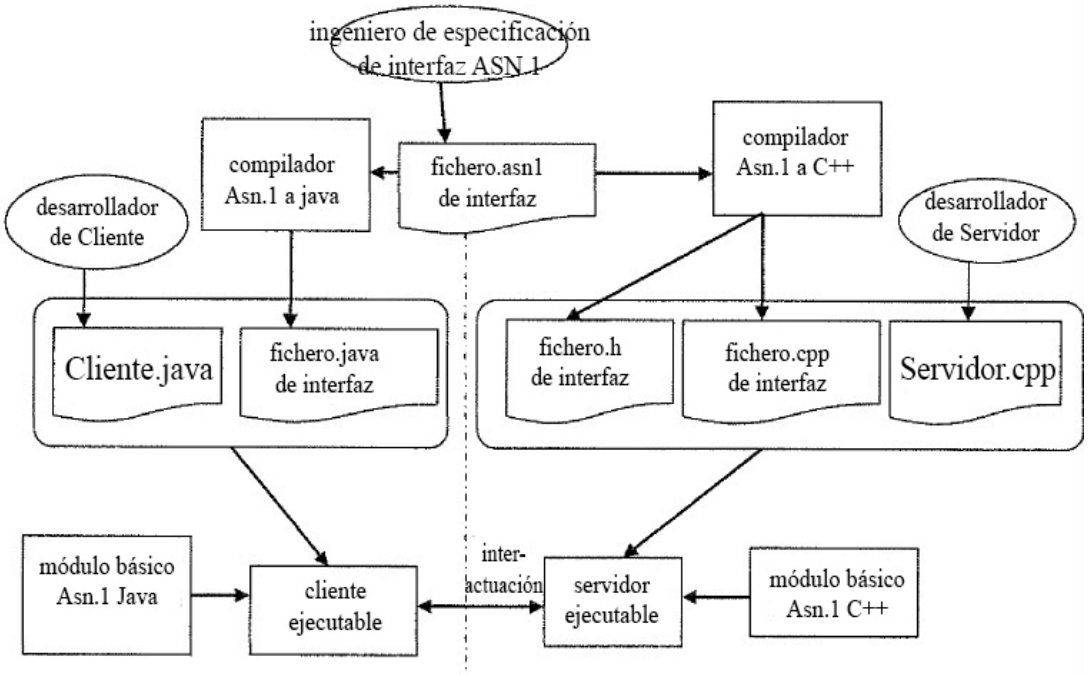
ingeniero de especificación de interfaz ASN.1
compilador Asn.1 a java
fichero.asn1 de interfaz
compilador Asn.1 a C++
desarrollador de Cliente
desarrollador de Servidor
Cliente.java
fichero.java de interfaz
fichero.h de interfaz
fichero.cpp de interfaz
Servidor.cpp
módulo básico Asn.1 Java
cliente ejecutable
inter-actuación
servidor ejecutable
módulo básico Asn.1 C++

Fig. 3

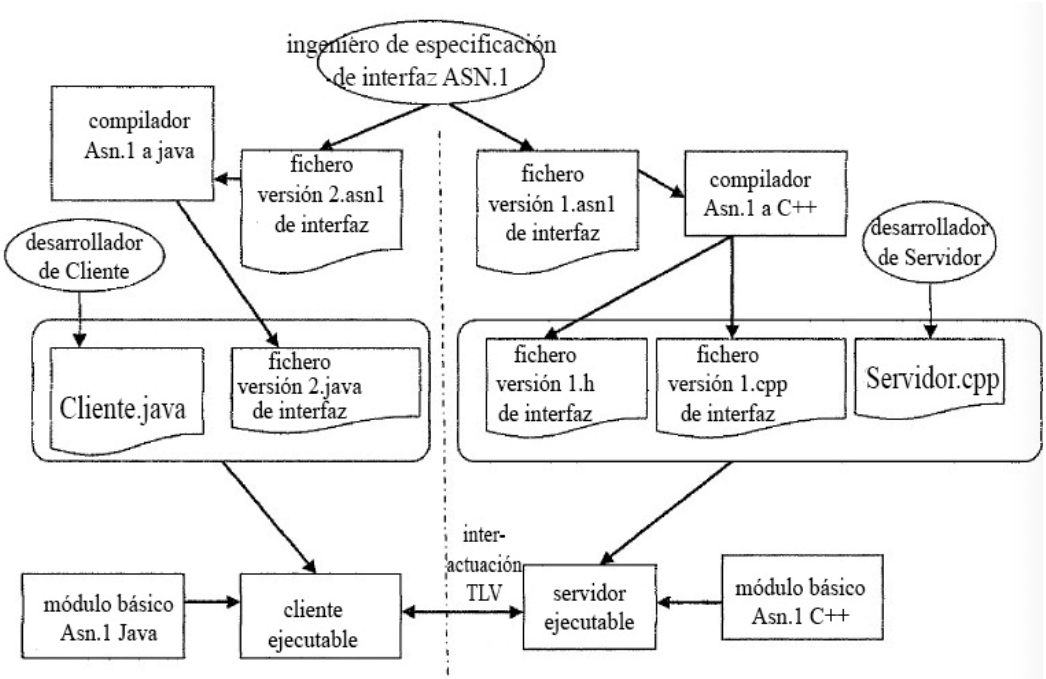
ingeniero de especificación de interfaz ASN.1
compilador Asn.1 a java
fichero versión 2.asn1 de interfaz
fichero versión 1.asn1 de interfaz
compilador Asn.1 a C++
desarrollador de Cliente
desarrollador de Servidor
Cliente.java
fichero versión 2.java de interfaz
fichero versión 1.h de interfaz
fichero versión 1.cpp de interfaz
Servidor.cpp
inter-actuación TLV
módulo básico Asn.1 Java
cliente ejecutable
servidor ejecutable
módulo básico Asn.1 C++

Fig. 4

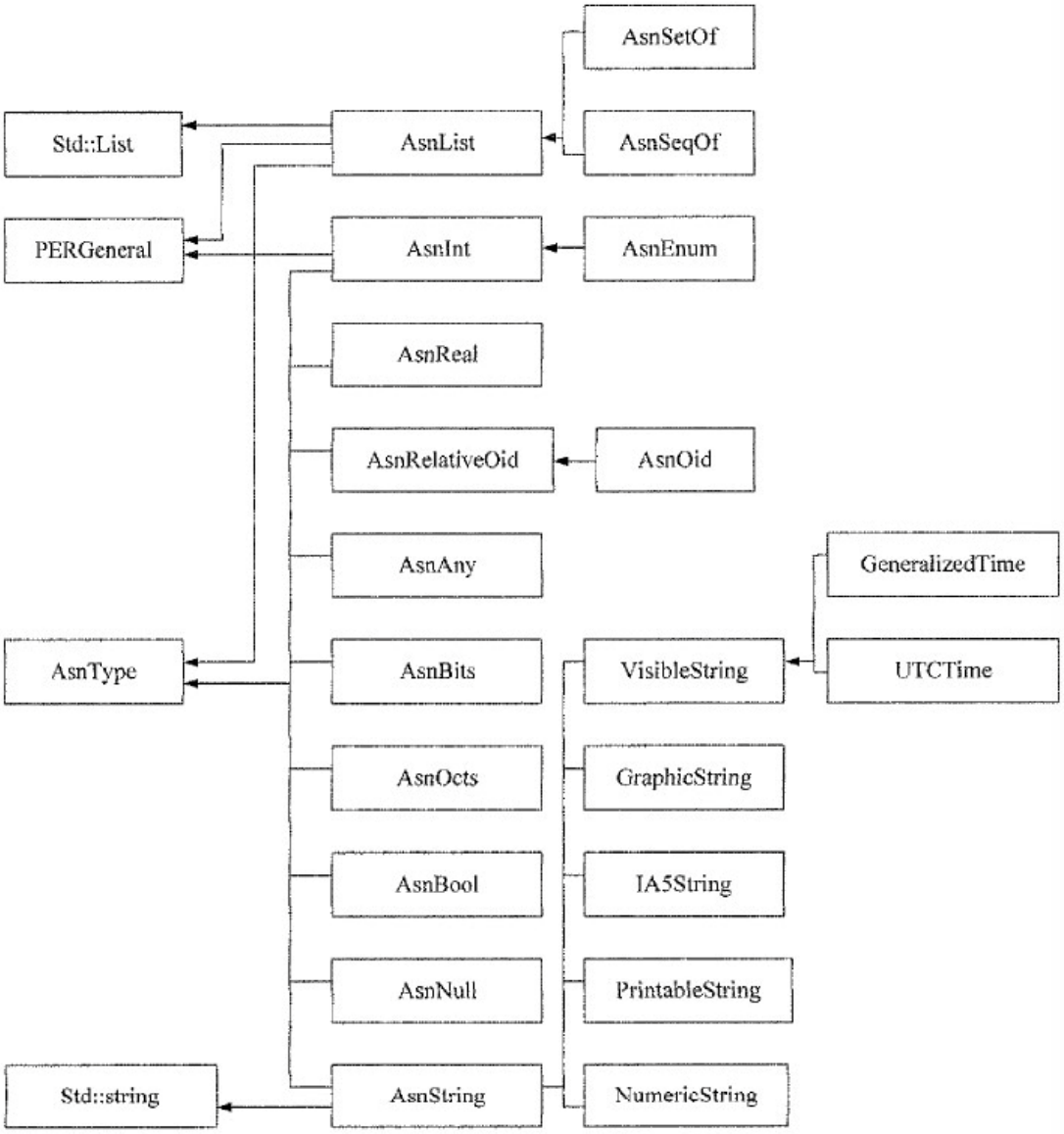
AsnSetOf
Std::List
AsnList
AsnSeqOf
PERGeneral
AsnInt
AsnEnum
AsnReal
AsnRelativeOid
AsnOid
AsnAny
GeneralizedTime
AsnType
AsnBits
VisibleString
UTCTime
AsnOcts
GraphicString
AsnBool
IA5String
AsnNull
PrintableString
Std::string
AsnString
NumericString

Fig. 5

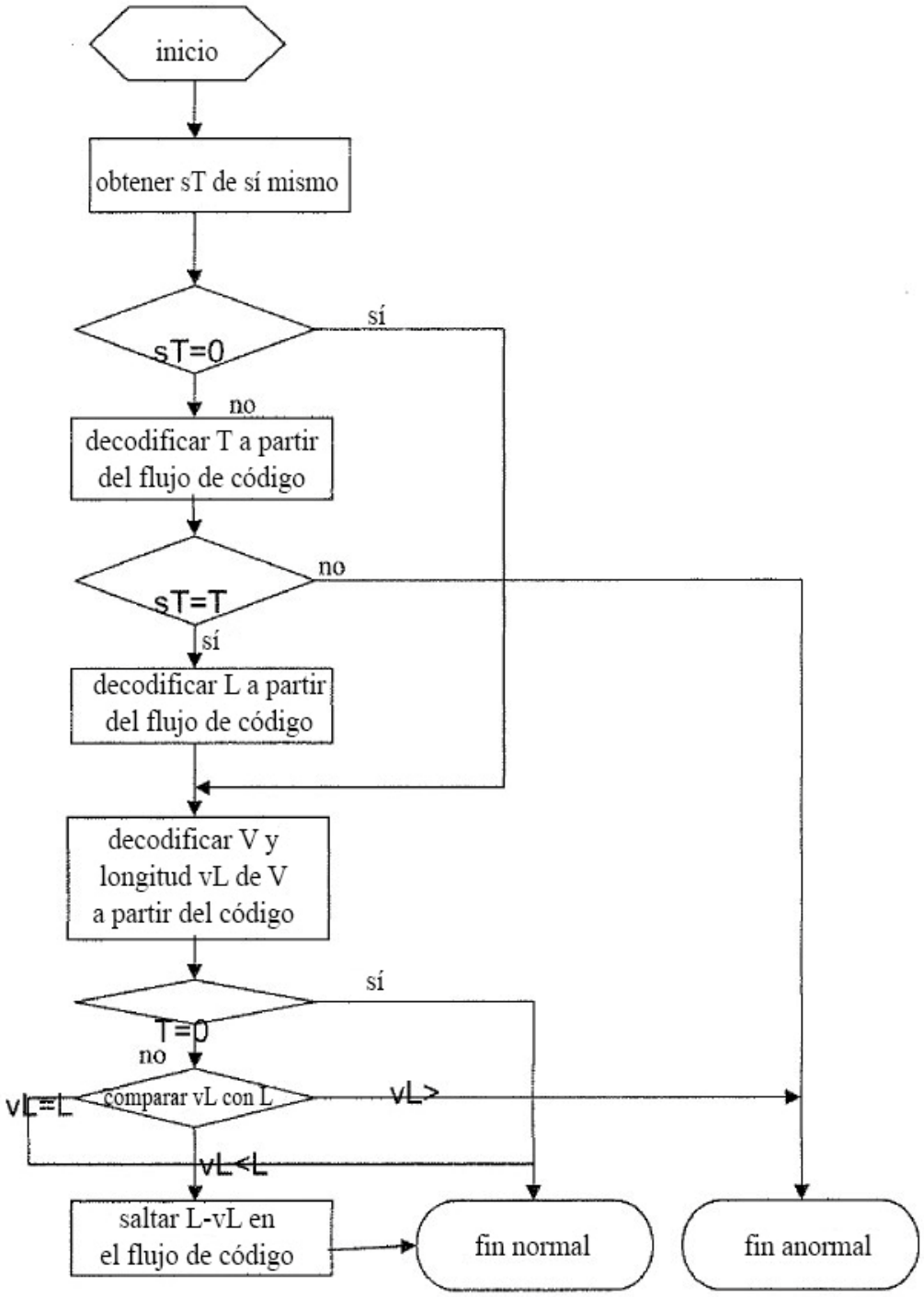
inicio
obtener sT de sí mismo
sT=0
sí
no
decodificar T a partir del flujo de código
sT=T
no
sí
decodificar L a partir del flujo de código
decodificar V y longitud vL de V a partir del código
T=0
sí
no
comparar vL con L
vL=L
vL>
vL<L
saltar L-vL en el flujo de código
fin normal
fin anormal

Fig. 6