(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 338 670**

(51) Int. Cl.:
G06F 9/46 (2006.01)
G06F 1/00 (2006.01)
H04L 29/06 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Número de solicitud europea: **05745281 .5**
(96) Fecha de presentación : **04.05.2005**
(97) Número de publicación de la solicitud: **1877899**
(97) Fecha de publicación de la solicitud: **16.01.2008**

(54) Título: **Procedimiento y sistema para el procesamiento de flujos de paquetes, y producto de programa informático para los mismos.**

(45) Fecha de publicación de la mención BOPI:
11.05.2010

(45) Fecha de la publicación del folleto de la patente:
11.05.2010

(73) Titular/es: **TELECOM ITALIA S.p.A.**
Piazza degli Affari 2
20123 Milano, IT

(72) Inventor/es: **Abeni, Paolo;**
Milani Comparetti, Paolo;
Di Paola, Sebastiano y
Lamastra, Gerardo

(74) Agente: **Ponti Sales, Adelaida**

DESCRIPCIÓN

Procedimiento y sistema para el procesamiento de flujos de paquetes, y producto de programa informático para los mismos.

Campo de la invención

La invención se refiere a técnicas de procesamiento de flujos de paquetes, por ejemplo, en una red de comunicación, y se ha desarrollado prestando especial atención para la posible aplicación de garantizar la seguridad de la red, y más específicamente, a técnicas de detección de intrusiones de red.

Descripción de la técnica relacionada

Un sistema de detección de intrusos de red (NIDS) es un dispositivo que monitorea la actividad en una red y analiza, por ejemplo, cada paquete que fluye en la red. El propósito del análisis es revelar los problemas de seguridad causados por la acción malévola de un agente externo o interno. Este agente puede ser un sistema automático (es decir, un virus o un gusano) o un intruso humano que trata de explotar algunas debilidades en el sistema para un propósito específico (es decir, el acceso no autorizado a datos reservados).

La implementación típica de un sistema de detección de intrusión de red se basa en el paradigma de "detección de uso indebido". Esto significa que un sensor tiene un conocimiento específico de la estructura de la acción malévola. Este conocimiento está incrustado en un conjunto de "firmas", que se asemejan a los patrones de ataques específicos. El corazón de este dispositivo es un mecanismo de procesamiento que compara rápidamente un amplio conjunto de firmas con cada paquete, para decidir si el paquete lleva un ataque o no; esto es a menudo llamado como "coincidencia de patrones".

Por supuesto, han sido discutidas otras aproximaciones en la literatura e implementado en dispositivos de trabajo reales; por ejemplo, sistemas que utilizan técnicas de inteligencia artificial, o analizan las propiedades estadísticas del tráfico de la red; sin embargo, una aproximación de patrón de coincidencia es la tecnología más común desplegada en este campo.

La eficacia de un sistema de detección de intrusión de red se mide generalmente en términos de "falsos positivos" y "falsos negativos". Los primeros son los paquetes que son erróneamente marcados como peligrosos, siendo estos últimos paquetes que llevan efectivamente un ataque, pero no están acompañados de ninguna firma; así, el sistema de detección de intrusión de red no es capaz de detectarlas.

La eficiencia de un sistema de detección de intrusión de red se mide en función del ancho de banda sostenido de la conexión monitorizada, o la fracción de paquetes descartados cuando el sensor está saturado por una cantidad excesiva de datos para analizar.

La evolución de la tecnología de redes es uno de los propulsores fundamentales del mundo de Internet. Durante los últimos años, el ancho de banda del usuario final ha aumentado un orden de magnitud, mientras que el ancho de banda disponible para la mayoría de sofisticadas infraestructuras está aumentando de manera aún más notable hacia cifras difíciles de predecir.

Sin embargo, este aumento de ancho de banda masivo también genera problemas de rendimiento diferentes en el contexto de los sistemas de detección de intrusiones de red. De hecho, el número de vulnerabilidades también está aumentando rápidamente, y este efecto está en gran medida amplificado por el aumento de ancho de banda. Por lo tanto, un sistema de detección de intrusiones de red estándar listo para usar puede ser incapaz de hacer frente a un ancho de banda sostenido del orden de Gigabit/seg. La solución más obvia a este problema es aplicar algún tipo de técnica de equilibrado del tráfico de la red para repartir la carga entre varios sistemas diferentes e independientes. De esta manera, el rendimiento de un sistema de detección de intrusiones de red estándar se puede expandir.

Esta aproximación plantea, sin embargo, varias limitaciones. En primer lugar, se necesitan más máquinas, lo que significa, entre otras cosas, costos adicionales de mantenimiento; además, el aparato de equilibrio de carga se convierte en un dispositivo crítico en la infraestructura; si el dispositivo se avería, el funcionamiento del conjunto del sistema se interrumpe.

Una aproximación diferente es la implementación de una arquitectura basada en tubería en un sistema multiprocesador (es decir, multi-CPU); mientras es eficaz, esta aproximación requiere una cuidadosa ingeniería de los diversos componentes que conforman el Sistema de Detección de Intrusión. Además, el equilibrio efectivo de las diferentes actividades entre las diferentes unidades centrales de procesamiento (CPU) puede ser difícil, porque cada CPU suele estar vinculada a una operación específica.

La idea de utilizar un sistema de equilibrio de carga para mitigar los efectos de una carga de red de alto ancho de banda ha sido ya descrita en la literatura.

Por ejemplo, la patente US-B-6 578 147 describe un sistema que utiliza múltiples sensores de detección de intrusiones de red conectado a un dispositivo de interconexión, tal como un enrutador o conmutador, que realiza una operación de equilibrio de carga. Los sensores funcionan en paralelo y cada uno recibe una parte del tráfico a través del dispositivo de interconexión, en un nivel basado en una sesión o en un nivel inferior (basado en paquetes). Dependiendo del tipo de dispositivo de interconexión (enrutador o conmutador), el mecanismo de equilibrio de carga que distribuye los paquetes puede ser interno o externo al dispositivo de interconexión. Además, dependiendo del nivel de la distribución de paquetes (basada en sesiones o basada en paquetes), los sensores comparten un analizador de red (si está basado en sesiones) o un analizador de red y un analizador de sesión (si se basa en paquetes).

Este documento de la técnica anterior describe, entre otras, una realización basada en una modificación del software de enrutamiento comúnmente desplegado en enrutadores del protocolo de Internet (IP) estándar. Así, la fracción de paquetes que se procesan mediante un sensor dado se basa esencialmente en la dirección de protocolo de Internet del destino, u otros tipos de decisiones de tipo de enrutamiento. En una realización adicional, la operación del equilibrio de carga se realiza mediante un conmutador, que es otra vez un dispositivo de interconexión, que usualmente opera en una capa diferente de la pila de red.

La patente US-A-2004/0107361 describe otra disposición para la implementación de un sistema de detección de intrusiones de red de alta velocidad. Esta solución de la técnica anterior explota una técnica en la que el final de la señal de interrupción no se entrega instantáneamente, sino que se introduce un cierto retraso. Esto permite que otros paquetes, que llegan durante ese período de tiempo, sean atendidos sin generar otra interrupción. Las interrupciones introducen sobrecargas, y la reducción de la sobrecarga hace que el sistema sea capaz de hacer frente a una carga mayor de la red. Además, el sistema introduce el uso de una "memoria intermedia de anillo", que permite la detección de software y que el dispositivo de red comparta la memoria intermedia para su funcionamiento. Al hacerlo, el sistema no tiene que copiar el paquete del dispositivo de memoria a la memoria del sistema.

Tal como se menciona expresamente en este documento, esta solución utiliza un sistema de un único procesador estándar en la implementación. La principal ventaja es una modificación del controlador del dispositivo de red para evitar múltiples copias de los paquetes procesados y para evitar que se generen un número excesivo de solicitudes de interrupción, cuando la carga de la red es excesivamente alta.

La patente US-B-6 631 422 describe un sistema de hardware que consiste en un dispositivo de red Ethernet modificado, que puede procesar directamente paquetes a bordo (sin la intervención explícita de la CPU central). El procesamiento de la entrada de red se distribuye a varias CPUs en sistemas de multiprocesador para mejorar el rendimiento de la red y tomar ventaja de la escalabilidad del multiprocesador. Los paquetes son recibidos por el adaptador de red y se distribuyen a N depósitos de memoria intermedia de recepción establecidos por el controlador del dispositivo, basado en N CPUs que están disponibles para el procesamiento de entrada de los paquetes. Cada depósito de memoria intermedia de recepción tiene una CPU asociada. Los paquetes acceden directamente a la memoria de uno de los N depósitos de memoria intermedia de recepción mediante una función hash, que se basa en la dirección MAC de origen, la dirección IP de origen o la fuente de paquetes y los números de puerto TCP de destino o la totalidad o una combinación de los anteriores. El mecanismo hash garantiza que la secuencia de los paquetes dentro de una sesión de comunicación determinada será preservada. La distribución se efectuará mediante el adaptador de red, que envía una interrupción a la CPU correspondiente al depósito de memoria intermedia de recepción, tras el paquete, utilizando un mecanismo de acceso directo de memoria, en el depósito de memoria intermedia. Esto optimiza la eficiencia del sistema de multiprocesador mediante la eliminación de cualquier dependencia del programador y aumentando el ancho de banda entre el controlador del dispositivo y el adaptador de red, mientras se mantiene las secuencias de paquetes adecuadas. El paralelismo aumenta así en el procesamiento I/O de red, la eliminación de cuellos de botella de la CPU para los I/O de red de alta velocidad y, así mejorando el rendimiento de la red.

La patente EP 1 349 065 describe una arquitectura genérica para la aplicación de un sistema de equilibrio de carga en un multiprocesador simétrico (SMP). El sistema aplica una función hash a los flujos de entrada, con el fin de distribuirlos entre las CPUs disponibles. Todo el proceso es dirigido por interrupciones, con la tarjeta Ethernet señalando a un procesador de que un nuevo paquete ha llegado. El procesador que sirve a la interrupción ejecuta una llamada de procedimiento diferido (DPC) de mayor prioridad, que asocia el paquete, lo etiqueta y lo asignar para su posterior procesamiento a una CPU específica.

El artículo de Z. Cao, Z. Whang, y E. Zegura: "Performance Hashing-Based Schemes for Internet Load Balancing", Tech. Rep. GIT-CC-99-14, College of Computing, Georgia Tech, 1999, describe varias técnicas que pueden utilizarse para dividir una carga de tráfico dada entre varios procesadores independientes.

El artículo de S. Lu, J. Gong, Rui S.: "A Load Balancing Algorithm for High Speed Intrusion Detection", APAN Meeting, 2003 (Busan, Corea) describe una función hash específica que es particularmente adecuada para aplicaciones de detección de intrusos.

Objeto y descripción de la invención

El objeto de la invención es aumentar la eficacia de las disposiciones de procesamiento en paralelo (por ejemplo, para su uso en un sistema de detección de intrusiones de red) cuando la carga computacional, es decir, el flujo de paquetes entrantes, se distribuye entre un conjunto de unidades de cálculo. Un objeto específico de la invención es

proporcionar esta disposición mejorada adaptada para confiar en hardware listo para usar, sin necesidad de ningún dispositivo de propósito especial para lograr esta eficiencia.

Según la presente invención, ese objeto se consigue mediante un procedimiento que tiene las características indicadas en las reivindicaciones adjuntas. La invención también se refiere a un sistema correspondiente, así como a un producto de programa de ordenador relacionado, que se puede cargar en la memoria de al menos un ordenador, y que incluye porciones de código de software para realizar las etapas del procedimiento de la invención cuando el producto se ejecuta en un ordenador. Tal como se usa aquí, la referencia a dicho producto de programa de ordenador está destinado a ser equivalente a la referencia a un medio legible por ordenador que contiene instrucciones para el control de un sistema informático para coordinar el funcionamiento del procedimiento de la invención. La referencia a "al menos una computadora" tiene la evidente intención de destacar la posibilidad de que la presente invención que se implemente en forma distribuida/modular.

Las reivindicaciones son una parte integral de la descripción de la invención aquí proporcionada.

Una realización preferida de la disposición aquí descrita proporciona así flujos de paquetes de procesamiento en una red mediante un sistema multiprocesador que incluye una pluralidad de unidades de procesamiento (CPUs, por ejemplo) mediante la distribución de paquetes para el procesamiento entre las unidades de procesamiento a través de una función de distribución. La función de distribución se asigna selectivamente a una de las unidades de procesamiento de la pluralidad.

La disposición aquí descrita tiene por objeto resolver una serie de problemas básicos que se plantean en las soluciones de la técnica anterior, mediante la implementación de una infraestructura de software específica, adaptada para soportar un tráfico de red de Gigabit/seg mediante el uso de hardware estándar, normalmente en forma de una máquina multiprocesador simétrica (SMP) estándar.

Un multiprocesador simétrico es esencialmente un ordenador que proporciona varias CPUs independientes que comparten el bus y la memoria. Las máquinas de multiprocesador simétricas son bastante comunes en estos días, y varios fabricantes de ordenadores las ofrecen a un costo muy conveniente.

A modo de referencia directa, si por ejemplo, un sistema de detección de intrusión de red se ejecuta sencillamente en una máquina multiprocesador simétrica estándar, no se producirían incrementos apreciables de rendimiento. Por ejemplo, en el caso de una SMP que incluye cuatro CPUs, estaría teóricamente previsto un aumento del rendimiento de más de cuatro veces respecto al rendimiento de una sola CPU. Sin embargo, el aumento de rendimiento sería mucho menor. Esto se debe a que la implementación típica de un sistema de detección de intrusión de red es del tipo de una sola tarea, y, como tal, no es capaz de aprovechar las múltiples CPUs disponibles en un SMP.

La disposición aquí descrita produce una arquitectura diferente para un sistema de detección de intrusión de red, con la participación de una modificación en el controlador del dispositivo de red, que aumenta efectivamente en máquinas multiprocesador simétricas estándar. La disposición aquí descrita se basa en un mecanismo multitarea modificado que permite la implementación de una aplicación del sistema de detección de intrusiones de red adecuado para una escala casi lineal en arquitecturas de multiprocesadores simétricos.

Una realización preferida de la disposición aquí descrita consiste en utilizar una única máquina multiprocesador simétrica con un puerto de red único para todo el proceso de tráfico de un enlace, por ejemplo un enlace de Gigabit/seg. La arquitectura del sistema correspondiente no requiere de ningún dispositivo intermedio, o ningún mecanismo de equilibrio de carga externo. Todo el trabajo de procesamiento se realiza en un único sistema, que es capaz de equilibrar dinámicamente la carga de tráfico entre las diferentes CPUs independientes. Al recurrir a una disposición de programación específica, este sistema es capaz de distribuir de manera efectiva las computaciones requeridas para realizar el equilibrio de carga y las operaciones de detección. De esta manera, la utilización del sistema se maximiza mediante la obtención de un mejor factor de escalado.

Breve descripción de los dibujos adjuntos

La invención será ahora descrita, a modo de ejemplo solamente, con referencia a las figuras adjuntas de dibujo, en donde:

- La figura 1 es un diagrama de bloques general de un sistema tal como se describe aquí;

- La Figura 2 es un diagrama de bloques funcional representativo de la operación de uno de los elementos que se muestran en la Figura 1;

- La Figura 3 es otro diagrama de bloques funcional representativo de la operación de uno de los elementos que se muestran en la Figura 1, y

- Las Figuras 4 y 5 son diagramas de flujo representativos de la operación del sistema tal como se describe aquí.

Descripción detallada de realizaciones preferidas de la invención

La disposición de procesamiento de ejemplo aquí descrita tiene por objeto el procesamiento de un flujo de entrada de paquetes recibidos a través de una llamada interfaz de sensor 101 asignado en un dispositivo de red y el controlador de dispositivo relacionado mediante el uso de una máquina multiprocesador simétrica (SMP) 100 que incluye una pluralidad de CPUs, por ejemplo cuatro CPUs.

Las CPUs en cuestión no están explícitamente presentadas como tales en ninguna de las figuras de dibujo adjuntas, que están destinadas principalmente a retratar la arquitectura lógica implementada a través de dichas CPUs.

Por ejemplo, el flujo de entrada de paquetes puede comprender paquetes intercambiados dentro de una red (no mostrada como un conjunto), con la disposición de procesamiento aquí descrita incluida en el sistema de detección de intrusión de red (NIDS) asociado con esa red. Las tareas de procesamiento realizadas con la finalidad de detectar intrusos dentro de la CPUs de la SMP 100 pueden ser de cualquier tipo conocido, y la naturaleza de este procesamiento es de por sí de un momento no específico para los fines de comprensión y puesta en práctica de la invención.

La disposición del sistema que se muestra en la Figura 1 está pensada para trabajar en relación con al menos dos interfaces de red diferentes. La interfaz de sensor 101 se utiliza para recibir el flujo de paquetes que se van a analizar. Una segunda interfaz (que no se muestra en los diagramas incluidos) se utiliza como una interfaz de control para realizar la administración del NIDS y transmitir las alertas sobre eventos de seguridad detectados. Esta es una configuración bastante estándar para esta máquina. Otras disposiciones, sin embargo, son admisibles para las interfaces de red.

La interfaz de sensor 101 tiene un ancho de banda suficiente y puede operar en modo de bus maestro, usando un área específica de la memoria principal para almacenar el paquete que recoge; el área de memoria ha de ser configurable mediante el controlador del dispositivo. El controlador del dispositivo de la interfaz 101 se utiliza para leer los paquetes. Este tipo de dispositivo activa típicamente el controlador Ethernet en modo promiscuo (el sistema captura todos los paquetes detectados en el cable, incluyendo paquetes que no están directamente dirigidos a ese controlador). En la disposición aquí descrita, el controlador del dispositivo de interfaz 101 opera solamente colocando los paquetes recibidos en una memoria intermedia compartida 102, asignada por el núcleo para este propósito, sin emitir ninguna interrupción. Típicamente, el procesamiento de paquetes normales (tal como el protocolo de control de transmisión/protocolo de Internet (TCP/IP) u otras operaciones de pila de red) está deshabilitado para los paquetes que llegan a través de la interfaz del sensor. El controlador hace que los paquetes sean accesibles a un programa de espacio de usuario mediante el uso de una asignación de memoria específica para la memoria intermedia compartida. Un programa de espacio de usuario es capaz de consultar el controlador del dispositivo y de obtener una dirección específica que puede ser mapeada en memoria para poder acceder a los paquetes.

La estructura de la memoria intermedia compartida 102 se muestra en la Figura 2. En particular, la memoria intermedia compartida 102 comprende celdas llenas y vacías, y hay dos punteros diferentes que se ocupan de diferentes celdas de memoria para leer o escribir un paquete. El puntero del lector 200 direcciona una celda completa que contiene el primer paquete que será procesado por el sistema. Las celdas llenas están designadas con la referencia 220, mientras que las celdas vacías están designadas con la referencia 230. El puntero escritor 210 direcciona la primera celda vacía después de las celdas llenas, y esta celda vacía recibirá el primer paquete que llega desde la red. La memoria intermedia compartida 102 se utiliza así como una memoria intermedia circular.

La comunicación entre el controlador Ethernet y la aplicación del espacio del usuario se basa así en el paradigma de “un solo escritor/un solo lector” estándar, por lo que esta memoria intermedia no necesita utilizar ningún bloqueo de exclusión mutua por ambas partes. Si la memoria intermedia está llena, el controlador del dispositivo descarta el paquete y marca este evento en su registro estadístico. En cualquier momento, un módulo de espacio de usuario puede consultar las estadísticas para saber cuántos paquetes se han perdido, o cuántos paquetes se espera en la cola.

La porción del espacio de usuario del sensor se implementa usando tareas N+1, donde N es el número de unidades centrales de procesamiento independientes. La primera tarea es una “tarea de sondeo” 103; las otras tareas N son “tareas de detección” 106.

La tarea de sondeo 103 lee los paquetes de la memoria intermedia compartida 102, y los procesa mediante una función hash 104. La función hash 104 etiqueta cada paquete con un número natural en el intervalo [0, ..., N-1]. La tarea de sondeo 103 elimina el paquete desde la memoria intermedia compartida 102 y copia el paquete en una cola implementada como una memoria intermedia circular 105. Esta memoria intermedia se comparte entre la tarea de sondeo 103 y una tarea de detección única 106 (acoplándose así a esta tarea de detección única), y de nuevo se trata de un contexto de único lector/único escritor, de modo que no hay necesidad de bloqueos de exclusión mutuos. Dicho de otra manera, el paradigma de único lector/único escritor se aplica también a los paquetes incluidos en las memorias intermedias 105.

Estas otras N tareas de detección 106 realizan las operaciones básicas del sensor (la coincidencia de patrones y las demás actividades pertinentes). La etiqueta generada por la aplicación de la función hash 104 identifica qué tarea tiene que analizar el paquete específico.

Cada vez que se detecta una condición peligrosa, la tarea de detección 106 genera una alerta, al invocar la subrutina de generación de alerta 107, que a su vez llena un bloque de memoria con la información adecuada, y proporciona un puntero a la memoria donde se almacena el paquete. Este bloque se coloca en una cola de propósito especial de primera entrada primera salida (FIFO) 108, que la pone a disposición para su procesamiento adicional sin la necesidad copiar explícitamente la memoria que contiene la alerta. Esta cola de primera entrada primera salida se implementa como un controlador de dispositivo especial en el núcleo, que proporciona acceso directo al bloque de memoria compartida en la memoria intermedia compartida 102 que contiene la alerta mediante el uso una asignación de memoria dedicada (a través de un sistema de llamada mmap ()). Mediante el uso de una cola primera entrada primera salida habitual, se evita la copia extra que se requiere para procesar la alarma en otro proceso.

Un aspecto interesante de la solución aquí descrita se encuentra en la elección adecuada de la función hash. Esta función preferiblemente muestra por lo menos una de las siguientes propiedades, más preferiblemente las presenta todas:

- *Coherencia*: La función debe etiquetar con el mismo índice paquetes diferentes que pertenecen al mismo flujo de red; flujo de red significa una secuencia de paquetes que constituyen un único TCP, UDP, ICMP, u otra sesión de protocolo. El hecho de que estos paquetes son procesados por la misma tarea de detección es ventajoso porque el procedimiento de detección es un proceso basado en el estado, y la información sobre los paquetes anteriores, es importante para tomar la decisión correcta;

- *Equidad*: La probabilidad de que un paquete determinado, perteneciente a un flujo de red específico, es asigne a una tarea de detección específica debe ser distribuida de manera uniforme e igual a 1/N. Esta característica garantiza una distribución equitativa de la carga entre las diferentes CPUs;

- *Seguridad*: Cualquier observador externo no debería ser capaz de crear una secuencia de flujos que está etiquetados de acuerdo con una secuencia elegida por el propio observador. En pocas palabras, debería ser imposible para cualquier observador externo crear una secuencia de flujos independientes que están etiquetados con el mismo número. Si este requisito no se cumple, un atacante externo puede forzar el sistema a funcionar como un sistema de una sola CPU, ya que la distribución de la carga sobre las diferentes CPUs se determina mediante la función hash.

Preferentemente, la función hash también debe ser una que sea eficiente para el cálculo, y esto significa que su complejidad debe ser una función lineal (o mejor, en el sentido de la complejidad) de la dimensión de la entrada.

La propiedad de coherencia puede lograrse con facilidad mediante la restricción del cálculo hash en el triplete: Fuente de paquetes IP, Destino de paquetes IP, Protocolo IP.

Si el Puerto de origen y el puerto de destino (si están disponibles) también se añaden a los parámetros de entrada, se obtiene un mayor grado de aleatorización (mejor equidad). A los paquetes fragmentados se les puede asignar una etiqueta que no es la misma etiqueta asignada a paquetes IP reconstruidos y a otros paquetes no fragmentados en el mismo flujo (donde también los puertos pueden ser tenidos en cuenta para el cálculo hash). Reinyectar el paquete en la cola de procesamiento adecuada, después de que los paquetes fragmentados se han vuelto a juntar puede resolver este problema.

Para garantizar la propiedad de seguridad, es suficiente realizar una operación XOR entre los datos de entrada y una clave aleatoria secreta, con una longitud que corresponda a la de los datos de entrada. La función hash no es lineal, por lo tanto, el XOR no distribuye a sí mismo el resultado de la función; es decir,

$$\text{Hash}\ (A \oplus B) \neq \text{Hash}(A) \oplus \text{Hash}(B)$$

De esta forma, un atacante, que no puede observar directamente el resultado de la dispersión y no conoce la clave secreta (pero sabe el algoritmo que implementa la función “hash”) será incapaz de adivinar una secuencia de datos de entrada que le daría el misma resultado idéntico de varios patrones de entrada diferentes.

Si la función hash utiliza una semilla interna para hacer su cálculo, es posible obtener la propiedad de seguridad de forma aleatoria seleccionando esta semilla (no hay necesidad de XOR de una secuencia secreta con los datos de entrada).

Basado en las pruebas realizadas hasta ahora por los inventores, una función que es especialmente adecuada para esta operación es la descrita en “Hash Function for Hash Table Lookup” por R.J. Jenkins libremente disponible en la fecha de presentación de esta solicitud en la dirección del sitio web: http://burtleburtle.net/bob/hash/evahash.html. Esta función se incrusta una semilla aleatoria, que puede ser utilizada para proporcionar la propiedad de seguridad requerida. De hecho, con la selección aleatoria de las semillas (por supuesto, valores de degeneración tales como 0 deben descartarse durante esta selección), es posible modificar la salida de la función hash de una manera impredecible.

El mecanismo adoptado para la selección de qué actividad se tiene que ejecutar en una CPU en concreto en un instante de tiempo es otra característica de la solución aquí descrita. Este mecanismo pertenece a una categoría que por lo general se refiere al “algoritmo de programación” en la literatura del sistema operativo. La arquitectura general

del sistema está compuesta por N+1 tareas diferentes, y el sistema tiene N CPUs diferentes. Durante el arranque, el sistema detecta cuántas CPUs están disponibles y crea N procesos diferentes; cada uno de los cuales es asignado a una CPU específica, mediante el uso del mecanismo de "afinidad" previsto por el núcleo. La "afinidad" de un proceso indica el núcleo que un proceso específico sólo debe ejecutarse en una CPU específica. Esto favorece la localidad del programa, evitando costosas operaciones computacionales relacionadas con la operación de conmutación.

Cada proceso puede operar en tres funciones diferentes:

- Función de sondeo,

- Función de detección, y

- Función de reposo.

En cada momento, solamente un único proceso puede operar en la función de sondeo, y este proceso tiene una señal especial para este propósito, la llamada "Credencial de sondeo". Los parámetros enteros Psi+(Ψ+) y Psi-(Ψ-) representan los límites en las colas de detección 105, donde el segundo límite (Ψ-) es menor que el primer límite (Ψ+).

En una realización preferida, se ha desarrollado el siguiente algoritmo para programar las actividades en el sistema.

Un credencial de sondeo 300 es una estructura de datos especial, descrita en la figura 3, que contiene la siguiente información: un conjunto de N marcadores booleanos 301, siendo N el número de tareas de detección 105, es decir, N = 4 en el ejemplo considerado), usado para marcar un proceso de reposo, y otro indicador booleano 302, que indica si el credencial de sondeo 300 es libre (no asignado a ningún proceso) o no. El credencial de sondeo 300 debe ser accedido por varios procesos independientes, por lo que la estructura de datos correspondiente está protegida mediante el uso de un bloqueo 303.

La Figura 4 muestra un diagrama de flujo de un primer ejemplo de realización de la disposición aquí descrita.

1. En el arranque (etapa 400), el credencial de sondeo 300 es asignado a un proceso aleatorio; así, este proceso opera en la función de sondeo, mientras que los otros N-1 procesos operan en la función de detección. Todos los demás procesos se ponen en el estado de reposo, y la marca correspondiente en el credencial de sondeo 300 se establece en verdadera.

2. El primer proceso está ejecutando la tarea de sondeo, mientras que los otros procesos están en reposo.

3. El proceso en la función de sondeo en una etapa 402 empieza a extraer los paquetes de la memoria intermedia 102 compartida con el controlador del dispositivo Ethernet 101; en una etapa 404 se aplica la función hash en este paquete, y se inserta el paquete en la cola de detección 105 del proceso específico, identificado por el resultado hash. En una etapa 406 se controla si la cola de detección 105 que debe aceptar el paquete está completa. En caso afirmativo, en una etapa 408 el paquete se deja caer y un contador de caídas se incrementa.

4. En una etapa 410 se realiza un control de si el proceso que sirve a una cola de detección específica 105 está en reposo. En caso afirmativo, el proceso en la función de sondeo lo señala, mediante el uso del siguiente procedimiento:

- adquiere el bloqueo de 303 en el credencial de sondeo 300,

- se restablece la marca correspondiente en la matriz,

- en una etapa 412 activa el correspondiente proceso de reposo, y

- libera el bloqueo 303.

5. El proceso continúa, en una etapa 414, para operar en la función de sondeo hasta que se satisface una de estas tres condiciones:

- su propia cola de detección 105 contiene más de Ψ+ paquetes,

- su propia cola de detección 105 contiene más de Ψ- paquetes y un paquete se deja caer en cualquiera de la cola de detección 105, y

- su propia cola de detección 105 contiene más de Ψ- paquetes y la cola de la sondeo se ha vaciado.

Cuando se cumple una de estas condiciones, en una etapa 416 el proceso de sondeo libera el credencial de sondeo 300 y en una etapa 418 se conmuta para operar en la función de detección; el procedimiento es el siguiente:

- el proceso adquiere el bloqueo 303,

- uno de los procesos de reposo (si está disponible) se activa; usualmente se elige el primero disponible en orden de todos entre sí,

- el credencial de sondeo 300 se marca como no asignado (libre),

- el proceso cambia a la función de detección, y

- el proceso libera el bloqueo 303.

5. Cuando opera en la función de detección, en una etapa 420 el proceso realiza la detección de un paquete de red. El proceso continúa haciéndolo hasta que en una etapa 422 la cola de detección 105 está vacía; cuando esto ocurre, el proceso tiene que decidir si ir a reposo o cambiar a la función de sondeo; el procedimiento de decisión es el siguiente:

- el proceso adquiere el bloqueo 303,

- si el credencial de sondeo 300 no se asigna (es decir, libre), que se comprueba en una etapa 424, en una etapa 428 el proceso de toma el control de la credencial de sondeo 300 y cambia a la función de sondeo,

- además, en una etapa 426 el proceso establece su propia marca en la matriz de marcas 301 del credencial de sondeo, libera el bloqueo 303, y se prepara para el reposo.

La experimentación con diferentes valores de $\Psi+$ y $\Psi-$ ha indicado que es seguro asumir $\Psi- = 0$ en este caso; los valores adecuados para $\Psi+$ son alrededor de 1/10 del tamaño de la cola de detección 105.

Una implementación alternativa para el algoritmo de programación se describe ahora; esta implementación representa una segunda realización de la solución aquí descrita.

El objetivo de esta segunda implementación es proporcionar un procedimiento que reduce la necesidad de contención del bloqueo, mediante la adopción cuidadosa del paradigma de único lector/único escritor. En esta realización alternativa, el credencial de sondeo no contiene la matriz de marcas 301 requerida en el caso anterior. Esta disposición es sustituida por una matriz de pares de contadores único lector/único escritor, definidos “Contador de solicitud de sondeo” y “Contador completo de sondeo”. Estos contadores se pueden acceder en cualquier momento sin ningún tipo de bloqueo. Cada proceso tiene su propio par de contadores.

El credencial de sondeo 300 está todavía protegido con un bloqueo 303; sin embargo, este bloqueo 303 tiene que ser adquirido/liberado solamente en condiciones especiales; en la implementación anterior, es necesario adquirir el bloqueo 303 cada vez que un proceso tiene que modificar su estado actual.

Para un mejor equilibrio de la carga computacional entre la tarea de sondeo y las tareas de detección, el proceso que tiene el credencial de sondeo 300 se le permite realizar una iteración en la función de detección de M iteraciones en la función de sondeo. El límite de M es igual a $N*(N+1)/2$ (donde N es el número de CPUs disponibles). Este valor inicial de M puede obtenerse mediante la siguiente asunción:

- Siendo $T_{Detección}$ el tiempo necesario para una interacción de detección y T_{Sondeo} el tiempo necesario para una interacción de sondeo $T_{Detección}/T_{Sondeo} = (N+1)$; este es un límite superior razonable a esta relación, usualmente, la proporción será menor;

- ningún paquete se pierde.

En un determinado período de tiempo T, para la tarea que opera en la función de sondeo:

- $K* T_{Detección} + K*M*T_{Sondeo} = T$, donde K es el número de paquetes procesados en la función de detección, mientras está en la función de sondeo, en un marco de tiempo T.

Mientras que para las otras tareas, que operan en la función de detección, es:

- $K'*T_{Detección} = T$, donde K’ es el número de paquetes que cada tarea de detección procesa en un marco de tiempo T.

A partir de las relaciones anteriores, el valor de K' se puede calcular como:

$$K' = (K*M*T_{sondeo} + K*T_{detección})/T_{detección}$$

Teniendo en cuenta el número total de paquetes sondeados, y el número total de paquetes que son procesados en el sistema de detección, si ningún paquete se pierde, el número de paquetes sondeados puede ser igual al número de paquetes analizados en un marco de tiempo determinado:

$$N_{sondeo}(T) = K*M$$

$$N_{detección}(T) = (N-1)*K' + K$$

Mediante el ajuste

$$N_{sondeo}(T) = N_{detección}(T)$$

es

$$M = N + M*(N-1)*T_{sondeo}/T_{detección}$$

En general:

$$M = N/(1-(N-1)*T_{sondeo}/T_{detección}$$

Si se aplica la primera hipótesis, al sustituir el factor

$$T_{sondeo}/T_{detección} = 1/(N+1)$$

el valor para el parámetro M es

$$M = N/(1 - (N-1)/(N+1)) = N*(N+1)/2$$

Una segunda realización de la solución aquí descrita se explica ahora con referencia a la Figura 5.

1) Después del arranque (etapa 500), en una etapa 502 se selecciona un proceso aleatorio para soportar el credencial de sondeo 300; por lo tanto, este proceso opera en la función de sondeo, mientras que los otros N-1 procesos operan en la función de detección. El primer proceso ejecuta la tarea de sondeo, mientras que los otros procesos ejecutan la tarea de detección.

2) El proceso en la función de sondeo en una etapa 504 empieza a extraer paquetes de la memoria intermedia 102 compartida con el controlador del dispositivo Ethernet de la interfaz 101; se aplica la función hash en este paquete, y se inserta el paquete en la cola de detección 105 del proceso específico, identificado por el resultado hash. Los otros procesos, que se ejecutan en el modo de detección, empiezan a consumir paquetes de sus colas de detección 105.

3) Cuando el proceso que se ejecuta en la función de sondeo, en una etapa 506 ha completado M iteraciones, ejecuta en una etapa 508 una sola iteración de función de detección, y luego vuelve a su función de computación de sondeo. Se trata de una conmutación de funciones temporal.

4) El proceso continúa operando en la función de sondeo hasta que en una etapa 510 una de estas condiciones es verdadera:

- su propia cola de detección 105 contiene más de Ψ+ paquetes;

- se ha completado más de P conmutadores de función temporal. El valor de P se puede configurar de manera arbitraria; no influye en el funcionamiento general del procedimiento; un valor común es la mitad de la longitud de la memoria intermedia compartida 102.

A continuación, en una etapa 512, el proceso pasa la credencial de sondeo 300 al proceso siguiente. Esto se logra:

- incrementando el contador de solicitud de credencial de sondeo del proceso de destino, y

- disminuir su propio contador completado de credencial de sondeo.

A continuación, en una etapa 514, el proceso comienza a operar en el modo de detección. Si el número de paquetes en la memoria intermedia compartida 102 es menor que un relación predefinida entre la longitud de la cola completa, normalmente igual al 50% de la longitud de la cola, el valor de M se incrementa en uno; si el número de paquetes es mayor que una segunda relación predefinida entre la longitud de la cola completa, normalmente igual al 75% de la longitud de la cola, el valor M se reduce en uno; de lo contrario, M se conserva intacto. El valor mínimo de M es N (el número de CPUs disponibles).

5) El proceso que se elija, en una etapa 516, para recibir la credencial de sondeo 300 necesariamente opera en el modo de detección. Si su propia cola de detección 105 contiene menos de Ψ paquetes, acepta en una etapa 518 la credencial de sondeo 300, y en una etapa 522 empieza a operar en la función de sondeo, prácticamente reemplazando el otro proceso.

6) Si el proceso de recepción no puede aceptar el credencial de sondeo 300 (porque la cola contiene más de Ψ paquetes), en una etapa 520, pasa el credencial de sondeo 300 al proceso siguiente. Esto se realiza otra vez mediante:

- incrementando el contador de solicitud de credencial de sondeo del proceso de destino, y disminuyendo su propio contador completo de credencial de sondeo.

7) En una etapa 524, si no hay ningún proceso que pueda aceptar el credencial de sondeo, porque ha pasado a lo largo de todos los procesos posibles, el credencial de sondeo sigue sin asignarse; así, en una etapa 528 el proceso marcará el credencial de sondeo como libre, reiniciando la marca 302.

8) Como que no se está actuando en la función de sondeo, y todos los procesos están trabajando en la función de detección, después de algún tiempo, una o más colas de detección 105 contienen menos de Ψ paquetes. Si no está asignado el credencial de sondeo 300, todos los procesos que sirven a las colas de detección 105 con menos de Ψ paquetes pueden tratar de obtener el bloqueo 303 en el credencial; el primero que lo consigue, obtiene el credencial y comienza a operar en la función de sondeo, mientras que los otros reanudarán su función de detección. Esta es la única contención que se puede producir en esta segunda solución; este es el único caso en que el bloqueo debe ser adquirido y liberado.

De manera significativa, todo el procesamiento que se puede hacer sin usar ningún bloqueo para proteger las colas compartidas, ya que se acceden usando el paradigma único lector/único escritor. El único bloqueo necesario se coloca en el credencial de sondeo 300, cuando es necesario adquirirlo después de cada proceso en el sistema que se negó a hacerlo (porque su propia cola de detección 105 contiene más de Ψ+ paquetes); de hecho, en este momento, puede haber más de un único proceso que trata de obtener el credencial de sondeo 300. Un mecanismo de bloqueo es necesario para garantizar que sólo un único proceso eventualmente lo consigue. Es posible ajustar el procedimiento seleccionando adecuadamente los valores de los límites; empíricamente, se han adoptado los valores de Ψ- = 0,3*QL, Ψ+ = 0,7*QL, siendo QL la duración máxima de la cola.

Cualquier función hash que garantiza las tres propiedades mencionadas (coherencia, equidad y seguridad) puede ser adoptada. Por ejemplo, la clásica función hash criptográfica tal como MD-4, MD-5 o SHA-1 puede utilizarse, aunque son computacionalmente costosos, en comparación con otro tipo de funciones. La función seleccionada en la solución aquí descrita es empíricamente una de las funciones más eficaces que pueden ser utilizados para este propósito.

En una realización diferente, también es posible sustituir la función hash, que es sin estado, con un mecanismo de estado seguro. En este escenario, el mecanismo de selección “recuerda” la CPU que se ha asignado para el análisis de los paquetes que pertenecen a un período específico. Cuando un paquete que inicia un nuevo período de sesiones llega, el paquete (y el correspondiente período de sesiones) se asigna a la CPU que tiene el menor número de paquetes en proceso. Con esta aproximación, es posible lograr un óptimo equilibrio de carga; sin embargo, más memoria y más tiempo de cálculo se requieren para hacer un ciclo de sondeo.

En una realización diferente, los paquetes en la misma sesión no deben ser analizados por la misma tarea de detección. No obstante, el análisis de los flujos de datos para la detección de intrusos se basa en el estado, y es necesario recordar algunos datos sobre el período de sesiones, para detectar adecuadamente una intrusión. Por lo tanto, en esta realización un área de memoria compartida se utiliza para colocar información sobre los flujos de datos monitorizados y permitir que todos los procesos puedan manipularlos; esta información de estado puede incluir el estado de una conexión TCP o pseudo-conexión UDP, así como fragmentos de paquetes IP y paquetes TCP que se van a volver a unir.

Esta información compartida puede ser manipulada de forma segura sólo mediante el uso de las operaciones que no se pueden interrumpir. Por lo tanto, surge la necesidad de proteger la memoria intermedia compartida con algún mecanismo de bloqueo. Por lo tanto, esta aproximación aumenta la contención entre los diferentes procesos, pero permite que los paquetes de la misma sesión sean analizados mediante procesos de detección diferentes. En este caso, la función hash no es necesaria para cumplir con la propiedad de la coherencia, y puede distribuir mejor la carga entre las diferentes CPUs.

Sin prejuicio de los principios subyacentes de la invención, los detalles y las realizaciones pueden variar, también sensiblemente, con referencia a lo que se ha descrito a modo de ejemplo solamente, sin apartarse del alcance de la invención tal como se define en las reivindicaciones adjuntas.

Referencias citadas en la descripción

Esta lista de referencias citadas por el solicitante está prevista únicamente para ayudar al lector y no forma parte del documento de patente europea. Aunque se ha puesto el máximo cuidado en su realización, no se pueden excluir errores u omisiones y la OEP declina cualquier responsabilidad al respecto.

Documentos de patente citados en la descripción

- US 6578147 B [0012]
- US 20040107361 A [0014]
- US 6631422 B [0016]
- EP 1349065 A [0017]

Documentos no procedentes de patentes citados en la descripción

- Performance Hashing-Based Schemes for Internet Load Balancing. Z. **Cao**; Z. **Whang**; E. **Zegura**. *Tech. Rep*. GIT-CC-99-14. College of Computing [0018]
- S. **Lu**; J. **Gong**; S. **Rui**. A Load Balancing Algorithm for High Speed Intrusion Detection. *APAN Meeting*, 2003 [0019]
- R. J. **Jenkins** freely. Hash Function for Hash Table Lookup [0048]

REIVINDICACIONES

1. Procedimiento de procesamiento de flujos de paquetes mediante un sistema multiprocesador (100) que incluye una pluralidad de unidades de procesamiento, incluyendo el procedimiento la etapa de distribuir dichos paquetes para su procesamiento entre las unidades de procesamiento de dicha pluralidad a través de una función de distribución (103, 104), en donde dicha función de distribución (103, 104) se asigna selectivamente a una de las unidades de procesamiento de dicha pluralidad,

caracterizado por el hecho de que dicha función de distribución incluye una función hash (104) y dicha función hash incluye una clave aleatoria secreta.

2. Procedimiento según la reivindicación 1, **caracterizado** por el hecho de que incluye la etapa de generar etiquetas a través de dicha función hash (104), identificando las etiquetas generadas por dicha función hash (104) qué unidad de procesamiento de dicha pluralidad tiene que procesar un paquete específico.

3. Procedimiento según la reivindicación 1, **caracterizado** por el hecho de que incluye la etapa de disponer dicha función de distribución como una combinación de una función de sondeo (103) que lee los paquetes de entrada en dichos flujos de paquetes y dicha función hash (104) que actúa en dichos paquetes leídos mediante dicha función de sondeo (103).

4. Procedimiento según la reivindicación 1, **caracterizado** por el hecho de que dicha etapa de distribución de dichos paquetes incluye la etapa de copiar los paquetes entrantes en dicho flujo en una memoria intermedia (105) acoplada a una sola unidad de procesamiento de dicha pluralidad.

5. Procedimiento según la reivindicación 1, **caracterizado** por el hecho de que incluye la etapa de recepción (102, 105) de los paquetes en dichos flujos según el paradigma único escritor/único lector.

6. Procedimiento según la reivindicación 5, **caracterizado** por el hecho de que incluye la etapa de recepción (102, 105) de los paquetes en dichos flujos según el paradigma único escritor/único lector tanto antes (102) como después (105) de su distribución a través de dicha función de distribución (103, 104).

7. Procedimiento según la reivindicación 1, **caracterizado** por el hecho de que incluye la etapa de configuración de dicha función de distribución (103, 104), para satisfacer al menos una de las siguientes características:

- dicha función de distribución (103, 104) distribuye a una determinada unidad de procesamiento en dicha pluralidad de diferentes paquetes que pertenecen al mismo flujo de entrada,

- la probabilidad de distribuir dichos paquetes entrantes en dichos flujos a dichas unidades de procesamiento (105) en dicha pluralidad se distribuye uniformemente,

- dicha función de distribución (103, 104) distribuye dichos paquetes para su procesamiento entre las unidades de procesamiento de dicha pluralidad según una secuencia creada de forma independiente.

8. Procedimiento según la reivindicación 1, **caracterizado** por el hecho de que incluye dichas unidades de procesamiento en dicha pluralidad que realizan una función seleccionado entre:

- una función de sondeo, asignado en cada momento sólo para una seleccionada de dichas unidades de procesamiento de dicha pluralidad para realizar dicha función de distribución,

- una función de procesamiento de dichos paquetes en dichos flujos de paquetes, y

- una función inactivo en el que la unidad de procesamiento respectiva de dicha pluralidad está inactiva.

9. Procedimiento según la reivindicación 1, **caracterizado** por el hecho de que una unidad de procesamiento de dicha pluralidad asignada a dicha función de distribución suspende su actividad de distribución cuando se satisface al menos una de las siguientes condiciones:

- la cola respectiva de paquetes (105) a procesar por dicha una unidad de procesamiento ha llegado a un primer límite (Ψ+),

- la cola respectiva de paquetes (105) a procesar por dicha una unidad de procesamiento ha llegado a un segundo límite (Ψ-), menor que dicho primer límite, y al menos un paquete es entregado en cualquiera de las respectivas colas (105) de las unidades de procesamiento de dicha pluralidad, y

- la cola respectiva de procesamiento (105) contiene más paquetes de dicho segundo límite (Ψ-) y la cola de entrada (102) al sistema (100) está vacía.

10. Sistema multiprocesador (100), que incluye una pluralidad de unidades de procesamiento para el procesamiento de flujos de paquetes, en el que dichas unidades de procesamiento de dicha pluralidad están configuradas para soportar una función de distribución (103, 104) para la distribución de dichos paquetes a las unidades de procesamiento de dicha pluralidad para su procesamiento, en el que dichas unidades de procesamiento de dicha pluralidad están configurados para asignar de manera selectiva dicha función de distribución (103, 104),

caracterizado por el hecho de que dichas unidades de procesamiento de dicha pluralidad están configuradas para soportar una función de distribución (103, 104) que incluye una función hash (104) y dicha función hash incluye una clave aleatoria secreta.

11. Sistema según la reivindicación 10, **caracterizado** por el hecho de que dichas unidades de procesamiento de dicha pluralidad están configuradas para soportar una función de distribución (103, 104) para la generación de etiquetas a través de dicha función hash (104), identificando las etiquetas generadas por dicha función hash (104) qué unidad de procesamiento de dicha pluralidad tiene que procesar un paquete específico.

12. Sistema según la reivindicación 10, **caracterizado** por el hecho de que dichas unidades de procesamiento de dicha pluralidad están configuradas para apoyar una función de sondeo (103) que lee los paquetes de entrada en dichos flujos de paquetes y dicha función hash (104) actúa sobre dichos paquetes leídos mediante dicha función de sondeo (103).

13. Sistema según la reivindicación 10, **caracterizado** por el hecho de que incluye memorias intermedias (105) adaptadas para acoplarse cada una a de respectiva unidades de procesamiento de dicha pluralidad para copiar en las mismas los paquetes entrantes en dichos flujos a procesarse mediante dicha respectiva una de las unidades de procesamiento de dicha pluralidad.

14. Sistema según la reivindicación 10, **caracterizado** por el hecho de que incluye:

- una memoria intermedia de entrada (102) para cargar los paquetes entrantes en dichos flujos,

- un controlador de dispositivo (101) configurado para comprobar si dicha memoria intermedia de entrada (102) está llena, y descartar los paquetes entrantes cuando dicha memoria intermedia de entrada (102) está llena.

15. Sistema según la reivindicación 10, **caracterizado** por el hecho de que el sistema está configurado (102, 105) para recibir los paquetes en dichos flujos de acuerdo con el paradigma único escritor/único lector, y por el hecho de que incluye al menos una memoria intermedia (102, 105) para recibir los paquetes en dichos flujos de paquetes, en dicha al menos una memoria intermedia (102, 105) proporcionándose:

- un puntero lector (200) que dirige cualquier celda completa que contiene un primer paquete a procesar, y

- un puntero escritor (110) que dirige la primera celda vacía después de la celda llena para recibir cualquier paquete entrante siguiente.

16. Sistema según la reivindicación 15, **caracterizado** por el hecho de que incluye registros (102, 105) configurados para recibir (102, 105) los paquetes en dichos flujos según el paradigma único escritor/único lector antes (102) y después (105) de su distribución a través de dicha función de distribución (103, 104).

17. Sistema según la reivindicación 10, **caracterizado** por el hecho de que dichas unidades de procesamiento de dicha pluralidad están configuradas para soportar una función de distribución (103, 104) que satisface al menos una de las siguientes características:

- dicha función de distribución (103, 104) distribuye una determinada unidad de procesamiento en dicha pluralidad de diferentes paquetes que pertenecen al mismo flujo de entrada,

- la probabilidad de distribuir dichos paquetes entrantes en dichos flujos en dichas unidades de procesamiento (105) en dicha pluralidad se distribuye uniformemente,

- dicha función de distribución (103, 104) distribuye dichos paquetes para su procesamiento entre las unidades de procesamiento de dicha pluralidad según una secuencia creada de forma independiente.

18. Sistema según la reivindicación 10, **caracterizado** por el hecho de que dichas unidades de procesamiento de dicha pluralidad están configuradas para realizar de manera selectiva una función seleccionado de:

- una función de sondeo, asignado en cada momento sólo para una seleccionada de dichas unidades de procesamiento de dicha pluralidad para realizar dicha función de distribución,

- una función de procesamiento de dichos paquetes en dichos flujos de paquetes, y

- una función inactivo en el que la unidad de procesamiento respectiva de dicha pluralidad está inactiva.

19. Sistema según la reivindicación 10, **caracterizado** por el hecho de que dichas unidades de procesamiento de dicha pluralidad se configuran para interrumpir la función de distribución asignada a las mismas, cuando se satisface al menos una de las siguientes condiciones:

- la cola respectiva de paquetes (105) a procesar mediante dicha una unidad de transformación ha llegado a un primer límite (Ψ+),

- la cola respectiva de paquetes (105) a procesar mediante dicha una unidad de procesamiento ha llegado a un segundo límite (Ψ-), más bajo que dicho primer límite, y al menos un paquete es entregado en cualquiera de las colas respectivas (105) de las unidades de procesamiento de dicha pluralidad, y

- la cola de procesamiento respectiva (105) contiene más paquetes de dicho segundo límite (Ψ-) y la cola de entrada (102) al sistema (100) está vacía.

20. Sistema de detección de intrusiones de red para monitorizar la actividad en una red mediante el análisis de los paquetes de dicha red, que comprende un sistema multiprocesador (100) realizado según cualquiera de las reivindicaciones 10 a 19.

21. Producto de programa de ordenador, que se puede cargar en la memoria de al menos un ordenador y que incluye porciones de código de software para realizar las etapas del procedimiento según cualquiera de las reivindicaciones 1 a 9.

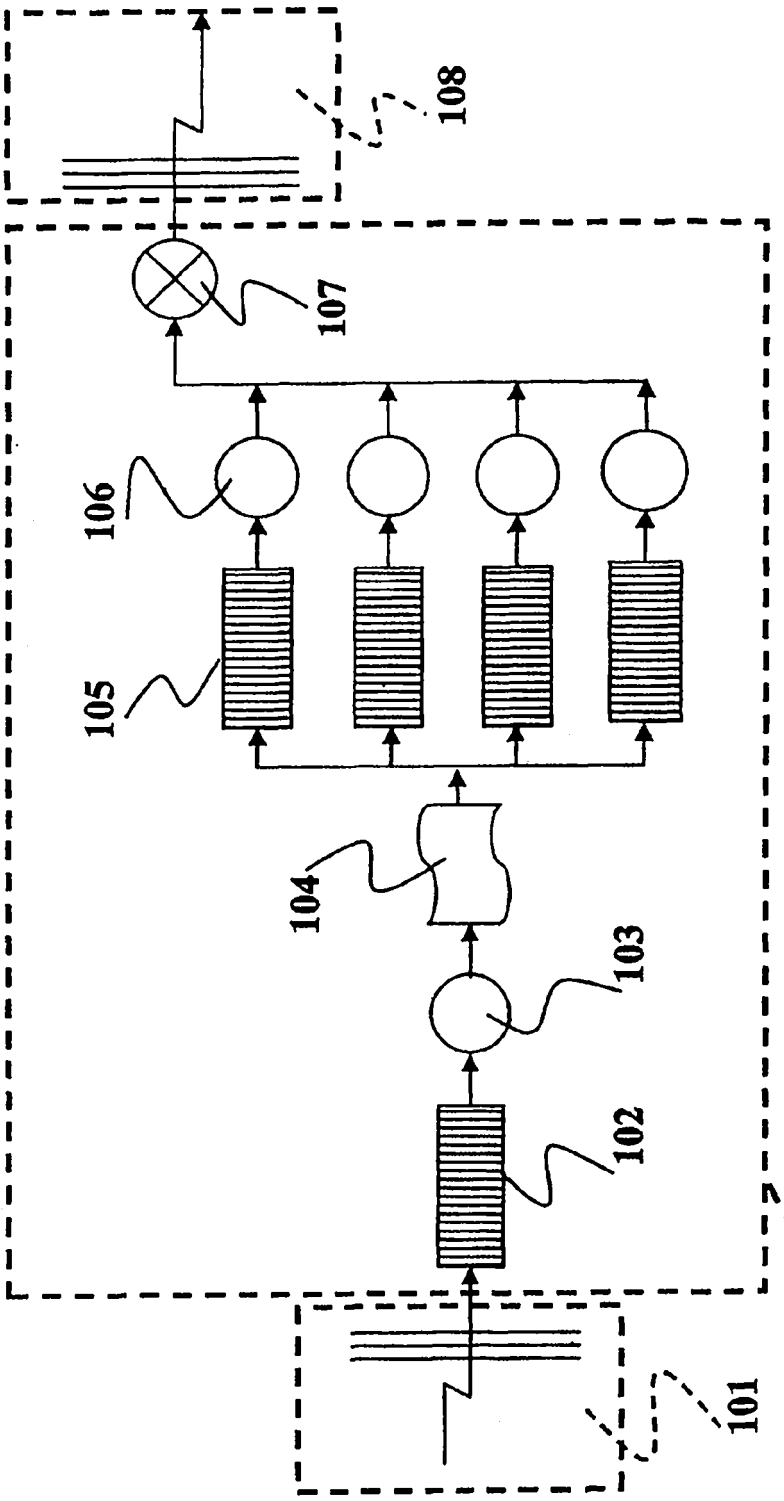

Fig. 1

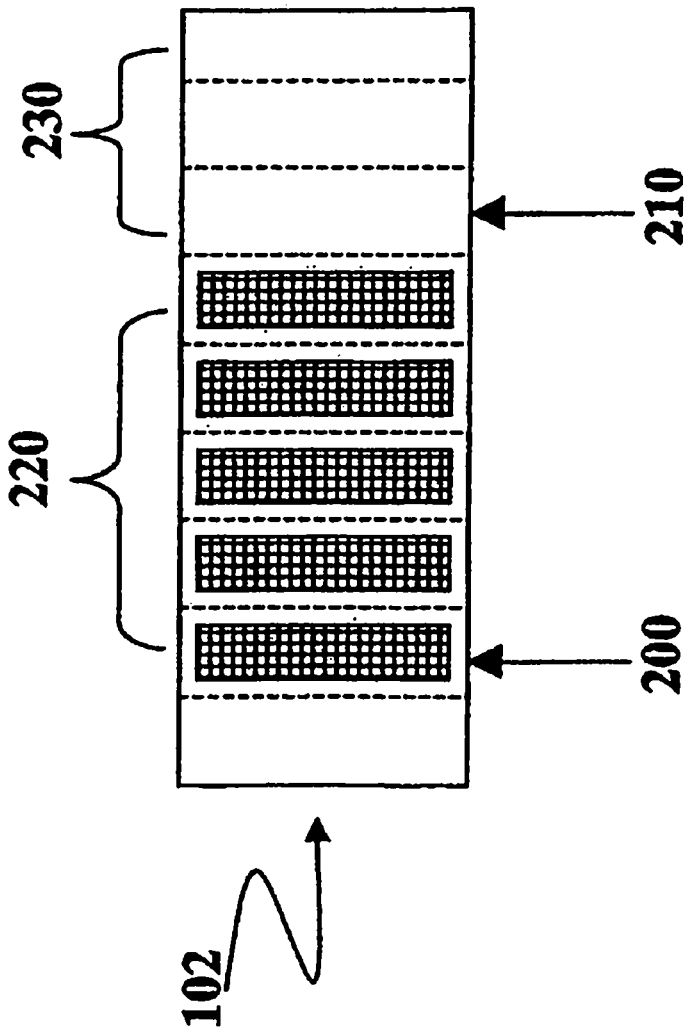

Fig. 2

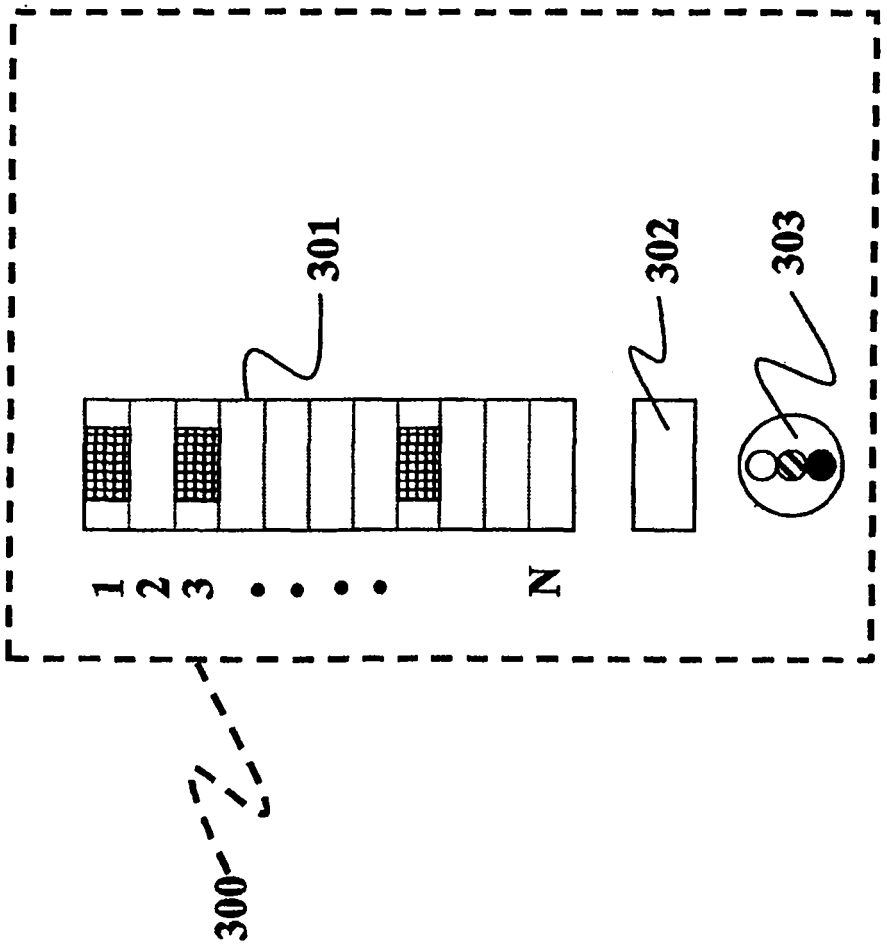

Fig. 3

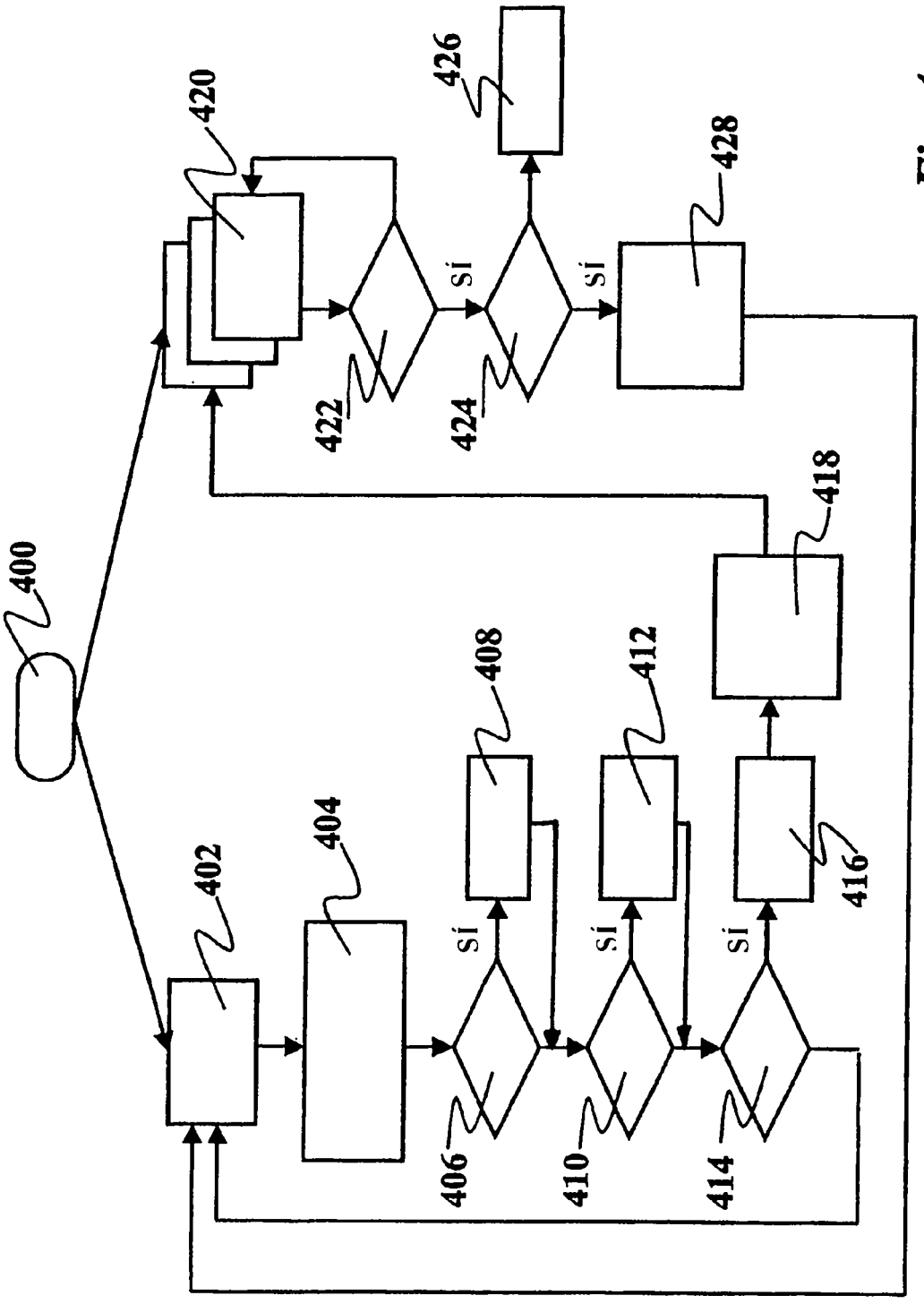
400
402
404
406
408
410
412
414
416
418
420
422
424
426
428
Sí
Sí
Sí
Sí
Sí

Fig. 4

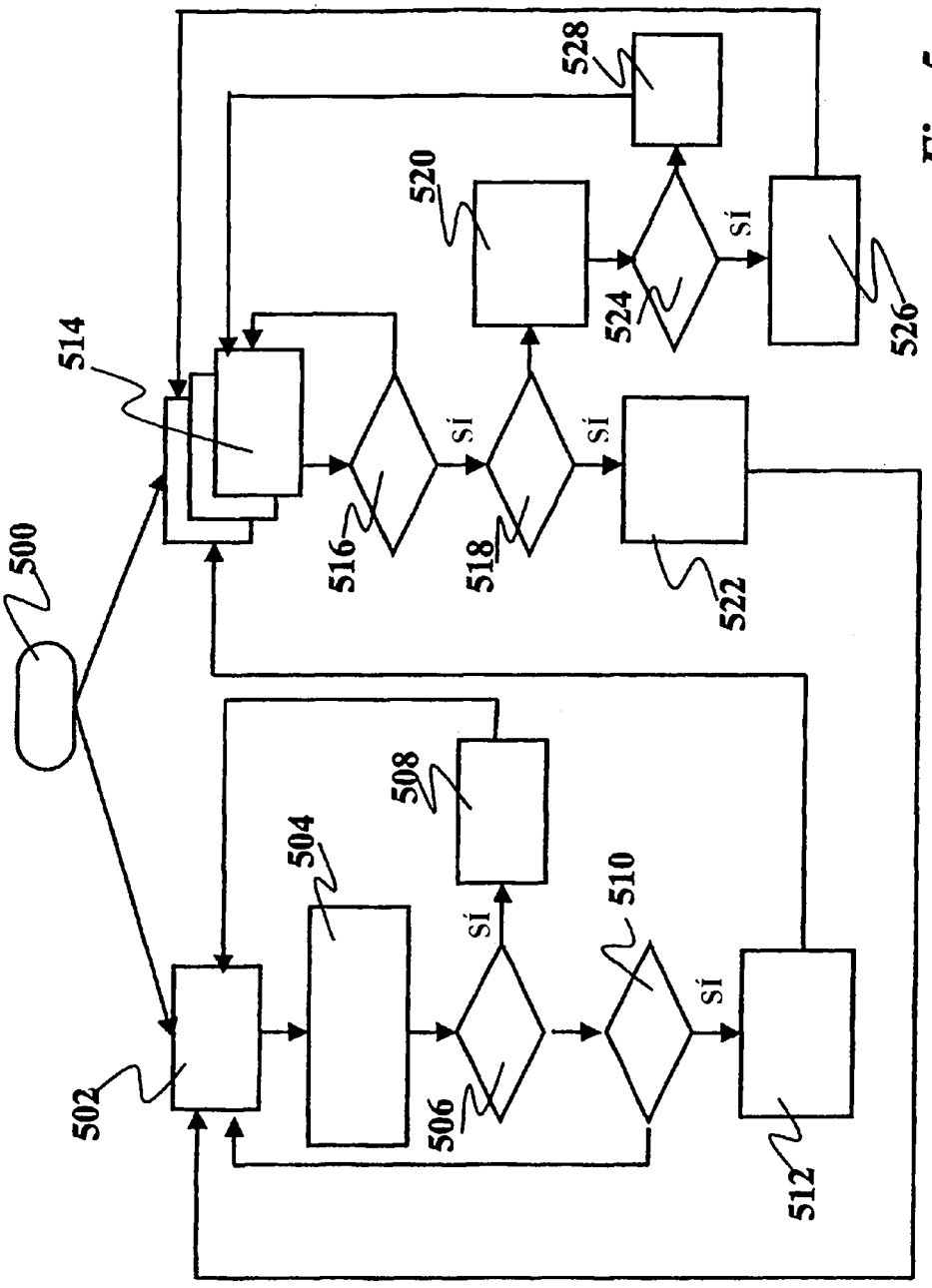
500
502
504
506
508
510
512
514
516
518
520
522
524
526
528
sí
sí
sí
sí
sí

Fig. 5