(19) RÉPUBLIQUE FRANÇAISE

INSTITUT NATIONAL
DE LA PROPRIÉTÉ INDUSTRIELLE

PARIS

(11) **N° de publication :** **2 792 431**
(à n'utiliser que pour les commandes de reproduction)

(21) **N° d'enregistrement national :** **99 04886**

(51) Int Cl[7] : **G 06 F 13/24**

(12) DEMANDE DE BREVET D'INVENTION A1

(22) **Date de dépôt :** 19.04.99.

(30) **Priorité :**

(43) **Date de mise à la disposition du public de la demande :** 20.10.00 Bulletin 00/42.

(56) **Liste des documents cités dans le rapport de recherche préliminaire :** *Se reporter à la fin du présent fascicule*

(60) **Références à d'autres documents nationaux apparentés :**

(71) **Demandeur(s) :** *CANON KABUSHIKI KAISHA* — JP.

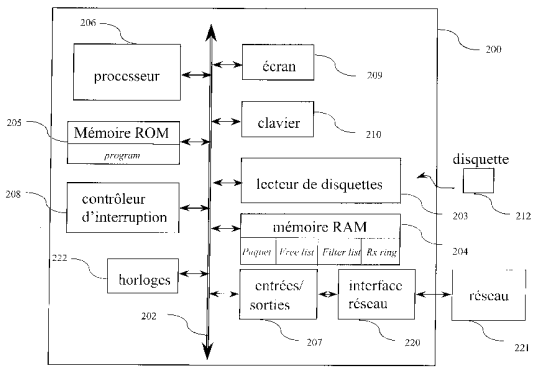
(72) **Inventeur(s) :** SANCHEZ LEIGHTON VICENTE, MAZE FREDERIC et DUPIN BENOIT.

(73) **Titulaire(s) :**

(74) **Mandataire(s) :** RINUY SANTARELLI.

(54) DISPOSITIF ET PROCEDE POUR FILTRER ET/OU DELIVRER DES PAQUETS DE DONNEES ENTRANTS A DES APPLICATIONS CLIENTES DANS DES ARCHITECTURES DE COMMUNICATION A ACCES DIRECT AU RESEAU.

(57) Procédé de filtrage de paquets de données entrants dans un système informatique fonctionnant en mode multitâche, en vue de déterminer chaque application à laquelle lesdits paquets de données sont destinés, chaque application s'exécutant sous le contrôle d'un ordonnanceur effectuant des étapes d'ordonnancement, caractérisé en ce que, pour chaque application susceptible de recevoir lesdits paquets de données, le procédé comporte un fil d'exécution de filtrage au cours duquel on détermine si lesdits paquets de données sont destinés, ou non, à ladite application, l'ordonnanceur contrôlant individuellement l'exécution de chacun des fils d'exécution de chaque application et l'exécution de chaque fil d'exécution de filtrage.



FR 2 792 431 - A1

La présente invention concerne un dispositif et un procédé pour filtrer et/ou délivrer des paquets de données entrants à des applications clientes dans des architectures de communication à accès direct au réseau.

La présente invention se rapporte en particulier à un mécanisme permettant le filtrage, le traitement de paquets de données, et la redirection de celles-ci vers et pour le compte d'au moins une application, l'application étant composée de son propre espace d'adressage mémoire et d'au moins un fil d'exécution (aussi appelé fil de contrôle ou « thread » en anglais).

Considérons une machine informatique (ou terminal) connectée par l'intermédiaire de cartes d'interface à des dispositifs véhiculant des flots d'informations sous forme de paquets de données. Par exemple de tels dispositifs pourraient être des dispositifs de communication du type Ethernet (marque déposée).

D'une manière générale, sur un tel dispositif, lorsqu'une information est lue ou reçue par la carte d'interface, celle-ci génère une interruption. Cette interruption est utilisée pour prévenir le processeur central du terminal de la présence d'une information afin qu'il puisse prendre les mesures nécessaires à sa réception (ou prise en compte). L'interruption peut provoquer une suspension temporaire du fil d'exécution en cours de traitement par le microprocesseur (CPU) qui exécute alors le sous-programme d'interruption (SPI) associé. L'exécution du sous-programme d'interruption ne peut être éventuellement interrompue avant son terme que par une interruption de priorité supérieure ou par une interruption non masquable (connue sous

l'acronyme "NMI" pour "Non-Maskable interrupt"). A la fin de l'exécution du SPI, le CPU peut reprendre son exécution normale après un éventuel réordonnancement des fils d'exécution présents dans le système.

Habituellement, les paquets de données ainsi reçus sont transmis à un protocole de communication (par exemple TCP/IP) qui peut procéder à un certain nombre de traitements sur les paquets (vérification de l'intégrité des données par vérification de totaux de contrôle (en anglais "checksums"), réassemblage de paquets de données, acquittement, ...). Parmi ces différentes tâches pouvant être effectuées par le protocole, il y en a une qui consiste à déterminer chaque destinataire (c'est à dire chaque application cliente) devant recevoir les paquets de données au sein du terminal. Cette tâche est aussi appelée démultiplexage.

Dans le cadre d'architecture de communication par accès direct au réseau, on cherche à permettre aux applications clientes de recevoir les informations circulant sur le réseau au plus près du matériel afin qu'elles puissent elles-mêmes implémenter leur propre protocole de communication. A cette fin, à la réception d'un paquet de données par le terminal, il est nécessaire de procéder à son démultiplexage juste au dessus du matériel et avant traitement par un éventuel protocole. Classiquement, cette opération requiert alors l'utilisation de filtres de paquets. Les filtres de paquets sont des petits morceaux de code installés dans le système par des applications ou générés par le système à partir des informations fournies par les applications.

De multiples filtres de paquet existent dans la littérature technique.

Par exemple, dans l'article "*DPF: Fast, Flexible Message Demultiplexing using Dynamic Code Generation*" de Dawson R. Engler et M. Frans Kaashoek (dans les "*Proceedings of SIGCOMM'96 Conference*"), il est décrit une méthode de filtrage de paquets, rapide et flexible, qui consiste à compiler les différents filtres en code machine directement compréhensible par le processeur. Chaque filtre est initialement organisé en structure arborescente. Des méthodes de parcours d'arbres sont utilisées afin de réduire la durée d'exécution des filtres et des méthodes de factorisation sont utilisées afin d'intégrer au mieux les différents filtres. Cependant, si la factorisation des filtres

est rendue impossible du fait de leur complexité ou de leur trop grande différence, ceux-ci sont exécutés de manière séquentielle.

Ce type de filtrage comporte un certain nombre de défauts. S'il prend place au sein d'un traitant d'interruption (aussi appelé "SPI" pour sous-programme d'interruption), il n'est pas possible de déterminer de façon précise combien de temps va durer l'exécution du SPI. Par exemple, si l'opération de filtrage comporte la comparaison de l'adresse de destination contenue dans le paquet de données et de l'adresse d'une ou plusieurs applications, même avec une structure arborescente, même modifiable dynamiquement, la durée d'exécution du filtre dépend du nombre de comparaisons à effectuer.

Si la première comparaison est positive, la durée du SPI sera courte, si c'est la dernière des comparaison qui est positive, le SPI durera plus longtemps.

Une fois que le SPI est lancé, l'ordonnanceur n'a plus aucun contrôle sur son exécution. On rappelle que l'ordonnanceur ("scheduler" en anglais) est un mécanisme qui contrôle l'exécution des différents fils d'exécution présents au sein d'un système, à l'exception des interruptions. L'ordonnanceur détermine notamment le prochain fil d'exécution à traiter par le processeur de manière à respecter une politique d'ordonnancement donnée. Du fait que l'ordonnanceur n'a pas de contrôle sur l'exécution de l'interruption, une qualité de service, connue sous l'acronyme "QOS" prenant en compte les filtres de paquets ne peut être envisagée. Il n'existe alors pas de moyen pour contrôler l'exécution des filtres les uns par rapport aux autres ou encore d'obtenir un retour d'information (en anglais, " feedback ") de la part de ces filtres pour adapter le contrôle exercé sur leur exécution par l'ordonnanceur et pouvant être intégré dans la mise en œuvre d'une qualité de service.

Plus généralement, si les filtres s'exécutent dans le cadre d'un SPI, il n'est pas possible de les prendre en compte dans le cadre de quelconques règles d'ordonnancement.

Dans le document US 5,469,571 est décrit une architecture logicielle de traitement d'interruption palliant le problème mentionné ci-dessus concernant la durée d'exécution du SPI. L'invention consiste à réduire la taille

du SPI au minimum en déléguant au maximum le traitement de l'interruption à un et un seul fil d'exécution sous contrôle de l'ordonnanceur avec une priorité adéquate. Si cette méthode permet de contrôler le temps d'exécution du SPI, elle ne permet, en revanche, pas de contrôler l'exécution des filtres les uns par rapport aux autres.

D'une manière générale, l'invention vise à pallier ces inconvénients en utilisant dans un dispositif de communication un mécanisme de filtrage comportant un premier sous-programme d'interruption activant au moins un fil d'exécution (dit de filtrage) dans le système. Chaque fil d'exécution activé filtre et traite les paquets de données relatifs à la ou aux applications qui lui sont associées.

Selon un premier aspect, la présente invention vise un procédé de filtrage de paquets de données entrants dans un système informatique fonctionnant en mode multitâche, en vue de déterminer chaque application à laquelle lesdits paquets de données sont destinés, chaque application s'exécutant sous le contrôle d'un ordonnanceur effectuant des étapes d'ordonnancement, caractérisé en ce que, pour chaque application susceptible de recevoir lesdits paquets de données, il comporte un fil d'exécution de filtrage au cours duquel on détermine si lesdits paquets de données sont destinés, ou non, à ladite application, l'ordonnanceur contrôlant individuellement l'exécution de chacun des fils d'exécution de chaque application et l'exécution de chaque fil d'exécution de filtrage.

Grâce à ces dispositions, l'ordonnanceur impute une durée d'exécution à chaque filtre. En particulier, il impute une durée d'exécution au fil d'exécution de filtrage conjointement à la durée imputée aux fils d'exécution de l'application, ladite application étant associée audit fil d'exécution de filtrage et, ainsi, applique aux fils d'exécution de filtrage les mêmes contraintes qu'aux applications auxquelles ils sont associés, par exemple dans le cadre d'une qualité de service ("QoS").

Ainsi, on peut ordonnancer les fils d'exécution de filtrage et donc les filtres comme les fils d'exécution des applications et appliquer une organisation dynamique de l'ordre d'exécution des filtres.

L'utilisation de plusieurs fils d'exécution contrôlés par l'ordonnanceur autorise un meilleur contrôle du temps d'exécution de chacun des filtres.

Du fait que l'ordonnanceur élit le filtre devant être traité selon des critères respectant une politique globale d'utilisation des ressources, par exemple d'une qualité de service :

- l'invention permet de faire respecter une politique de gestion de ressources sur les réseaux (ordonnancement, ordonnancement avec qualité de service) par application (ou " client ") et ce jusqu'à un niveau le plus bas (incluant les filtres de paquets) ;
- l'invention permet de modifier facilement et dynamiquement l'ordre d'exécution des filtres ;
- le SPI a une durée fixe et courte ;
- les filtres ont une durée variable mais contrôlée par l'ordonnanceur ; et
- l'invention facilite l'introduction ou le retrait de nouveau filtres avec des applications associées.

D'une manière générale, la mise en œuvre de la présente invention limite la durée d'exécution de chaque interruption correspondant à une entrée de paquets de données puisque cette interruption n'a pas en charge le filtrage des paquets de données entrants, celui-ci étant exécuter sous le contrôle de l'ordonnanceur.

Selon des caractéristiques particulières, chaque fil d'exécution de filtrage est associé à au moins une application.

Ainsi, certains fils d'exécution de filtrage seront associés à une seule application alors que d'autres seront associés à plusieurs applications, grâce à la mise en œuvre de "factorisation" (connues de l'homme du métier) de plusieurs opérations de filtrage (filtres).

Selon d'autres caractéristiques particulières, au cours desdites étapes d'ordonnancement, on ordonne consécutivement, d'une part, un fil d'exécution de filtrage avec, d'autre part, au moins un fil d'exécution d'une application à laquelle ledit fil d'exécution de filtrage est associé.

Ainsi, des paquets de données qui seraient déterminés comme destinés à une application peuvent être mis en œuvre immédiatement par un fil d'exécution de ladite application qui est mis en œuvre consécutivement.

Selon d'autres caractéristiques particulières, au cours desdites étapes d'ordonnancement, on met en œuvre une qualité de service.

Ainsi, la présente invention qui, de manière générale, limite la durée d'exécution de chaque interruption correspondant à une entrée de paquets de données, garantit une gêne minimale de la qualité de service due à ces interruptions.

Selon d'autres caractéristiques particulières, au cours desdites étapes d'ordonnancement, on décompte le temps de traitement par le processeur du fil d'exécution de filtrage du temps de traitement alloué à l'ensemble des fils d'exécution de l'application à laquelle ledit fil d'exécution de filtrage est associé.

Grâce à ces dispositions, chaque application peut se voir garantir la mise à disposition d'une durée d'exécution indépendante de la durée de chaque fil d'exécution de filtrage.

Selon un deuxième aspect, la présente invention vise un dispositif de filtrage de paquets de données entrants dans un système informatique fonctionnant en mode multitâche, en vue de déterminer chaque application à laquelle lesdits paquets de données sont destinés, chacun des fils d'exécution de chaque application s'exécutant sous le contrôle d'un moyen d'ordonnancement, caractérisé en ce que, pour chaque application susceptible de recevoir lesdits paquets de données, il comporte un moyen de filtrage adapté à déterminer si lesdits paquets de données sont destinés, ou non, à ladite application, le moyen d'ordonnancement contrôlant individuellement chacun des fils d'exécution de chaque application et chaque moyen de filtrage.

L'invention vise aussi un ordinateur, une caméra, un télécopieur, un appareil photographique, un téléviseur, une imprimante et un scanner, caractérisés en ce qu'ils comportent un dispositif tel que succinctement exposé ci-dessus.

L'invention vise aussi :

- un moyen de stockage d'informations lisible par un ordinateur ou un microprocesseur conservant des instructions d'un programme informatique caractérisé en ce qu'il permet la mise en œuvre du procédé de l'invention telle que succinctement exposée ci-dessus, et

- un moyen de stockage d'informations amovible, partiellement ou totalement, et lisible par un ordinateur ou un microprocesseur conservant des instructions d'un programme informatique caractérisé en ce qu'il permet la mise en œuvre du procédé de l'invention telle que succinctement exposée ci-dessus.

Les caractéristiques préférentielles ou particulières, et les avantages de ce dispositif, de cet ordinateur, de cette caméra, de ce télécopieur, de cet appareil photographique, de ce téléviseur, de cette imprimante, de ce scanner et de ces moyens de stockage d'information étant identiques à ceux du procédé tel que succinctement exposé ci-dessus, ces avantages ne sont pas rappelés ici.

D'autres avantages, buts et caractéristiques de la présente invention ressortiront de la description qui suit, faite en regard des dessins annexés, dans lesquels :

- la figure 1 représente, de manière schématique, les différences d'ordonnancement des fils d'exécution sur le processeur entre deux procédés de l'art antérieur et le procédé conforme à la présente invention ;

- la figure 2 représente, de manière schématique, un dispositif conforme à la présente invention,

- la figure 3 représente, de manière schématique, un exemple d'ordonnancement des fils d'exécution sur le processeur intégrant les fils d'exécution de filtrage, conformément au procédé objet de la présente invention,

- la figure 4 représente, de manière schématique, un exemple d'algorithme d'exécution d'un fil d'exécution de filtrage mis en œuvre dans la présente invention ; et

- la figure 5 représente, de manière schématique, la gestion des paquets de données à filtrer et à délivrer, conformément au procédé objet de la présente invention.

En haut de la figure 1, on observe que, conformément à l'art antérieur bien connu de l'homme du métier, un ordonnanceur provoque successivement l'exécution de différents fils d'exécution (ici seuls les fils d'exécution T1 et T2 sont représentés et leur exécution est symbolisée par une ligne verticale, tandis que le passage d'un fil d'exécution à un autre est représenté par une ligne horizontale). Conformément à cet art antérieur, l'ordonnanceur fait d'abord s'exécuter un fil d'exécution T2, puis, ensuite, un fil d'exécution T1.

On suppose qu'au cours du traitement par le processeur du fil d'exécution T1, une entrée de paquets de données dans le dispositif (voir figure 2) provoquent le déclenchement d'une interruption IT.

Les paquets de données entrants dans le dispositif sont ainsi signalés par le déclenchement d'une interruption qui provoque :

- la suspension par l'unité centrale du fil d'exécution en cours de traitement et

- l'exécution du sous-programme d'interruption associé à l'interruption.

Le sous-programme d'interruption contient, d'une part, la gestion du mécanisme matériel d'interruption (notamment son acquittement) et, d'autre part, le traitement spécifique associé à l'événement signalé par l'interruption, par exemple le filtrage des paquets de données, filtrage nécessaire à la délivrance des paquets entrants à l'application cliente à laquelle ces paquets de données sont destinées. Ce sous-programme d'interruption ne peut pas être interrompu par l'ordonnanceur et sa durée ne peut être contrôlée.

A la fin de l'interruption, le traitement par le processeur du fil d'exécution T1 est poursuivie.

Dans l'exemple représenté, si le fil d'exécution T1 est une tâche critique et que les paquets de données entrants sont destinés à être traités par le fil d'exécution T2, qui n'est pas critique, le fil d'exécution T1 a été,

conformément à l'art antérieur, exagérément perturbé par l'arrivée des paquets de données destinés au fil d'exécution T2.

Dans la partie centrale de la figure 1, on a représenté le séquencement des exécutions de fils d'exécution et d'interruption conformément à l'enseignement du document US 5,469,571. Ce document enseigne que, quand l'arrivée de nouveaux paquets de données provoque une interruption, le sous-programme d'interruption se limite à la gestion du mécanisme matériel d'interruption. Le traitement de l'événement ainsi signalé est délégué à un autre fil d'exécution possédant une priorité d'exécution adaptée, sous le contrôle de l'ordonnanceur. Dans le domaine concerné par ce document, le traitement de l'événement consiste notamment en un filtrage des paquets de données entrants.

En figure 1 (partie centrale), on a représenté l'exécution successive d'un fil d'exécution T2, puis d'un fil d'exécution T1, sous le contrôle de l'ordonnanceur, avant que l'arrivée de paquets de données provoque une interruption. Seul un contrôle (avec acquittement) d'interruption est effectué, à l'exclusion de tout filtrage des paquets de données.

Ce filtrage est effectué par un fil d'exécution spécifique T3.

Ainsi, très rapidement, l'interruption est achevée et le fil d'exécution T1 reprend son exécution. Ensuite, l'ordonnanceur provoque l'exécution du fil d'exécution T3 et au moins le début du filtrage des paquets de données entrants.

On observe que cet art antérieur n'est pas appliqué aux dispositifs à accès direct au réseau, pour lesquels le démultiplexage doit être fait le plus tôt possible car chaque application contient son protocole de communication.

L'invention s'applique préférentiellement aux cas des dispositifs à accès direct au réseau. Dans ce type de dispositif, le démultiplexage ne peut plus être assuré par le protocole de communication puisque celui-ci se situe alors dans l'application. Avant que les paquets de données puissent être traités, par le protocole, il est donc nécessaire que le destinataire desdits paquets de données soit clairement identifié.

En effet, dans une architecture traditionnelle, le démultiplexage à l'arrivée d'un paquet se limite à identifier la première couche de protocole devant traiter le paquet (par exemple dans une architecture traditionnelle Ethernet/TCP/IP, le démultiplexage à l'arrivée d'un nouveau paquet détermine s'il faut passer le paquet à la couche de protocole IP ou ARP en consultant un marqueur précis dans l'en-tête de protocole dans le paquet). Dans une telle architecture, l'application destinataire n'est connue qu'une fois que le paquet a traversé l'ensemble des couches de protocole de communication.

Au contraire, dans une architecture à accès direct, en l'absence d'un marqueur spécifique identifiant de manière unique l'application, l'opération de démultiplexage à l'arrivée d'un paquet est plus complexe et potentiellement plus longue car elle peut nécessiter le traitement de l'ensemble de l'en-tête de protocole dès la réception du paquet. En conclusion, dans cette architecture, la fonction de démultiplexage est déplacée du protocole de communication vers les filtres de paquets.

En bas de la figure 1, on a représenté le séquencement des exécutions de fils d'exécution et d'interruption conformément à la présente invention : un fil d'exécution T2, puis un fil d'exécution T1 s'exécutant successivement, sous le contrôle de l'ordonnanceur, lorsque l'arrivée de paquets de données provoque une interruption, seul un contrôle (avec acquittement) d'interruption est effectué, à l'exclusion de tout filtrage des paquets de données.

Ce filtrage est effectué par autant de fils d'exécution qu'il y a de filtres à appliquer.

On observe ici que l'on peut décomposer la gestion d'une interruption matérielle en deux parties. Une première partie consiste à traiter l'interruption au niveau du matériel et une seconde partie consiste à traiter l'interruption au niveau du système informatique (exécution du logiciel chargé de traiter l'événement signalé par l'interruption). Lorsqu'une interruption survient, causée ici par l'arrivée des paquets de données entrants, le traitement de l'interruption se limite à la gestion du contrôleur d'interruption (composant ("chip" en anglais) sur la carte mère) et acquitte l'interruption au niveau de ce contrôleur.

Le traitement de l'événement proprement dit est relégué à un fil d'exécution contrôlable par l'ordonnanceur.

Ainsi, très rapidement, la suspension du fil d'exécution en cours, est achevée, et ce fil d'exécution, ici le fil d'exécution T1 reprend son exécution. Ensuite, l'ordonnanceur provoque l'exécution des fils d'exécution T3 et T4, qui correspondent au filtrage des données destinées aux applications clientes qui comportent respectivement les fils d'exécution T1 et T2.

Ainsi, grâce à la mise en œuvre de la présente invention, toute la souplesse de la programmation multitâche s'applique aussi à la programmation des filtres. En outre, lorsque l'ordonnanceur met en œuvre une qualité de service, chacun des fils d'exécution qui comporte un filtre associé à une application (par exemple une application de traitement de données), est géré de la même manière que les fils d'exécution d'une application et est contrôlé en accord avec les règles d'ordonnancement ("scheduling policy" en anglais) du système, notamment en ce qui concerne sa durée d'exécution.

La figure 2 représente le circuit électronique 200 d'un dispositif de communication connecté à un réseau 221 de type Ethernet, par exemple.

Ce dispositif de communication peut être intégré dans un équipement de type ordinateur, imprimante, scanner, télécopieur (fax), appareil photographique, caméra. Le circuit électronique 200 comporte, reliés entre eux par une architecture de communication 202 (par exemple un bus d'adresses et de données) :

- des horloges 222,
- un contrôleur d'entrée/sortie 207, relié, par l'intermédiaire d'une interface réseau 220 (comportant, ici le périphérique réseau dit "Fast Ethernet 10/100 “TULIP” DEC21140A"), au réseau 221,
- un contrôleur d'interruption 208,
- une unité centrale de traitement 206,
- un lecteur de disquette 203, adapté à lire et à écrire sur des disquettes 212,
- un écran 209,
- un clavier 210,

- une mémoire morte ROM 205, et

- une mémoire vive RAM 204.

L'interface réseau 220 gère l'interface avec le réseau externe 221, par exemple de type Réseau Téléphonique Commuté.

Tous les composants illustrés en figure 2 sont bien connus de l'homme du métier des circuits de communication à processeur et plus généralement des circuits de traitement de l'information. Ils ne sont donc pas détaillés ici.

L'unité centrale 206 est, en outre, adaptée à mettre en œuvre le procédé de l'invention.

La mémoire vive 204 conserve, dans des registres qui, par commodité, portent chacun le même nom que les données qu'il contient :

- un registre "*paquet*" dans lequel sont conservés les paquets à filtrer ;

- un registre "*Rx ring*" dans lequel est conservée une table (ou une liste) de descripteurs de réception (Le pilote de la carte réseau place dans ces descripteurs les informations sur les blocs de mémoire tampon (ou "buffers" selon le terme anglais communément utilisé par l'homme du métier) de réception présents dans la mémoire hôte) ;

- un registre "*filter list*" dans lequel sont représentés des blocs de mémoire tampon ("buffer") contenant les paquets reçus dans l'attente d'être traités par les différents fils d'exécution de filtrage en activité ; et

- un registre "*free list*" dans lequel sont représentées des zones mémoire libres à allouer statiquement ou dynamiquement.

La mémoire morte 205 conserve, dans des registres qui, par commodité portent, chacun, le même nom que les données qu'il contient :

- un registre "*program*" dans lequel est conservé le programme de fonctionnement de l'unité centrale 206.

La mémoire morte 205 constitue un moyen de stockage d'informations lisibles par un ordinateur ou un microprocesseur, conservant des instructions d'un programme informatique caractérisé en ce qu'il permet la mise en œuvre du procédé de l'invention. Selon une variante, la mémoire morte 205

est amovible, partiellement ou totalement, et comporte, par exemple, une bande magnétique, une mémoire flash, une disquette ou un compact disque à mémoire figée ("CD-ROM" en anglais).

On observe, en figure 3, des événements en relation avec (dans l'ordre de haut en bas) :

- chaque interruption IT provoquée par des paquets de données entrants en provenance du réseau 221 (sous forme d'impulsions) ;

- une sortie d'ordonnanceur (sous forme de passage à un état haut) provoquant le changement du fil d'exécution en cours d'exécution sur le processeur ;

- l'exécution d'un sous-programme d'interruption (sous forme du passage à un état haut) ;

- l'exécution d'un fil d'exécution de filtrage 11 pour filtrer les paquets de données destinés à une application cliente dénommée tâche 1 (sous forme du passage à un état haut) ;

- l'exécution d'un fil d'exécution de filtrage 12 pour filtrer les paquets de données destinés à une application cliente dénommée tâche 2 (sous forme du passage à un état haut) ;

- l'exécution d'un autre fil d'exécution indépendant (sous forme du passage à un état haut).

En figure 3, on observe que se succèdent :

- à l'instant *t1*, le processeur 206 est interrompu par l'arrivée d'un nouveau paquet de données en provenance du réseau 221 ;

- le processeur 206 exécute le sous-programme correspondant à l'interruption, SPI, entre l'instant *t1* et l'instant *t2* ;

- à l'instant *t2*, le SPI est terminé.

On observe que les fils d'exécution de filtrage 11 et 12 liés aux applications dénommées respectivement tâches 1 et 2 (non représentées) ont été réveillés par le SPI. Ils sont alors éligibles par l'ordonnanceur pour être exécutés par le processeur ; à l'instant *t2*, l'ordonnanceur décide de reprendre l'exécution du fil d'exécution de filtrage 11 ;

- à l'instant *t3*, l'ordonnanceur décide d'interrompre l'exécution du fil d'exécution de filtrage 11 car ce fil d'exécution de filtrage 11 a dépassé son quantum de temps. Le fil d'exécution de filtrage 12 est alors exécuté, sous le contrôle de l'ordonnanceur ;

- à l'instant *t4*, le fil d'exécution de filtrage 12 décide de rendre la main (par exemple parce qu'il n'y a plus de paquet à filtrer) ;

- à l'instant *t5*, le processeur 206 est interrompu par l'arrivée d'un nouveau paquet. Le processeur 206 exécute alors le sous-programme correspondant à l'interruption ;

- à l'instant *t6*, le SPI est terminé et les fils d'exécution de filtrage 11 et 12 sont éligibles par l'ordonnanceur. L'ordonnanceur décide cette fois-ci d'exécuter d'abord le fil d'exécution de filtrage 12 en accord avec sa stratégie de qualité de service ;

- à l'instant *t7*, le fil d'exécution de filtrage 12 rend la main et l'ordonnanceur décide d'exécuter le fil d'exécution de filtrage 11 ; et

- à l'instant *t8*, la tâche de filtrage 11 rend la main.

Un fil d'exécution de filtrage est dit éligible quand il fait partie des fils d'exécution que l'ordonnanceur peut choisir lors d'une décision de réordonnancement. Dans l'implémentation décrite dans la description, quand le fil d'exécution de filtrage n'a pas de paquets de données à filtrer, il est bloqué sur un sémaphore, ce qui a pour effet de le retirer de l'ensemble des fils d'exécution pouvant être choisis par l'ordonnanceur lors d'une décision de réordonnancement. Quand un nouveau paquet arrive, le traitement d'interruption signale le sémaphore, ce qui a pour effet de réveiller le fil d'exécution concerné par ce sémaphore et ce fil d'exécution peut alors être choisi au cours d'une décision de réordonnancement.

L'ordonnanceur peut alors lancer l'exécution du fil d'exécution de filtrage. Dès qu'un paquet de données survient, tous les fils d'exécution de filtrage sont immédiatement éligibles et dès qu'un filtre d'un fil d'exécution de filtrage a filtré tous les paquets de données entrants, le fil d'exécution de filtrage devient inéligible.

Les structures de données utilisées par le procédé et le dispositif objets de la présente invention sont représentés en regard de la figure 5. Elles comportent :

- un groupe de blocs de mémoire tampon 501 ("pool de buffers" en anglais : on rappelle qu'un buffer est une zone mémoire allouée statiquement ou dynamiquement dans laquelle sont stockées des données) de réception ("*free_list*"). Une partie des blocs de mémoire tampon sont alloués lors de l'initialisation de la carte réseau, ils forment le groupe (pool) de base. Ce groupe peut être complété par des blocs de mémoire tampon supplémentaires à l'ouverture d'une connexion entre une application et la carte. Ces blocs de mémoire tampon supplémentaires seront désalloués à la fermeture de la connexion.

- une liste de filtrage 502 ("*filter_list*"). Les blocs de mémoire tampon contenant les paquets de données reçus sont placés dans cette liste 502 dans l'attente d'être traités par les différents fils d'exécution de filtrage en activité. Ces blocs de mémoire tampon sont organisés en liste doublement chaînée.

Dans le mode de réalisation décrit et représenté, comportant le périphérique réseau Fast Ethernet 10/100 "TULIP" DEC21140A, les paquets de données reçus à partir du réseau sont placés directement dans la mémoire de la machine hôte. A cet effet, ce périphérique utilise des accès direct en mémoire.

Ce périphérique possède un registre pointant sur une table (ou une liste) de descripteurs de réception 503 (cette table (ou liste) est aussi appelée "*Rx ring*"). Le pilote de la carte réseau (pilote connu sous le nom anglais "driver") place dans ces descripteurs les informations sur les blocs de mémoire tampon de réception présents dans la mémoire hôte (pointeurs sur les blocs de mémoire tampon de données, taille de ces blocs de mémoire tampon, information de contrôle spécifiant certaines des caractéristiques de la réception).

La carte réseau 220 parcourt cette table (ou liste) de manière circulaire en ordre premier entré, premier sorti (connu sous le nom de "FIFO").

Le bit “OWN” du descripteur indique s’il est la propriété de la carte ou du pilote ("driver" en anglais).

A l’arrivée d’un nouveau paquet, la carte réseau 220 cherche à acquérir le descripteur de réception suivant. S’il est la propriété du pilote ("driver"), le paquet est rejeté. Si le descripteur est libre, la carte réseau 220 place le paquet reçu dans le(s) bloc(s) de mémoire tampon de réception pointés par le descripteur, positionne le statut de réception du descripteur et génère une interruption.

Le mécanisme se décompose alors en trois étapes distinctes :

A/ traitement de l’interruption : Le traitant d’interruption transfère les paquets reçus dans la liste de filtrage et réveille les fils d’exécution de filtrages qui ne sont pas déjà réveillés. Au niveau de la liste de réception ("ring") de la carte, *Rx_ring*, le traitant d'interruption remplace le pointeur sur le paquet reçu par un pointeur sur un nouveau bloc de mémoire tampon de réception qu’il a préalablement alloué dans la liste des blocs de mémoire tampon disponibles. Le traitant d'interruption nettoie la liste de filtrage (c’est-à-dire qu’il transfert les paquets traités par les filtres dans la liste des blocs de mémoire tampon disponibles).

B/ filtrage : Les fils d’exécution de filtrage actifs parcourent la liste de filtrage. En appliquant leur filtre sur chacun des paquets reçus, ils déterminent si l’application qui leur est associée est destinataire du paquet.

C/ traitement du paquet reçu : Si l’application ou les applications associées au filtre sont destinataires dudit paquet alors le fil d’exécution peut exécuter tout type de traitement adéquats : il peut s’agir d’exporter les paquets de données vers la ou les applications, de l’envoi d’un acquittement, de la vérification de l’intégrité des paquets de données...

En variante, selon le type de traitement, les phases B et C peuvent être combinées avantageusement pour tirer profit du parcourt des paquets de données réalisé par le filtre. Cette variante s'applique, par exemple, au calcul d’un total de contrôle (“ checksum ” en anglais).

Les fils d’exécution de filtrage n’ayant pas de paquet à traiter (i.e. fil d’exécution en bout de liste de filtrage ou liste de filtrage vide) se mettent en

sommeil sur un sémaphore (méthode préférée), ceci afin de ne pas consommer de temps CPU en effectuant une tâche d'"attente active" (également connue de l'homme du métier sous le nom de "polling actif").

On rappelle ici qu'un "polling" est une action de scruter une ou plusieurs information afin d'en identifier un changement d'état. On parle de "polling actif" lorsqu'on scrute une information de manière continue (dans une boucle infinie par exemple). Les fils d'exécution en attente (ou en "sommeil") sur un sémaphore ne sont plus éligibles par l'ordonnanceur. A l'arrivée d'un nouveau paquet, l'interruption va "réveiller" les fils d'exécution en attente (ou en "sommeil") en signalant leur sémaphore. Les fils d'exécution redeviennent alors éligibles par l'ordonnanceur pour être exécutés.

Si les paquets de données reçus par la carte réseau 220 ne sont pas susceptibles d'être reçus par plusieurs filtres, dès qu'un paquet de données est filtré et traité, il est placé directement dans la liste des blocs de mémoire tampon libres 501.

Si les paquets de données reçus par la carte réseau 220 sont susceptibles d'être reçus par plusieurs filtres, ils restent disponibles dans la liste de filtrage 502 jusqu'à ce que le traitant d'interruption les place dans la liste des blocs de mémoire tampon libres 501. A cet effet, le traitant d'interruption remonte la liste de filtrage à partir du début jusqu'au premier bloc de mémoire tampon en cours de traitement par un fil d'exécution de filtrage (un tel bloc de mémoire tampon est signalé par un champ spécifique). Chacun des paquets de données rencontrés pendant le parcours et qui ne possède pas ce champ spécifique est placé dans la liste des blocs de mémoire tampon libres 501.

Concernant la gestion des blocs de mémoire tampon de réception au niveau de la carte réseau 220, il est clair que différentes stratégies peuvent être utilisées selon le mode de transfert des paquets de données du filtre vers l'application.

Selon un premier exemple, un paquet de données est retiré de la liste de filtrage lorsque tous les fils d'exécution de filtrages l'ont traité. Selon un autre exemple, un paquet de données est retiré de la liste de filtrage au bout d'une durée prédéterminée. Selon un troisième exemple, un paquet de données

est retiré de la liste de filtrage dès qu'un fil d'exécution de filtrage a déterminé qu'il était destiné à une application.

En ce qui concerne l'initialisation d'une connexion, à l'ouverture d'une connexion, l'application passe un " contrat " avec le gestionnaire de qualité de service. Si le contrat est accepté par le gestionnaire de qualité de service, une connexion directe entre l'application et le driver est mise en place par ce dernier. Celui-ci crée un nouveau fil d'exécution dans le système chargé du filtrage et/ou du traitement des paquets de données destinés à cette application. Le code de ce fil d'exécution peut provenir :

- soit du système de réception qui peut tenir à la disposition de l'application un ensemble de codes pré-programmés et paramétrables,
- soit de l'application elle-même par des mécanismes de génération ou de chargement dynamique de code,
- soit par une combinaison des deux mécanismes précédents (par exemple le filtre est fourni par l'application alors que le code de transfert des paquets de données vers l'application est fourni par le driver),
- et/ou le code peut également résider dans l'espace d'adressage de l'application et être accédé par des appels inter espaces (aussi appelés " upcalls ").

Ce nouveau fil d'exécution de filtrage se positionne à la fin de la liste de filtrage (il n'est pas a priori concerné par les paquets déjà présent dans la liste).

La figure 4 présente un exemple de fil d'exécution de filtrage.

Ce fil d'exécution de filtrage comporte, tout d'abord, une opération d'initialisation 401, de type connu. Puis, à chaque fois que le fil d'exécution est choisi par l'ordonnanceur, au cours d'un test 402, l'unité centrale 206 détermine s'il y a un paquet de données à filtrer par ledit fil d'exécution, ou non, en lisant une information dans l'en-tête de chaque paquet, sachant que la liste étant chaînée, cet en-tête contient un identificateur du paquet de données suivant dans la liste, lorsqu'il y en a un.

Lorsque le résultat du test 402 est négatif, le fil d'exécution rend la main et l'ordonnanceur provoque un changement de fil d'exécution sur le

processeur, opération 406. Lorsque le résultat du test 402 est positif, une opération de filtrage 403 des données du paquet est effectuée. Au cours de cette opération, de manière connue, différents éléments du paquet de données sont comparés à des valeurs prédéterminées qui sont liées à une application cliente particulière.

A la suite de l'opération 403, au cours d'un test 404, l'unité centrale 206 détermine si le paquet de données concerné est destiné à l'application, ou non, en fonction du résultat de l'opération de filtrage 403. Lorsque le résultat du test 404 est négatif, le test 402 est réitéré. Lorsque le résultat du test 404 est positif, au cours d'une opération 405, le paquet de données concerné est traité.

REVENDICATIONS

1. Procédé de filtrage de paquets de données entrants dans un système informatique fonctionnant en mode multitâche, en vue de déterminer chaque application à laquelle lesdits paquets de données sont destinés, chaque application s'exécutant sous le contrôle d'un ordonnanceur effectuant des étapes d'ordonnancement, caractérisé en ce que, pour chaque application susceptible de recevoir lesdits paquets de données, le procédé comporte un fil d'exécution de filtrage au cours duquel on détermine si lesdits paquets de données sont destinés, ou non, à ladite application, l'ordonnanceur contrôlant individuellement l'exécution de chacun des fils d'exécution de chaque application et l'exécution de chaque fil d'exécution de filtrage.

2. Procédé de filtrage de paquets de données selon la revendication 1, caractérisé en ce que, chaque fil d'exécution de filtrage est associé à au moins une application.

3. Procédé de filtrage de paquets de données selon l'une quelconque des revendications 1 ou 2, caractérisé en ce que, au cours desdites étapes d'ordonnancement, on ordonne consécutivement, d'une part, un fil d'exécution de filtrage avec, d'autre part, au moins un fil d'exécution d'une application à laquelle ledit fil d'exécution de filtrage est associé.

4. Procédé de filtrage de paquets de données selon l'une quelconque des revendications 1 à 3, caractérisé en ce que au cours desdites étapes d'ordonnancement, on met en œuvre une qualité de service.

5. Procédé de filtrage de paquets de données selon l'une quelconque des revendications 1 à 4, caractérisé en ce que, au cours desdites étapes d'ordonnancement, on décompte le temps d'exécution d'un fil d'exécution de filtrage du temps d'exécution alloué à l'ensemble des fils d'exécution de l'application à laquelle ledit fil d'exécution de filtrage est associé.

6. Procédé de filtrage de paquets de données selon l'une quelconque des revendications 1 à 5, caractérisé en ce que, pour chaque paquet de données entrant, il comporte une opération de mémorisation de descripteurs de blocs de mémoire tampon où est conservé ledit paquet de données entrant, dans une liste dite "de réception" ("Rx ring") et une opération de déclenchement d'interruption.

7. Procédé de filtrage de paquets de données selon l'une quelconque des revendications 1 à 6, caractérisé en ce que, pour chaque paquet de données entrant, il comporte une opération d'interruption au cours de laquelle on mémorise au moins un descripteur de bloc de mémoire tampon où est conservé ledit paquet de données entrant, dans une liste dite "de filtrage" ("Filter list").

8. Procédé de filtrage de paquets de données selon les revendications 6 et 7, caractérisé en ce que au cours de ladite opération d'interruption, on retire de la liste de réception ("Rx ring") chaque descripteur correspondant à un paquet de données entrant traité au cours de ladite opération d'interruption.

9. Procédé de filtrage de paquets de données selon l'une quelconque des revendications 7 ou 8, caractérisé en ce que, au cours de chaque fil d'exécution de filtrage associé à au moins une application, on détermine, si des paquets de données représentés dans la liste de filtrage sont destinés à une desdites applications, ou non.

10. Procédé de filtrage de paquets de données selon l'une quelconque des revendications 1 à 9, caractérisé en ce que, lorsque au cours d'un fil d'exécution de filtrage on détermine que des paquets de données sont destinés à une application, on effectue une étape de traitement desdits paquets de données.

11. Procédé de filtrage de paquets de données selon la revendication 10, caractérisé en ce que au cours de ladite opération de traitement desdits paquets de données, on exporte les paquets de données vers chaque application à laquelle le fil d'exécution de filtrage est associé.

12. Procédé de filtrage de paquets de données selon l'une quelconque des revendications 10 ou 11, caractérisé en ce que au cours de ladite opération de traitement desdits paquets de données, on envoie un acquittement à la source desdits paquets de données entrants.

13. Procédé de filtrage de paquets de données selon l'une quelconque des revendications 6 à 8, caractérisé en ce que, au cours de chaque opération d'interruption, on réveille les fils d'exécution de filtrage qui ne sont pas déjà réveillés.

14. Procédé de filtrage de paquets de données selon l'une quelconque des revendications 7 à 13, caractérisé en ce que lorsque il n'y a plus de paquets de données entrants à traiter par le fil d'exécution de filtrage en cours d'exécution, on met en sommeil ledit fil d'exécution de filtrage.

15. Procédé de filtrage de paquets de données selon les revendications 13 et 14, caractérisé en ce que le réveil des fils d'exécution de filtrage et leur mise en sommeil est effectuée par mise en œuvre d'un sémaphore.

16. Dispositif de filtrage de paquets de données entrants dans un système informatique fonctionnant en mode multitâche, en vue de déterminer chaque application à laquelle lesdits paquets de données sont destinés, chacun des fils d'exécution de chaque application s'exécutant sous le contrôle d'un moyen d'ordonnancement, caractérisé en ce que, pour chaque application susceptible de recevoir lesdits paquets de données, il comporte un moyen de

filtrage adapté à déterminer si lesdits paquets de données sont destinés, ou non, à ladite application, le moyen d'ordonnancement contrôlant individuellement chacun des fils d'exécution de chaque application et chaque moyen de filtrage.

17. Dispositif de filtrage de paquets de données selon la revendication 16, caractérisé en ce que chaque moyen de filtrage est associée à au moins une application.

18. Dispositif de filtrage de paquets de données selon l'une quelconque des revendications 16 ou 17, caractérisé en ce que le moyen d'ordonnancement est adapté à ordonner consécutivement, d'une part, un fil d'exécution de filtrage avec, d'autre part, au moins un fil d'exécution d'une application à laquelle le moyen de filtrage est associé.

19. Dispositif de filtrage de paquets de données selon l'une quelconque des revendications 16 à 18, caractérisé en ce que le moyen d'ordonnancement met en œuvre une qualité de service.

20. Dispositif de filtrage de paquets de données selon l'une quelconque des revendications 16 à 19, caractérisé en ce que le moyen d'ordonnancement est adapté à décompter le temps d'exécution d'un moyen de filtrage du temps d'exécution alloué à l'ensemble des fils d'exécution d'une application à laquelle ledit moyen de filtrage est associé.

21. Dispositif de filtrage de paquets de données selon l'une quelconque des revendications 16 à 20, caractérisé en ce qu'il comporte un moyen de traitement adapté, pour chaque paquet de données entrant, à mettre en mémoire au moins un descripteur de blocs de mémoire tampon où est conservé ledit paquet de données entrant, dans une liste dite "de réception" ("Rx ring") et à déclencher une interruption.

22. Dispositif de filtrage de paquets de données selon l'une quelconque des revendications 16 à 21, caractérisé en ce qu'il comporte un moyen de traitement adapté, pour chaque paquet de données entrant, à exécuter une interruption au cours de laquelle il met en mémoire au moins un descripteur de blocs de mémoire tampon où est conservé ledit paquet de données entrant, dans une liste dite "de filtrage" ("Filter list").

23. Dispositif de filtrage de paquets de données selon les revendications 21 et 22, caractérisé en ce que le moyen de traitement est adapté au cours de ladite interruption, à retirer de la liste de réception ("Rx ring") chaque descripteur correspondant à un paquet de données entrant traité au cours de ladite interruption.

24. Dispositif de filtrage de paquets de données selon l'une quelconque des revendications 22 ou 23, caractérisé en ce que chaque moyen de filtrage associé à au moins une application est adapté à déterminer si des paquets de données représentés dans la liste de filtrage sont destinés à une desdites applications, ou non.

25. Dispositif de filtrage de paquets de données selon l'une quelconque des revendications 16 à 24, caractérisé en ce que chaque moyen de filtrage est adapté, lorsqu'il a déterminé que des paquets de données sont destinés à une application, à effectuer un traitement desdits paquets de données.

26. Dispositif de filtrage de paquets de données selon la revendication 25, caractérisé en ce que le moyen de filtrage est adapté, à titre de traitement desdits paquets de données, à exporter les paquets de données vers chaque application à laquelle le moyen de filtrage est associée.

27. Dispositif de filtrage de paquets de données selon l'une quelconque des revendications 25 ou 26, caractérisé en ce que le moyen de

filtrage est adapté, à titre de traitement desdits paquets de données à envoyer un acquittement à la source desdits paquets de données entrants.

28. Dispositif de filtrage de paquets de données selon l'une quelconque des revendications 21 à 23, caractérisé en ce que, au cours de chaque interruption, le moyen de traitement réveille les moyens de filtrage qui ne sont pas déjà réveillés.

29. Dispositif de filtrage de paquets de données selon la revendication 28, caractérisé en ce que lorsque le moyen de traitement est adapté à mettre en sommeil chaque moyen de filtrage qui n'a plus de paquets de données entrants à traiter.

30. Dispositif de filtrage de paquets de données selon les revendications 28 et 29, caractérisé en ce que le moyen de traitement est adapté à mettre en œuvre, pour chaque moyen de filtrage, un sémaphore pour réveiller et mettre en sommeil ledit moyen de filtrage.

31. Ordinateur, caractérisé en ce qu'il comporte un dispositif selon l'une quelconque des revendications 16 à 30.

32. Caméra, caractérisé en ce qu'elle comporte un dispositif selon l'une quelconque des revendications 16 à 30.

33. Télécopieur, caractérisé en ce qu'il comporte un dispositif selon l'une quelconque des revendications 16 à 30.

34. Appareil photographique, caractérisé en ce qu'il comporte un dispositif selon l'une quelconque des revendications 16 à 30.

35. Téléviseur, caractérisé en ce qu'il comporte un dispositif selon l'une quelconque des revendications 16 à 30.

36. Imprimante, caractérisé en ce qu'elle comporte un dispositif selon l'une quelconque des revendications 16 à 30.

37. Scanner, caractérisé en ce qu'il comporte un dispositif selon l'une quelconque des revendications 16 à 30.

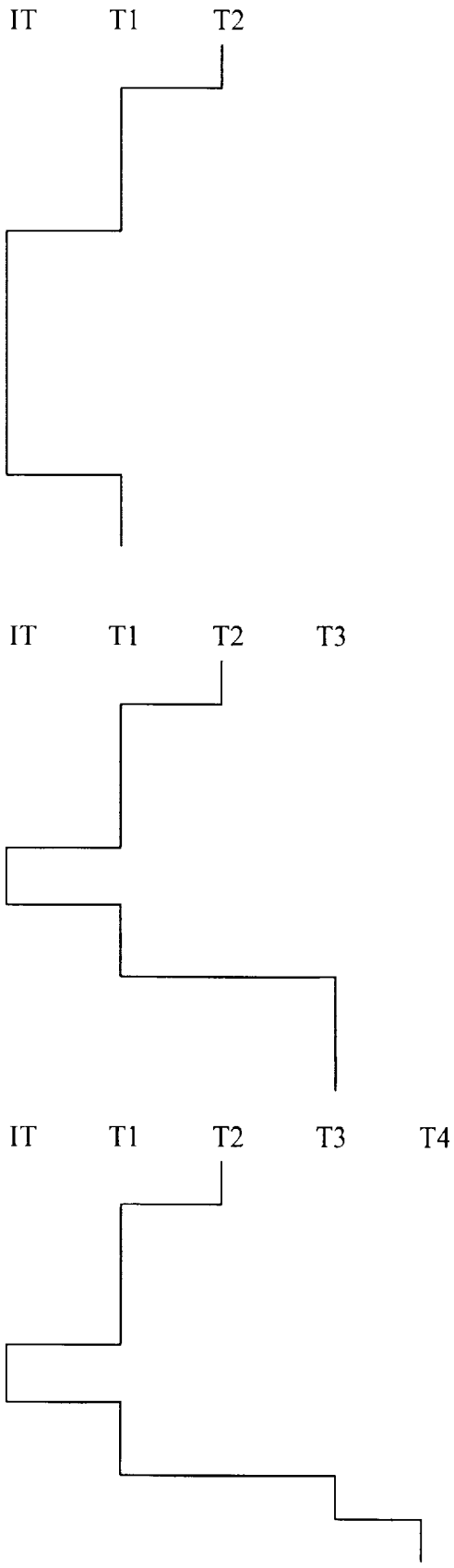


Fig. 1

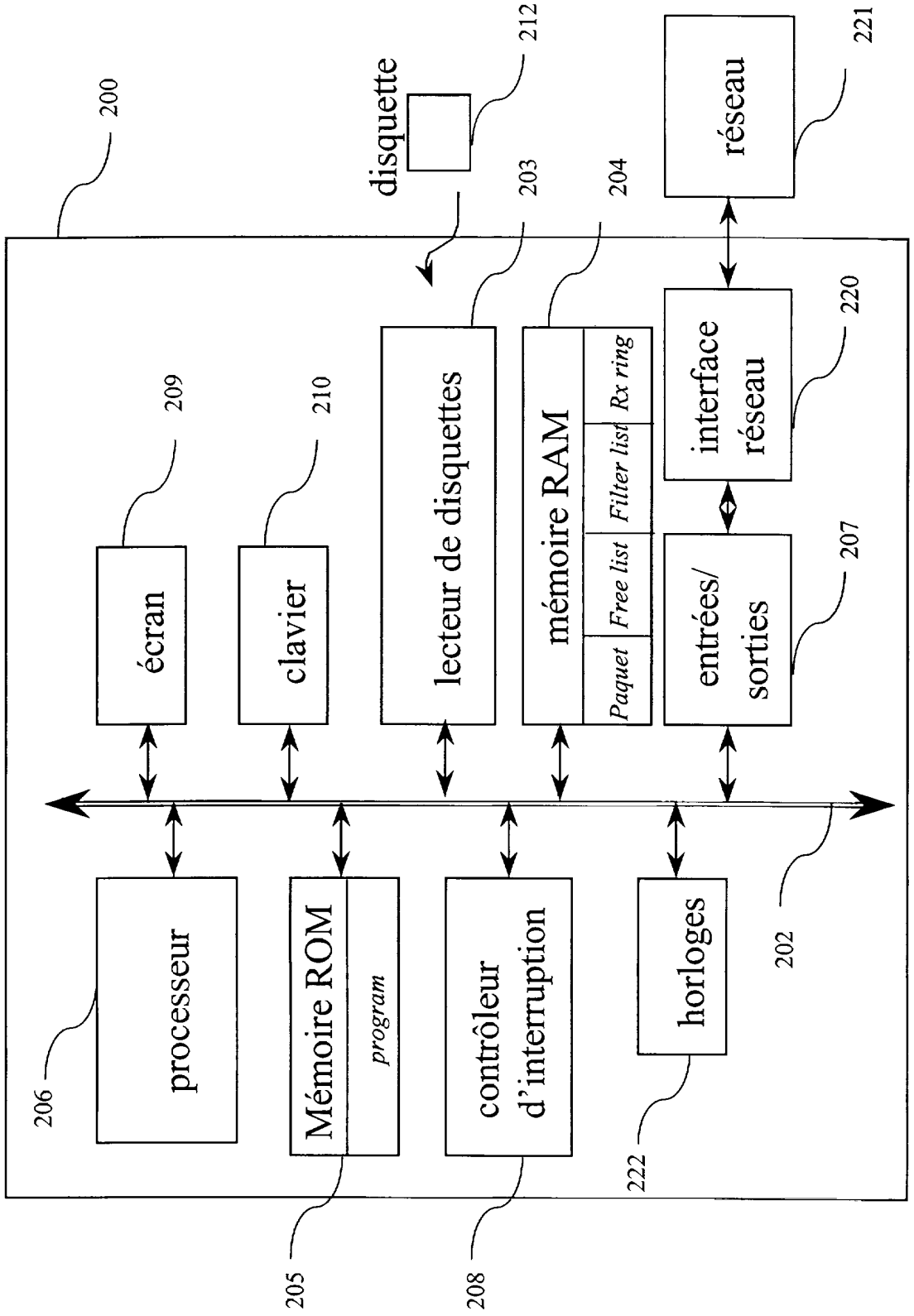

200
disquette
212
221
réseau
203
204
209
210
écran
clavier
lecteur de disquettes
mémoire RAM
Rx ring
Filter list
Free list
Paquet
interface réseau
220
entrées/ sorties
207
processeur
206
Mémoire ROM
program
contrôleur d'interruption
horloges
222
202
205
208

Fig. 2

2792431

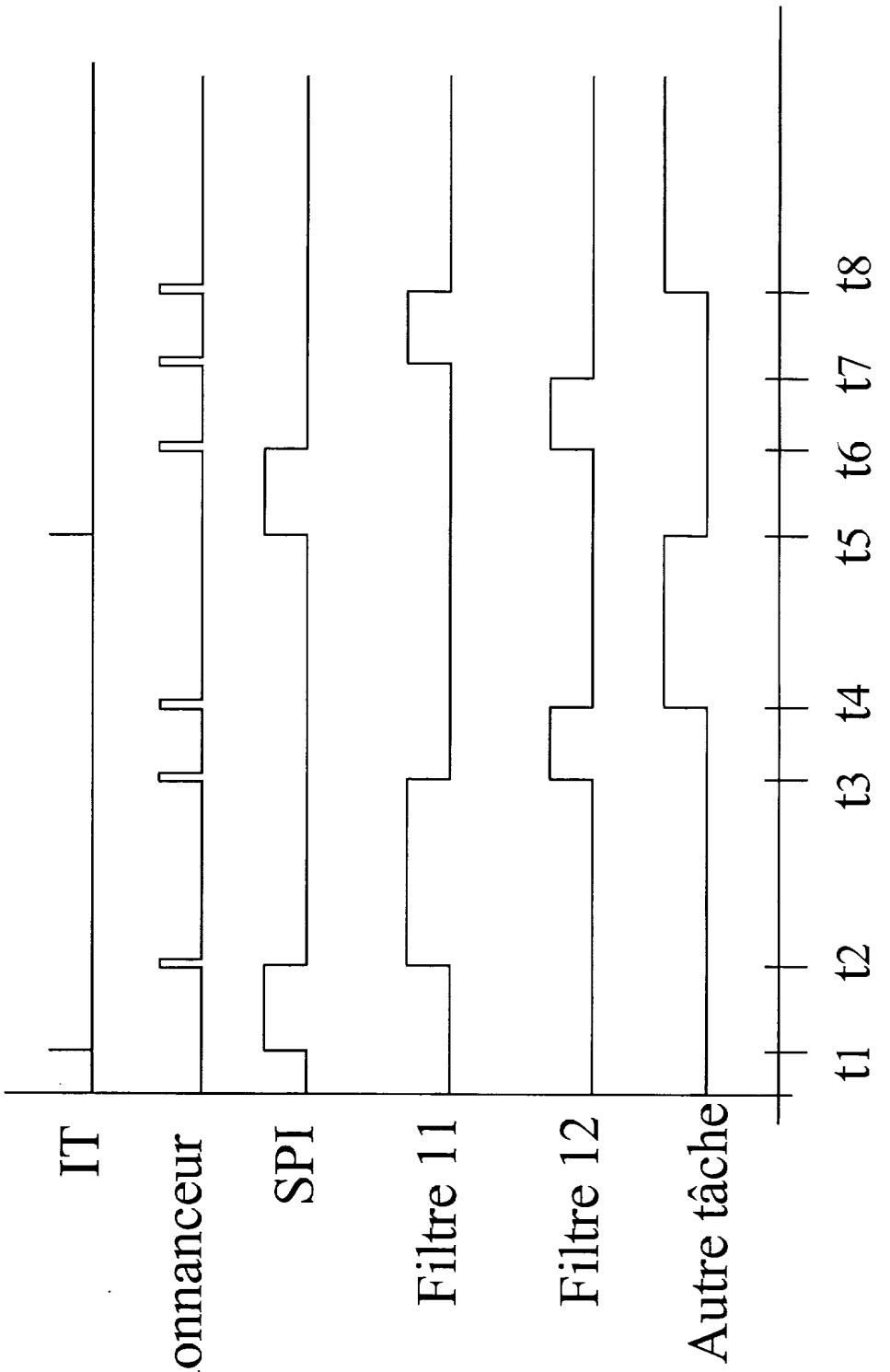


Fig. 3

3/5

2792431

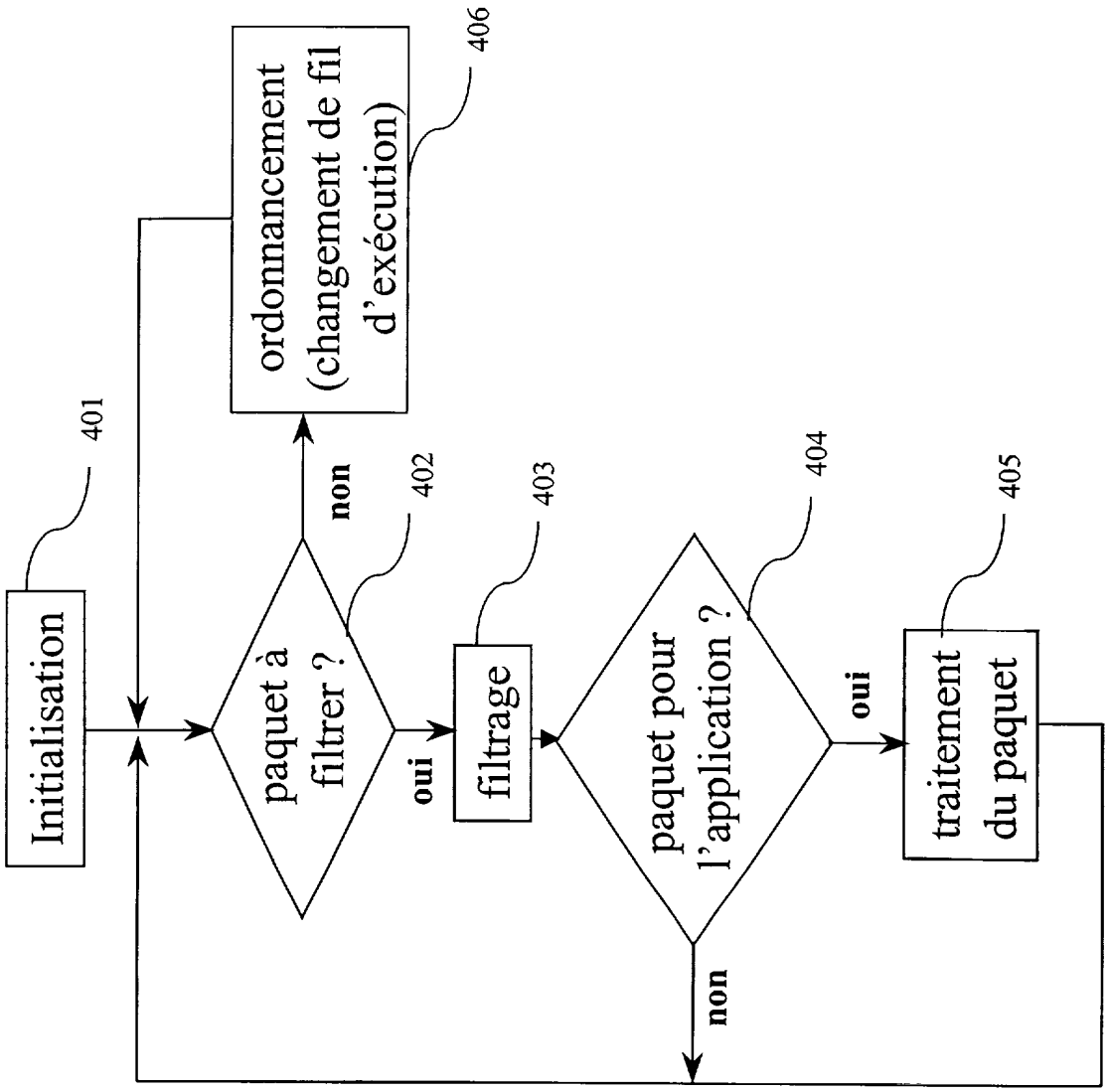


Fig. 4

4/5

2792431

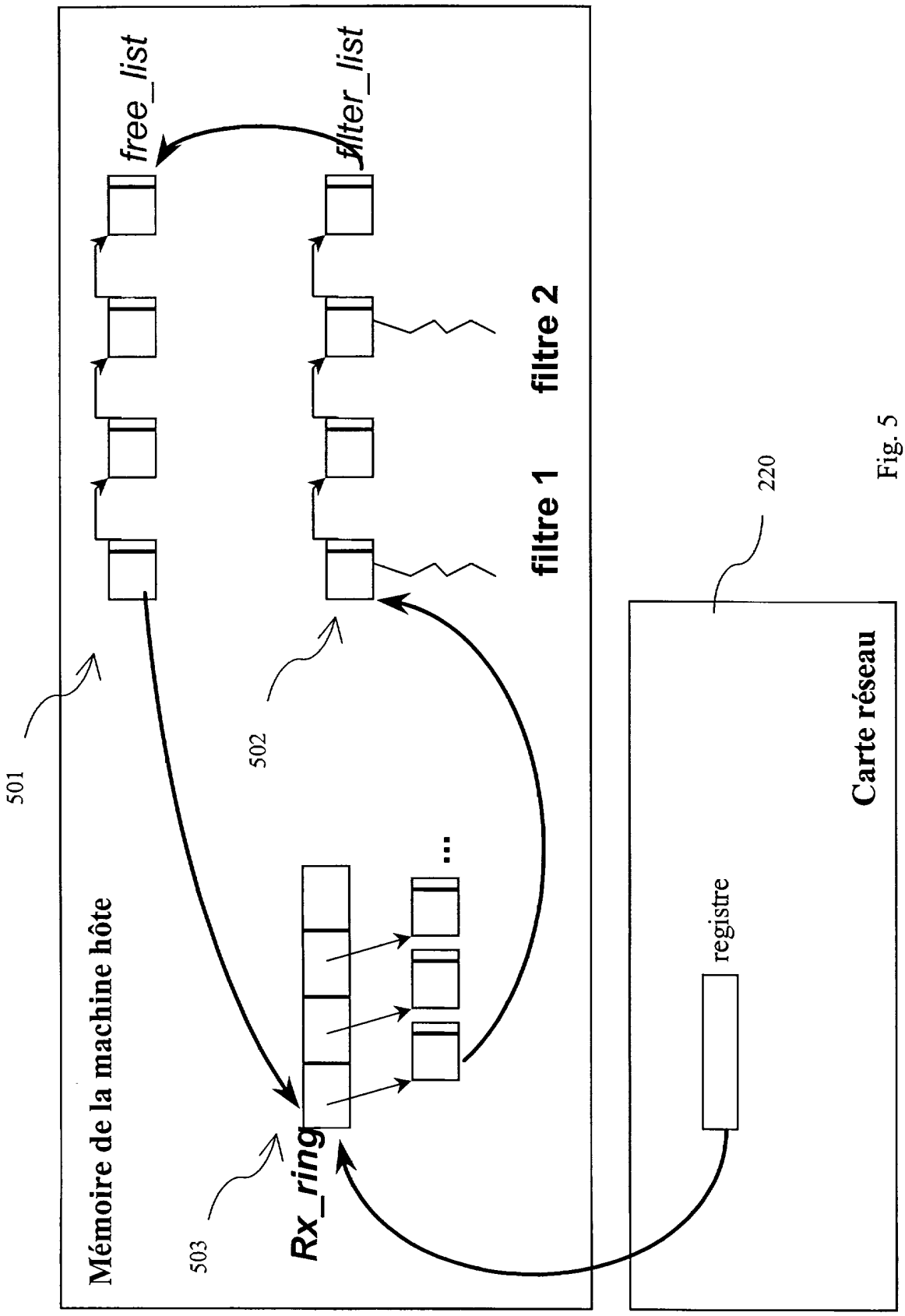


Fig. 5

REPUBLIQUE FRANÇAISE

INSTITUT NATIONAL de la PROPRIÉTÉ INDUSTRIELLE

RAPPORT DE RECHERCHE PRELIMINAIRE

établi sur la base des dernières revendications déposées avant le commencement de la recherche

2792431
N° d'enregistrement national

FA 578826
FR 9904886

DOCUMENTS CONSIDERES COMME PERTINENTS

Catégorie	Citation du document avec indication, en cas de besoin, des parties pertinentes	Revendications concernées de la demande examinée
A	YAU D K Y ET AL: "Migrating sockets-end system support for networking with quality of service guarantees" IEEE/ACM TRANSACTIONS ON NETWORKING, 'Online! vol. 6, no. 6, décembre 1998 (1998-12), pages 700-716, XP000799237 IEEE;ACM, USA ISSN: 1063-6692 Retrieved from the Internet: <URL:http://iel.ihs.com:80/cgi-bin/iel_cgi ?sess=177931718&prod=IEL&page=%2fiel4%2f90 %2f16154%2f00748083%2epdf> 'retrieved on 2000-03-14! * abrégé * * page 701, colonne de droite, ligne 44 - page 703, colonne de gauche, ligne 12; figure 1 * * page 707, colonne de droite, ligne 38 - page 709, colonne de gauche, ligne 36; figure 7 * ---	1,16
A	MOGUL J C ET AL: "THE PACKET FILTER: AN EFFICIENT MECHANISM FOR USER-LEVEL NETWORK CODE" OPERATING SYSTEMS REVIEW (SIGOPS),US,ACM HEADQUARTER. NEW YORK,1987, pages 39-51, XP002913603 * abrégé * * page 41, colonne de droite, ligne 10 - page 45, colonne de gauche, ligne 11; figures 3.1-3.9 * --- -/--	1,16

DOMAINES TECHNIQUES RECHERCHES (Int.CL.7)

G06F
H04L

1

Date d'achèvement de la recherche	Examinateur
17 mars 2000	Wiltink, J

CATEGORIE DES DOCUMENTS CITES

X : particulièrement pertinent à lui seul
Y : particulièrement pertinent en combinaison avec un autre document de la même catégorie
A : pertinent à l'encontre d'au moins une revendication ou arrière-plan technologique général
O : divulgation non-écrite
P : document intercalaire

T : théorie ou principe à la base de l'invention
E : document de brevet bénéficiant d'une date antérieure à la date de dépôt et qui n'a été publié qu'à cette date de dépôt ou qu'à une date postérieure.
D : cité dans la demande
L : cité pour d'autres raisons

& : membre de la même famille, document correspondant

EPO FORM 1503 03.82 (P04C13)

REPUBLIQUE FRANÇAISE

INSTITUT NATIONAL
de la
PROPRIETE INDUSTRIELLE

RAPPORT DE RECHERCHE PRELIMINAIRE

établi sur la base des dernières revendications déposées avant le commencement de la recherche

2792431
N° d'enregistrement national

FA 578826
FR 9904886

DOCUMENTS CONSIDERES COMME PERTINENTS

Catégorie	Citation du document avec indication, en cas de besoin, des parties pertinentes	Revendications concernées de la demande examinée
A	REYNOLDS F ET AL: "KERNEL SUPPORT FOR NETWORK PROTOCOL SERVERS" PROCEEDINGS OF THE USENIX MACH SYMPOSIUM, 20 - 22 novembre 1991, XP000602354 Berkeley, CA, USA * abrégé * * page 151, ligne 4 - ligne 29 *	1,16

DOMAINES TECHNIQUES RECHERCHES (Int.CL.7)

Date d'achèvement de la recherche	Examinateur
17 mars 2000	Wiltink, J

CATEGORIE DES DOCUMENTS CITES

X : particulièrement pertinent à lui seul
Y : particulièrement pertinent en combinaison avec un autre document de la même catégorie
A : pertinent à l'encontre d'au moins une revendication ou arrière-plan technologique général
O : divulgation non-écrite
P : document intercalaire

T : théorie ou principe à la base de l'invention
E : document de brevet bénéficiant d'une date antérieure à la date de dépôt et qui n'a été publié qu'à cette date de dépôt ou qu'à une date postérieure.
D : cité dans la demande
L : cité pour d'autres raisons

& : membre de la même famille, document correspondant

1

EPO FORM 1503 03.82 (P04C13)