

12 DEMANDE DE BREVET D'INVENTION

A1

22 Date de dépôt : 22.03.22.

30 Priorité :

43 Date de mise à la disposition du public de la
demande : 29.09.23 Bulletin 23/39.

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 :

○ Demande(s) d'extension :

71 Demandeur(s) : ORANGE Société anonyme — FR,
*Institut National de Recherche en Informatique et en
Automatique (Inria) Etablissement public à caractère
scientifique et technologique — FR et Sorbonne Univer-
sité Etablissement public à caractère scientifique, cultu-
rel et professionnel — FR.*

72 Inventeur(s) : GHIGOFF Yoann, LAZRI Kahina,
MULLER Gilles et SOPENA Julien.

73 Titulaire(s) : ORANGE Société anonyme, Institut National
de Recherche en Informatique et en Automatique (Inria) Eta-
blissement public à caractère scientifique et technologique,
Sorbonne Université Etablissement public à caractère scienti-
fique, culturel et professionnel.

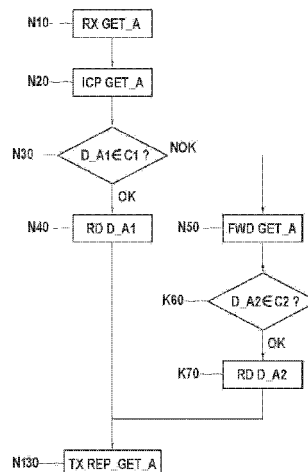
74 Mandataire(s) : CABINET BEAU DE LOMENIE.

54 Procédé de traitement de requêtes d'accès à des données, entité, programme d'ordinateur, et support
d'informations associés.

57 L'invention concerne un procédé de traitement de re-
quêtes d'accès (GET_A) à des données, mis en œuvre par
une entité (SERV) comprenant une carte réseau (NIC) et un
noyau de système d'exploitation (KER), ledit procédé com-
prenant des étapes de:

interception (N20) par la carte réseau (NIC) d'une re-
quête de lecture (GET_A) comprenant un identifiant (ID_A)
d'une donnée (D_A); transmission (N50) par la carte réseau
(NIC) de la requête (GET_A) au noyau (KER) en fonction de
la présence d'une donnée (D_A1) correspondant audit identi-
fiant (ID_A) dans une première mémoire cache (C1) de la
carte réseau (NIC); émission (N130) par la carte réseau
(NIC) d'une réponse (REP_GET_A) à la requête (GET_A)
en fonction de la présence d'une donnée (D_A1, D_A2) cor-
respondant audit identifiant (ID_A) dans la première mé-
moire cache (C1) ou dans une deuxième mémoire cache
(C2) du noyau (KER).

Figure pour l'abrégié : Fig. 2B



Description

Titre de l'invention : Procédé de traitement de requêtes d'accès à des données, entité, programme d'ordinateur, et support d'informations associés

Domaine technique

[0001] La présente invention se rapporte aux domaines des réseaux informatiques et du stockage d'informations. Plus particulièrement, la présente invention concerne un procédé de traitement de requêtes d'accès à des données, et également une entité, un programme d'ordinateur, et un support d'informations associés. La présente invention trouve une application particulièrement avantageuse, bien que nullement limitative, pour la mise en œuvre d'applications réseaux ou de stockage de données en réseau, par exemple des applications de cache réseau.

Technique antérieure

[0002] De manière générale, les dispositifs de stockage de données dans les réseaux informatiques sont des composants essentiels pour la mise en œuvre d'applications réseaux, et de services Internet. De tels dispositifs sont d'autant plus critiques pour le déploiement à grande échelle d'applications réseaux, ceux-ci devant traiter un nombre élevé de requêtes et fournir un accès aux données avec une faible latence, et ce en particulier pour les données fréquemment requêtées.

[0003] La mise en œuvre d'applications réseaux à grande échelle suppose toutefois de résoudre une problématique de charge sur les dispositifs de stockage de données. En effet, les dispositifs classiques, comprenant des moyens informatiques matériels et logiciels génériques, ne sont pas en mesure de traiter la totalité de la charge et des requêtes provenant du réseau avec des performances satisfaisantes. Cette situation peut conduire à une congestion du réseau au niveau des dispositifs de stockage de données et à une dégradation de la qualité de service des applications réseaux, telle qu'une latence d'accès aux données plus importante, un temps de traitement des requêtes plus long, voire le non-traitement de certaines requêtes ou la perte de données, etc.

[0004] En vue de réduire la charge au niveau d'un équipement réseau, il est connu d'utiliser des dispositifs de répartition de charge. Les dispositifs de répartition de charge distribuent l'ensemble des tâches sur un ensemble de ressources. En l'espèce, de tels dispositifs peuvent être utilisés pour distribuer la charge réseau et les requêtes entre plusieurs équipements d'une application réseau, ce qui permet de pallier les risques de congestion. Néanmoins, l'utilisation de dispositifs de répartition de charge présente l'inconvénient de nécessiter un surdimensionnement du nombre d'équipements réseaux, ce qui conduit à une sous-utilisation de ces derniers.

[0005] Afin de déployer des applications réseaux de débit d'informations élevé et de faible latence, il est également connu d'avoir recours à des équipements réseaux dédiés, optimisés pour le stockage de données et comprenant des piles réseaux spécialisées. Par exemple, de tels équipements réseaux dédiés peuvent incorporer des mécanismes de contournement du noyau de système d'exploitation (ou « kernel-bypass » en anglais). Ces piles réseaux spécialisées, et les mécanismes de contournement de noyau, permettent d'améliorer les performances réseaux en évitant les coûts en performance, tels que les temps de traitement et les latences, liés à la généricité des systèmes d'exploitation standards. Toutefois, ce type de solution requiert un accès exclusif à une carte réseau, ce qui rend alors impossible le partage de la carte réseau et des ressources matérielles avec d'autres applications nécessitant un accès réseau. De plus, la mise en œuvre d'un mécanisme de contournement de noyau n'est pas transparente pour une application réseau. En effet, pour mettre en œuvre un mécanisme de contournement de noyau, il est nécessaire de modifier de manière significative l'application réseau, notamment en y incorporant des fonctions réseaux spécifiques et en exécutant des programmes issus de bibliothèques dédiées.

[0006] Il est également connu d'utiliser une mémoire cache pour réduire le temps d'accès à des données. Dans le cas d'un dispositif de stockage de données en réseau, une mémoire cache peut être intégrée entre l'interface réseau et l'application de stockage de données. Le dispositif de stockage est par conséquent en mesure de traiter certaines requêtes en utilisant les données enregistrées dans la mémoire cache. Le traitement d'une requête grâce à la mémoire cache permet de réduire le temps de traitement de celle-ci par le dispositif de stockage. Néanmoins, lorsque de tels dispositifs de stockage sont utilisés, la latence d'accès aux données n'est pas entièrement satisfaisante. En vue de la mise en œuvre d'applications réseaux à grande échelle, il est nécessaire de réduire davantage la latence d'accès aux données et d'augmenter le débit des requêtes traitées.

Exposé de l'invention

[0007] La présente invention a pour objectif de remédier à tout ou partie des inconvénients de l'art antérieur, notamment ceux exposés précédemment.

[0008] À cet effet, selon un aspect de l'invention, il est proposé un procédé de traitement de requêtes d'accès à des données, le procédé étant mis en œuvre par une entité connectée à un réseau, ladite entité comprenant une carte réseau et un noyau de système d'exploitation. Le procédé comprend des étapes de :

- interception par la carte réseau d'un paquet réseau à destination d'une application configurée pour traiter et répondre auxdites requêtes d'accès, ledit

paquet réseau portant une requête de lecture comprenant un identifiant d'une donnée ;

- transmission par la carte réseau de la requête de lecture au noyau en fonction de la présence d'une donnée correspondant audit identifiant dans une première mémoire cache de la carte réseau ; et
- émission par la carte réseau d'une réponse à la requête de lecture, ladite réponse étant déterminée en fonction de la présence d'une donnée correspondant audit identifiant dans la première mémoire cache ou dans une deuxième mémoire cache du noyau.

[0009] Par « requêtes d'accès à des données », il est fait référence ici à des requêtes d'accès en écriture ou en lecture de données informatiques. Typiquement, les requêtes peuvent être émises par des dispositifs clients et transmises par l'intermédiaire d'un réseau de communication.

[0010] Par « mémoire cache », il est fait référence ici à une mémoire qui enregistre des copies de données d'une application en vue de diminuer le temps d'accès à ces données.

[0011] Par « noyau », il est fait référence ici à un noyau de système d'exploitation (ou « kernel » en anglais) et ainsi à un programme d'ordinateur permettant de gérer les ressources d'un ordinateur et permettant aux différents composants, matériels et logiciels, de communiquer entre eux. Dans le contexte de la présente invention, les étapes mises en œuvre par le noyau sont réalisées dans l'espace noyau. Par exemple, les étapes mises en œuvre par le noyau sont réalisées par les pilotes informatiques (ou « in-kernel drivers » en anglais) de la carte réseau, les pilotes informatiques étant installés dans le noyau.

[0012] Par « application », il est fait référence ici à un programme d'ordinateur mis en œuvre dans l'espace utilisateur, reposant ainsi sur les mécanismes d'abstraction fournis par le noyau. Dans le contexte de la présente invention, et de manière différente des étapes mises en œuvre par le noyau, des étapes mises en œuvre par l'application sont réalisées dans l'espace utilisateur. À titre d'exemple, l'application peut être une application de stockage de données en réseau.

[0013] Selon le procédé proposé, l'étape de transmission d'une requête de lecture au noyau est réalisée en fonction de la présence d'une donnée correspondant à l'identifiant dans la première mémoire cache. À titre d'exemple, si la carte réseau comprend la donnée demandée, la carte réseau traite la requête et ne la transmet donc pas au noyau. Selon un autre exemple, si la première mémoire cache ne comprend pas la donnée demandée, la carte réseau transmet la requête au noyau.

[0014] De plus, selon le procédé proposé, la réponse à une requête de lecture est déterminée en fonction de la présence d'une donnée correspondant à l'identifiant dans les

mémoires caches. Par exemple, si une des mémoires caches comprend la donnée demandée, la réponse émise comprend la donnée lue dans cette mémoire cache. Selon un autre exemple, si l'identifiant ne correspond à aucune donnée de l'application, la réponse émise peut être un message d'erreur.

- [0015] L'invention proposée consiste en une architecture hiérarchique de mémoires caches sur plusieurs niveaux d'un système informatique. Cette architecture multi-caches, comprenant un premier niveau de cache mis en œuvre par la carte réseau et un deuxième niveau de cache mis en œuvre par le noyau, vise notamment à accélérer la latence liée aux accès réseaux de lecture de données d'une application. Ainsi, les requêtes sont traitées au plus bas niveau possible, soit par la carte réseau, soit par le noyau. Si une requête de lecture de donnée ne peut être traitée par le premier niveau de cache mis en œuvre par la carte réseau, la requête est transmise au deuxième niveau de cache mis en œuvre par le noyau.
- [0016] Les premier et deuxième niveaux de cache permettent d'accélérer le traitement des requêtes d'accès à des données. La carte réseau est configurée pour traiter et répondre à une requête reçue dans un paquet réseau et comprend le premier niveau de cache. Le traitement d'une requête par la carte réseau permet d'éviter son transfert au noyau et ainsi de réduire la latence de traitement. Le traitement d'une requête par la carte réseau permet en outre d'économiser des ressources de traitement puisque le traitement du paquet réseau par le noyau n'est pas nécessaire. Le noyau est configuré pour traiter et répondre à une requête et comprend le deuxième niveau de cache. Par exemple, le noyau est configuré pour traiter les requêtes en interceptant les paquets réseaux au niveau des pilotes informatiques de la carte réseau. Par exemple, les pilotes informatiques comprennent un point d'attache (connu sous le nom de « hook » en anglais) permettant de programmer un traitement particulier sur les paquets réseaux reçus, ce qui permet de modifier le traitement par défaut des paquets réseaux réalisé par le noyau. Le traitement d'une requête par le noyau permet d'éviter son transfert à l'application et par conséquent de réduire la latence de traitement et d'économiser des ressources de traitement puisque la requête n'est pas traitée par l'application.
- [0017] En traitant les requêtes de lecture au niveau de la carte réseau ou au niveau du noyau, le procédé proposé permet entre autres de réduire la latence d'accès aux données, d'augmenter le nombre et le débit des requêtes traitées et de minimiser l'utilisation des ressources de traitement. Le procédé proposé permet ainsi de pallier les risques de congestion. En outre, le procédé proposé permet une utilisation efficace des ressources matérielles et logicielles de l'entité et le partage de celles-ci avec d'autres applications.
- [0018] En outre, la mise en œuvre du premier niveau de cache au niveau de la carte réseau et du deuxième niveau de cache au niveau de l'espace noyau est transparente pour l'application qui s'exécute dans l'espace utilisateur. Ainsi, mettre en œuvre et exploiter

les premier et deuxième niveaux de cache ne nécessitent aucunement de modifier l'application.

- [0019] Selon un mode de réalisation particulier, la transmission de la requête par la carte réseau au noyau est réalisée si la première mémoire cache ne comprend pas une donnée correspondant à l'identifiant compris dans la requête de lecture. De plus, si la première mémoire cache comprend une donnée correspondant à l'identifiant, alors la réponse à la requête comprend ladite donnée lue dans la première mémoire cache ; ou, si la deuxième mémoire cache comprend une donnée correspondant à l'identifiant compris dans la requête de lecture, ladite réponse comprend ladite donnée lue dans la deuxième mémoire cache.
- [0020] Ce mode de réalisation permet d'assurer la fiabilité des réponses aux requêtes de lecture émises par l'entité. En effet, selon ce mode de réalisation, une requête de lecture n'est traitée par la carte réseau que si la première mémoire cache comprend une donnée correspondant à l'identifiant compris dans la requête.
- [0021] Selon un mode de réalisation particulier, le procédé de traitement de requêtes comprend, si aucune desdites mémoires caches ne comprend une donnée correspondant audit identifiant, une étape de transmission de la requête de lecture à l'application, la réponse émise par la carte réseau comprenant une donnée correspondant audit identifiant, cette donnée étant émise par l'application en réponse à la requête de lecture.
- [0022] Ce mode de réalisation permet de garantir un traitement certain des requêtes de lecture. Si un niveau de cache ne comprend pas la donnée demandée, celui-ci transmettra nécessairement la requête au niveau supérieur ; et si aucun des premier et deuxième niveaux de cache ne sont en mesure de traiter la requête de lecture, alors celle-ci est traitée par l'application. Les étapes mises en œuvre par l'application sont réalisées au niveau de l'espace utilisateur, tandis que les étapes mises en œuvre par le noyau sont réalisées au niveau de l'espace noyau.
- [0023] Selon un mode de réalisation particulier, le procédé de traitement de requêtes comprend un mécanisme d'exception permettant à un niveau de cache de transmettre une requête au niveau supérieur, si ce niveau de cache n'est pas en mesure de traiter la requête. De manière préférentielle, une requête de lecture n'est transmise par un niveau de cache à un niveau supérieur que si la mémoire de ce niveau de cache ne comprend pas la donnée demandée.
- [0024] Selon un mode de réalisation, l'application est configurée pour gérer des accès concurrents. Par exemple, l'application est en mesure de traiter plusieurs requêtes d'écriture émises par des dispositifs clients différents et portant sur une même donnée, et une requête de lecture portant sur cette donnée. Ainsi, suite à une ou plusieurs requêtes d'écriture, une donnée enregistrée dans une des mémoires caches peut ne plus

être identique à la donnée de l'application correspondant au même identifiant. Il est alors nécessaire de tenir compte de la validité des données enregistrées dans les mémoires caches pour traiter des requêtes et répondre à celles-ci.

- [0025] Pour ce faire, selon un mode de réalisation particulier, la requête de lecture est transmise par la carte réseau au noyau si la première mémoire cache ne comprend pas une donnée valide correspondant audit identifiant ; et la requête de lecture est transmise par le noyau à l'application si aucune desdites mémoires caches ne comprend une donnée valide correspondant audit identifiant. En outre, selon ce mode de réalisation, ladite réponse émise par la carte réseau comprend une donnée valide correspondant audit identifiant.
- [0026] Par « donnée valide », il est fait référence ici, pour un certain identifiant, à une donnée qui peut être utilisée pour répondre à une requête de lecture comprenant cet identifiant. Pour un certain identifiant, une donnée valide est nécessairement identique à la donnée de l'application associée au même identifiant. Et par « donnée invalide », il est fait référence ici, pour un certain identifiant, à une donnée qui ne peut pas être utilisée pour répondre à une requête de lecture comprenant cet identifiant. Pour un certain identifiant, une donnée invalide n'est pas nécessairement identique à la donnée de l'application associée au même identifiant.
- [0027] Ce mode de réalisation permet de prendre en compte la validité des données enregistrées dans les mémoires caches pour traiter des requêtes et répondre à celles-ci. Ainsi, ce mode de réalisation permet de mettre en œuvre un traitement des requêtes par les différents niveaux de cache en fonction de la cohérence des données enregistrées dans les mémoires caches avec les données de l'application. Par ailleurs, ce mode de réalisation, en prenant en compte la validité des données, peut en outre être avantageux pour mettre en œuvre un mécanisme d'expiration de données.
- [0028] Dans le contexte de l'invention, la réponse à une requête de lecture comprend ladite donnée valide correspondant à l'identifiant compris dans la requête. En effet, si l'application comprend une donnée correspondant à l'identifiant compris dans la requête de lecture reçue, la réponse à une requête de lecture comprend la donnée demandée.
- [0029] Ce mode de réalisation permet d'assurer la fiabilité des réponses aux requêtes de lecture émises par l'entité. Si l'application comprend une donnée correspondant à l'identifiant de la requête de lecture, alors la donnée correspondant à l'identifiant est transmise par l'entité dans la réponse à la requête. Ainsi, pour les dispositifs clients émetteurs de requêtes, l'accès aux données de l'application présente, en plus d'une latence réduite, une fiabilité de traitement des requêtes par l'entité.
- [0030] Selon un mode de réalisation particulier, les requêtes d'accès à des données et/ou les réponses aux requêtes sont transmises par l'intermédiaire d'un réseau informatique

mettant en œuvre un protocole réseau IP (acronyme de l'expression anglo-saxonne « Internet Protocol »).

- [0031] L'utilisation du protocole IP permet d'acheminer des paquets de données entre un dispositif client et l'entité de traitement de requêtes. En outre, le protocole IP permet d'acheminer de manière efficace les paquets dans un réseau informatique mettant en œuvre un mécanisme de transmission « au mieux » (ou plus couramment désigné par l'expression anglo-saxonne « best-effort delivery »).
- [0032] Selon un mode de réalisation particulier, les requêtes de lecture et/ou les réponses aux requêtes de lecture sont transmises par l'intermédiaire d'un réseau informatique mettant en œuvre un protocole de transport UDP (acronyme de l'expression anglo-saxonne « User Datagram Protocol »).
- [0033] L'utilisation du protocole UDP permet de transmettre les requêtes de lecture et/ou leurs réponses de manière simple et rapide entre un dispositif client et l'entité de traitement de requêtes. La transmission de données avec le protocole UDP ne requiert aucune communication préalable pour établir la connexion entre un dispositif client et l'entité de traitement de requêtes et ne met pas en œuvre de retransmissions, ce qui permet réduire la latence d'accès aux données.
- [0034] Selon un mode de réalisation particulier, les paquets réseaux interceptés par la carte réseau sont les paquets réseaux dont le port de destination correspond au port de l'application.
- [0035] Selon un mode de réalisation particulier, le procédé de traitement de requêtes comprend en outre une étape de :
- enregistrement dans au moins une des mémoires caches de la donnée reçue de l'application en réponse à la requête de lecture, l'enregistrement étant réalisé en fonction dudit identifiant.
- [0036] Ce mode de réalisation permet d'enregistrer dans les mémoires caches des données. Le noyau utilise un mécanisme d'interception des données émises en réponse à des requêtes de lecture pour potentiellement les enregistrer dans les mémoires caches en fonction de leurs identifiants. En d'autres termes, le noyau enregistre ou non une donnée interceptée en fonction de son identifiant. Par exemple, le noyau peut enregistrer une donnée dans une mémoire cache, si cette donnée est fréquemment demandée. Ce mode de réalisation permet en particulier d'assurer la cohérence entre les données des mémoires caches et les données de l'application. De plus, le mécanisme d'interception permet d'enregistrer des données sans nécessiter de traitements et d'échanges de données supplémentaires entre les différents niveaux de cache et l'application. L'enregistrement d'une donnée dans la première mémoire de cache est effectué par le noyau, ce qui permet de libérer des ressources de traitement au niveau de la carte réseau.

- [0037] Selon un mode de réalisation particulier, l'enregistrement dans une des mémoires caches, de la donnée comprise dans la réponse à une requête de lecture, est réalisé si la mémoire cache comprend une donnée invalide ou aucune donnée correspondant à l'identifiant compris dans la requête de lecture.
- [0038] Un avantage de ce mode de réalisation est de permettre l'enregistrement d'une donnée dans la mémoire cache ou la mise à jour de la donnée enregistrée dans une mémoire cache. Ainsi, si une des mémoires caches comprend une donnée correspondant à l'identifiant de la requête de lecture et que cette donnée est invalide, alors la donnée émise en réponse à la requête est utilisée pour mettre à jour la donnée dans la mémoire cache. En particulier, la mise à jour de la donnée correspondant à l'identifiant compris dans la requête de lecture est réalisée en interceptant la donnée émise en réponse à la requête de lecture. Ce mécanisme d'interception permet de mettre à jour une donnée dans une mémoire cache sans nécessiter de traitements et d'échanges de données supplémentaires entre les différents niveaux de cache et l'application.
- [0039] Les mémoires caches des premier et deuxième niveaux de cache sont de tailles limitées. Il est donc essentiel d'enregistrer dans les mémoires caches les données de l'application fréquemment demandées, ce qui permet d'augmenter le nombre de requêtes traitées par la carte réseau ou le noyau et ainsi de réduire efficacement la latence d'accès aux données. Pour stocker une donnée dans une mémoire cache, l'entrée mémoire à utiliser est déterminée sur la base de l'identifiant de la donnée à stocker. Toutefois, des identifiants différents peuvent être associés à une seule et même entrée mémoire. Alors, plusieurs données sont en concurrence pour être enregistrées dans cette même entrée mémoire. En considérant une entrée mémoire pour laquelle plusieurs données sont en concurrence, enregistrer dans l'entrée mémoire la donnée la plus fréquemment demandée permet de réduire la latence d'accès aux données.
- [0040] Selon un mode de réalisation particulier, l'enregistrement dans une des mémoires caches, de la donnée émise par l'application en réponse à une requête de lecture, est réalisé en fonction d'une comparaison entre l'identifiant compris dans la requête de lecture et un identifiant d'une donnée enregistrée dans la mémoire cache. En particulier, l'identifiant compris dans la requête de lecture et l'identifiant de la donnée enregistrée sont associés à une seule et même entrée de la mémoire cache. La donnée comprise dans la réponse et la donnée enregistrée sont ainsi des données en concurrence pour une même entrée mémoire.
- [0041] Selon un mode de réalisation particulier, pouvant être combiné ou non avec le mode de réalisation précédent, l'enregistrement dans une des mémoires caches, de la donnée comprise dans la réponse à une requête de lecture, est réalisé en fonction d'une comparaison entre un seuil et un compteur associé à une donnée enregistrée dans la mémoire cache. En particulier, la donnée comprise dans la réponse et la donnée en-

registrée sont des données concurrentes pour une même entrée mémoire.

[0042] Un avantage de ces modes de réalisation est de maintenir dans la première mémoire cache et la deuxième mémoire cache les données fréquemment demandées tout en évitant que celles-ci ne soient remplacées par des données moins fréquemment accédées. En outre, ce mode de réalisation permet de minimiser la complexité des traitements pour sélectionner les données à enregistrer dans les différents niveaux de cache et de ne pas augmenter la latence d'accès aux données de l'application.

[0043] Selon un mode de réalisation particulier, les données enregistrées dans une mémoire cache sont les données demandées par le plus grand nombre de requêtes de lecture sur une période déterminée.

[0044] Le fait d'enregistrer dans les mémoires caches les données les plus fréquemment requêtées permet de traiter un nombre plus important de requêtes par le premier et le deuxième niveau de cache et ainsi de réduire davantage la latence d'accès aux données.

[0045] Selon un mode de réalisation particulier, au moins une des mémoires caches comprend une table de hachage, et le procédé de traitement de requêtes comprend une étape de :

- détermination d'un haché sur la base de l'identifiant compris dans la requête de lecture pour indexer la donnée correspondant à l'identifiant dans la table de hachage.

[0046] Le fait d'utiliser une table de hachage pour enregistrer les données dans une mémoire cache permet une association directe entre un identifiant et une donnée. Une telle table de hachage permet ainsi de retrouver une donnée rapidement, en la cherchant à un emplacement de la table correspondant au résultat d'une fonction de hachage. En particulier, une telle structure de données offre une complexité en temps constant pour les opérations de recherche, d'insertion ou d'invalidation d'une donnée. Ainsi, ce mode de réalisation permet de réduire la latence due aux traitements des requêtes et permet à l'entité de traiter une charge importante avec un débit de requêtes élevé.

[0047] Selon un mode de réalisation particulier, le haché pour indexer une donnée dans la table de hachage est déterminé sur la base de l'identifiant de la donnée et d'une fonction de hachage Fowler-Noll-Vo-1A, plus couramment désignée par l'acronyme FNV-1A.

[0048] Un avantage de ce mode de réalisation particulier est que la fonction de hachage FNV-1A permet notamment de déterminer le haché d'un identifiant tout en lisant l'identifiant dans la requête, permettant ainsi de réduire la latence due au traitement des requêtes et plus généralement de gagner en performance par rapport à un calcul de haché réalisé a posteriori.

[0049] Selon un mode de réalisation particulier, la mémoire cache de la carte réseau et la mémoire cache du noyau sont de tailles différentes. Plus particulièrement, la taille de la

mémoire cache de la carte réseau est inférieure à la taille de la mémoire cache du noyau.

[0050] Selon un mode de réalisation particulier, le procédé de traitement de requêtes comprend des étapes de :

- réception par la carte réseau d'une requête d'écriture comprenant un identifiant d'une donnée à écrire, et éventuellement la donnée à écrire ;
- transmission par la carte réseau de la requête d'écriture au noyau ; et
- transmission par le noyau de la requête d'écriture à l'application, l'application étant configurée pour traiter et répondre aux requêtes d'accès.

[0051] Un avantage de ces modes de réalisation particuliers est de garantir le traitement des requêtes d'écriture par l'application, permettant ainsi de garantir que les données de l'application restent à jour. L'application est configurée pour traiter des accès en écriture et en lecture concurrents. À titre d'exemple, l'application est configurée pour traiter deux requêtes d'écriture portant sur une même donnée et émises par des dispositifs clients différents, puis une requête de lecture portant sur cette donnée. Ainsi, ce mode de réalisation permet de contribuer à la gestion des accès concurrents.

[0052] Selon un mode de réalisation particulier, le procédé de traitement de requêtes comprend en outre une étape de :

- transmission d'une réponse à la requête d'écriture, la réponse comprenant un accusé de réception.

[0053] Ce mode de réalisation particulier permet d'accuser réception de la requête d'écriture au dispositif client émetteur de la requête, et ainsi de signaler au dispositif client le traitement de la requête d'écriture.

[0054] Selon un mode de réalisation particulier, les requêtes d'écriture et/ou les réponses aux requêtes d'écriture sont transmises par l'intermédiaire d'un réseau informatique mettant en œuvre un protocole de transport TCP (acronyme de l'expression anglo-saxonne « Transmission Control Protocol »).

[0055] Ce mode de réalisation particulier permet de mettre en œuvre des échanges de données fiables entre un dispositif client et une entité de traitement de requêtes dans le réseau informatique, et ainsi de garantir la livraison des requêtes d'écriture et/ou des réponses aux requêtes d'écritures.

[0056] Suite à la réception d'une requête d'écriture comprenant l'identifiant d'une donnée à écrire, l'application traite la requête d'écriture. Le traitement de la requête d'écriture par l'application peut entraîner ou non une modification de la donnée de l'application associée à cet identifiant. Si une mémoire cache comprend une donnée associée à cet identifiant, alors la donnée enregistrée peut : soit correspondre à une version antérieure et obsolète de la donnée de l'application ; soit être identique à la donnée de l'application. Ainsi, la donnée enregistrée dans la mémoire cache n'est pas néces-

sairement identique à la donnée de l'application. Il est alors nécessaire d'invalider la donnée enregistrée dans la mémoire cache pour que celle-ci ne soit pas utilisée pour répondre à une requête de lecture.

[0057] Pour ce faire, selon un mode de réalisation particulier, le procédé de traitement de requêtes comprend une étape de :

- si une des mémoires caches comprend une donnée correspondant à l'identifiant compris dans une requête d'écriture, invalidation dans la mémoire cache de la donnée correspondant à l'identifiant compris dans la requête d'écriture.

[0058] Ce mode de réalisation particulier permet de contribuer à garantir la cohérence des données de la première et de la deuxième mémoires caches avec les données de l'application. En outre, le fait d'invalider suite à une requête d'écriture les données correspondantes dans les différents niveaux de cache permet de s'assurer de ne pas transmettre une donnée obsolète en réponse à une requête de lecture. Ainsi, les requêtes d'écriture, avant d'être délivrées à l'application, sont utilisées par la carte réseau et le noyau pour invalider les entrées des différents niveaux de cache.

[0059] L'invalidation dans une mémoire cache d'une donnée peut être réalisée en mettant à une valeur nulle un indicateur de validité associé à la donnée. À l'inverse, une donnée valide peut être associée à un indicateur de validité de valeur non-nulle.

[0060] Selon un mode de réalisation particulier, le procédé de traitement de requêtes comprend une étape de :

- enregistrement d'un premier programme d'ordinateur dans la carte réseau, le premier programme d'ordinateur comportant des instructions pour la mise en œuvre des étapes réalisées par la carte réseau d'un procédé de traitement de requêtes tel que décrit-ci dessus, lorsque le premier programme d'ordinateur est exécuté par un premier processeur.

[0061] Selon un mode de réalisation particulier, le procédé de traitement de requêtes comprend une étape de :

- enregistrement d'un deuxième programme d'ordinateur dans le noyau, le deuxième programme d'ordinateur comportant des instructions pour la mise en œuvre des étapes réalisées par le noyau d'un procédé de traitement de requêtes tel que décrit-ci dessus, lorsque le deuxième programme d'ordinateur est exécuté par un deuxième processeur.

[0062] Selon un aspect de l'invention, il est proposé une entité de traitement de requêtes d'accès à des données, l'entité étant connectée à un réseau, l'entité comprenant une carte réseau et un noyau de système d'exploitation, la carte réseau comprenant une première mémoire cache, le noyau comprenant une deuxième mémoire cache, ladite entité comprenant :

- un module de réception réseau compris dans la carte réseau pour intercepter un paquet réseau à destination d'une application configurée pour traiter et répondre auxdites requêtes d'accès, ledit paquet réseau portant une requête de lecture comprenant un identifiant d'une donnée ;
- un premier module de détermination compris dans la carte réseau pour déterminer la présence dans la première mémoire cache d'une donnée correspondant audit identifiant ;
- un premier module de transmission compris dans la carte réseau pour transmettre la requête de lecture au noyau ;
- un deuxième module de détermination compris dans le noyau pour déterminer la présence dans la deuxième mémoire cache d'une donnée correspondant audit identifiant ; et
- un module d'émission réseau compris dans la carte réseau pour émettre une réponse à la requête de lecture.

[0063] Selon un mode de réalisation, ladite entité comprend :

- un premier module de lecture compris dans la carte réseau pour lire la donnée dans la première mémoire cache ; et
- un deuxième module de lecture compris dans le noyau pour lire la donnée dans la deuxième mémoire cache.

[0064] Selon un mode de réalisation, le premier module de détermination compris dans la carte réseau est configuré pour déterminer la présence et la validité d'une donnée correspondant à un identifiant dans la première mémoire cache. En outre, le deuxième module de détermination compris dans le noyau est configuré pour déterminer la présence et la validité d'une donnée correspondant à un identifiant dans la deuxième mémoire cache.

[0065] Selon un mode de réalisation particulier, l'entité comprend en outre une application, l'application comprenant une mémoire d'application pour enregistrer les données de l'application, l'entité comprenant en outre :

- un deuxième module de transmission compris dans le noyau pour transmettre la requête à l'application ;
- un module de traitement et de réponse compris dans l'application pour traiter et répondre aux requêtes d'accès à des données ;
- un deuxième module de réception compris dans le noyau pour intercepter la donnée valide émise par l'application en réponse à la requête ; et
- un premier module de réception compris dans la carte réseau pour recevoir en provenance de l'application une réponse à la requête de lecture.

[0066] Selon un mode de réalisation particulier, l'entité de traitement de requêtes est un serveur DNS (acronyme de l'expression anglo-saxonne « Domain Name System »), les

données de l'application étant par exemple des adresses IP et les identifiants étant des URLs (acronyme de l'expression anglo-saxonne « Uniform Resource Locator »).

- [0067] Selon un mode de réalisation particulier, ladite entité est un serveur de clés, comprenant par exemple des paires clé-valeur.
- [0068] Selon un autre aspect de l'invention, il est proposé un dispositif client connecté à un réseau, le dispositif client comprenant :
- un module d'émission pour transmettre des requêtes d'accès à des données ; et
 - un module de réception pour recevoir des réponses auxdites requêtes.
- [0069] Selon un aspect de l'invention, il est proposé un programme d'ordinateur comprenant instructions pour la mise en œuvre des étapes du procédé de traitement de requêtes tel que décrit ci-dessus lorsque les instructions sont exécutées par un ou plusieurs processeurs.
- [0070] Le programme d'ordinateur peut être formé d'une ou plusieurs sous-parties stockées dans une même mémoire ou dans des mémoires distinctes. Le programme peut utiliser n'importe quel langage de programmation, et être sous la forme de code source, code objet, ou de code intermédiaire entre code source et code objet, tel que dans une forme partiellement compilée, ou dans n'importe quelle autre forme souhaitable.
- [0071] Selon un mode de réalisation particulier, le programme d'ordinateur comprend des instructions de code eBPF (acronyme de l'expression anglo-saxonne « extended Berkeley Packet Filter »).
- [0072] Ce mode de réalisation particulier permet d'utiliser la technologie eBPF pour développer les différents niveaux de cache. La technologie eBPF permet notamment d'exécuter des instructions de codes dans l'espace noyau lors de l'exécution de pilotes informatiques, par exemple en exploitant un point d'attache des pilotes informatiques tel que décrit ci-dessus.
- [0073] Selon un aspect de l'invention, il est proposé un support d'informations lisible par ordinateur comprenant un programme d'ordinateur tel que décrit ci-dessus.
- [0074] Le support d'informations peut être n'importe quelle entité ou dispositif capable de stocker le programme. Par exemple, le support peut comporter un moyen de stockage, tel qu'une mémoire non-volatile ou ROM, par exemple un CD-ROM ou une ROM de circuit microélectronique, ou encore un moyen d'enregistrement magnétique, par exemple une disquette ou un disque dur. D'autre part, le support de stockage peut être un support transmissible tel qu'un signal électrique ou optique, qui peut être acheminé via un câble électrique ou optique, par radio ou par un réseau de télécommunication ou par un réseau informatique ou par d'autres moyens. Le programme selon l'invention peut être en particulier téléchargé sur un réseau informatique. Alternativement, le support d'informations peut être un circuit intégré dans lequel le programme est incorporé, le circuit étant adapté pour exécuter ou pour être utilisé dans l'exécution du

procédé en question.

Breve description des dessins

[0075] D'autres caractéristiques et avantages de la présente invention ressortiront de la description fournie ci-après de modes de réalisation particuliers. Les modes de réalisation sont donnés à titre d'exemple illustratif et sont dépourvus de tout caractère limitatif. La description fournie ci-après est illustrée par les dessins ci-joints :

[Fig.1A] la [Fig.1A] représente schématiquement des exemples de requêtes d'accès à des données entre des dispositifs clients et une entité de traitement de requêtes d'accès à des données selon un mode de réalisation particulier ;

[Fig.1B] la [Fig.1B] représente schématiquement des exemples de requêtes d'accès à des données entre un dispositif client et une entité de traitement de requêtes d'accès à des données selon un mode de réalisation particulier ;

[Fig.1C] la [Fig.1C] représente schématiquement des exemples de structures de données comprises dans une entité de traitement de requêtes d'accès à des données selon un mode de réalisation particulier ;

[Fig.2A] la [Fig.2A] représente, sous forme d'ordinogramme, des étapes d'un procédé de traitement de requêtes d'accès à des données selon un mode de réalisation particulier ;

[Fig.2B] la [Fig.2B] représente, sous forme d'ordinogramme, des étapes d'un procédé de traitement de requêtes d'accès à des données selon un mode de réalisation particulier ;

[Fig.3A] la [Fig.3A] représente schématiquement un exemple d'architecture fonctionnelle d'une entité de traitement de requêtes d'accès à des données selon un mode de réalisation particulier ;

[Fig.3B] la [Fig.3B] représente schématiquement un exemple d'architecture logicielle et matérielle d'une entité de traitement de requêtes d'accès à des données selon un mode de réalisation particulier ;

[Fig.4] la [Fig.4] représente, sous forme d'ordinogramme, des étapes d'un procédé de traitement de requêtes d'accès à des données selon un mode de réalisation particulier ;

[Fig.5A] la [Fig.5A] représente schématiquement un exemple d'architecture fonctionnel d'une entité de traitement de requêtes d'accès à des données selon un mode de réalisation particulier ;

[Fig.5B] la [Fig.5B] représente schématiquement un exemple d'architecture logicielle et matérielle d'une entité de traitement de requêtes d'accès à des données selon un mode de réalisation particulier ;

[Fig.6A] la [Fig.6A] représente, sous forme d'ordinogramme, des étapes d'un

procédé de traitement de requêtes d'accès à des données selon un mode de réalisation particulier ;

[Fig.6B] la [Fig.6B] représente, sous forme d'ordinogramme, des étapes d'un procédé de traitement de requêtes d'accès à des données selon un mode de réalisation particulier ;

[Fig.6C] la [Fig.6C] représente, sous forme d'ordinogramme, des étapes d'un procédé de traitement de requêtes d'accès à des données selon un mode de réalisation particulier ;

[Fig.7A] la [Fig.7A] représente schématiquement un exemple de structures de données comprises dans une entité de traitement de requêtes d'accès à des données selon un mode de réalisation particulier ;

[Fig.7B] la [Fig.7B] représente schématiquement un exemple de structures de données comprises dans une entité de traitement de requêtes d'accès à des données selon un mode de réalisation particulier ;

[Fig.7C] la [Fig.7C] représente schématiquement un exemple de structures de données comprises dans une entité de traitement de requêtes d'accès à des données selon un mode de réalisation particulier ;

[Fig.8A] la [Fig.8A] représente, sous forme d'ordinogramme, des étapes d'un procédé de traitement de requêtes d'accès à des données selon un mode de réalisation particulier ;

[Fig.8B] la [Fig.8B] représente, sous forme d'ordinogramme, des étapes d'un procédé de traitement de requêtes d'accès à des données selon un mode de réalisation particulier ;

[Fig.8C] la [Fig.8C] représente, sous forme d'ordinogramme, des étapes d'un procédé de traitement de requêtes d'accès à des données selon un mode de réalisation particulier ;

[Fig.9] la [Fig.9] représente schématiquement des étapes d'un procédé de traitement de requêtes d'accès à des données selon un mode de réalisation particulier ;

Description des modes de réalisation

[0076] La présente invention concerne un procédé de traitement de requêtes d'accès à des données, une entité, un programme d'ordinateur, et un support d'informations associés.

[0077] La [Fig.1A] représente schématiquement des exemples de requêtes d'accès à des données entre des dispositifs clients et une entité de traitement de requêtes d'accès à des données selon un mode de réalisation particulier.

[0078] Tel qu'illustré par la [Fig.1A], les dispositifs clients CLT sont connectés par l'intermédiaire d'un réseau informatique de communication NET à une entité SERV de

traitement de requêtes d'accès à des données. L'entité SERV comprend des données d'une application. À titre d'exemple, l'entité SERV peut être un serveur de stockage de données en réseau. L'entité SERV peut également mettre en œuvre un serveur DNS de résolution de noms de domaine comprenant des URLs et des adresses IP. L'entité SERV peut mettre en œuvre un serveur de clés comprenant des paires clé-valeur. Selon un mode de réalisation particulier, l'application APP est une application de stockage de paires clé-valeur et peut par exemple être l'application Memcached, une application bien connue de l'homme du métier et couramment déployée. Les dispositifs clients CLT peuvent être des équipements utilisateurs, des terminaux mobiles, des objets communicants, des serveurs d'application, ou tout autre dispositif émetteur de requêtes d'accès à des données. Les requêtes d'accès à des données émises par les dispositifs clients CLT sont des demandes d'accès en écriture SET d'une donnée dans l'entité SERV, ou des demandes d'accès en lecture GET d'une donnée dans l'entité SERV. L'entité SERV de traitement de requêtes reçoit en provenance des dispositifs clients CLT des demandes d'accès en écriture ou en lecture de données GET, SET et traite ces demandes. De plus, l'entité SERV peut émettre à destination des dispositifs clients CLT des réponses REP_GET aux requêtes d'accès à des données GET, SET.

[0079] Dans le mode de réalisation particulier décrit ici, le réseau informatique de communication peut être de type Internet mettant par exemple en œuvre le protocole de transport IP, les moyens de communication des dispositifs clients CLT et de l'entité SERV s'appuyant notamment sur une interface de communication apte aux échanges de données. Aucune limitation n'est attachée à la nature de l'interface de communication entre les dispositifs clients CLT et l'entité SERV, qui peut être filaire ou non filaire, et peut mettre en œuvre tout protocole connu de l'homme du métier (Ethernet, Wi-Fi, Bluetooth, 3G, 4G, 5G, 6G, etc.). En outre, le réseau informatique ne se limite pas à un réseau Internet, tout autre réseau (propriétaire, etc.) pouvant être envisagé, de sorte que l'entité SERV de traitement de requêtes puisse être accessible par l'intermédiaire du réseau NET.

[0080] La [Fig.1B] représente schématiquement des exemples de requêtes d'accès à des données entre un dispositif client et une entité de traitement de requêtes d'accès à des données selon un mode de réalisation particulier.

[0081] Comme le montre la [Fig.1B], les requêtes GET_A, SET_B d'accès à des données sont émises par un dispositif client CLT à destination de l'entité SERV de traitement de requêtes. Les requêtes d'accès à des données peuvent être des requêtes d'accès en lecture GET_A de données, et également être des requêtes d'accès en écriture SET_B de données. Une requête de lecture GET_A émise par un dispositif client CLT à destination de l'entité SERV comprend un identifiant ID_A d'une donnée D_A. À titre d'exemple, un identifiant peut être une URL et une donnée peut être une adresse IP.

Suite au traitement de la requête de lecture GET_A, et en réponse à celle-ci, l'entité SERV émet une réponse REP_GET_A. La réponse REP_GET_A peut comprendre la donnée D_A demandée par le dispositif client CLT ou peut comprendre un message d'erreur ERR. Un message d'erreur ERR peut être transmis lorsque l'entité SERV ne comprend pas la donnée demandée, ou n'est pas en mesure de traiter une requête, par exemple suite à un dysfonctionnement, un état de congestion, etc. Une requête d'écriture SET_B émise par un dispositif client CLT à destination de l'entité SERV comprend un identifiant ID_B d'une donnée à écrire D_B, et éventuellement la donnée à écrire D_B. L'identifiant ID_B peut être identique ou différent de l'identifiant ID_A. Suite au traitement de la requête d'écriture SET_B, l'entité SERV peut émettre ou non une réponse REP_SET_B à destination du dispositif client CLT. La réponse REP_SET_B peut par exemple comprendre un accusé de réception ACK.

- [0082] Selon un mode de réalisation particulier, les requêtes d'accès en lecture de données GET_A et leurs réponses REP_GET_A sont transmises par l'intermédiaire du réseau informatique NET mettant en œuvre un protocole réseau IP et un protocole de transport de type UDP. Le protocole de transport UDP permet de transmettre des données de manière simple et rapide entre deux entités en mode non connecté.
- [0083] Selon un mode de réalisation particulier, les requêtes d'accès en écriture de données SET_B et leurs réponses REP_SET_B sont transmises par l'intermédiaire du réseau informatique NET mettant en œuvre un protocole réseau IP et un protocole de transport de type TCP.
- [0084] La [Fig.1C] représente schématiquement des exemples de structures de données comprises dans une entité de traitement de requêtes d'accès à des données selon un mode de réalisation particulier.
- [0085] Dans la suite de la présente description, les signes de référence suivants sont utilisés : le signe de référence D_A désigne une donnée correspondant à l'identifiant ID_A ; les signes de référence D_A1, D_A2, D_A3 désignent des données valides correspondant à l'identifiant ID_A et enregistrées respectivement dans une première mémoire cache C1, une deuxième mémoire cache C2, et une mémoire C3 de l'application APP ; le signe de référence D_A1' désigne une donnée correspondant à l'identifiant ID_A et invalide dans la première mémoire cache C1.
- [0086] Tel qu'illustré par la [Fig.1C], une donnée D_A1, D_A2, D_A3 de l'entité SERV est identifiée par un identifiant ID_A. L'entité SERV peut comprendre plusieurs mémoires C1, C2, C3. En particulier, l'entité SERV peut comprendre : une carte réseau NIC comprenant une première mémoire cache C1 et un noyau de système d'exploitation KER comprenant une deuxième mémoire cache C2. Les données stockées dans la première mémoire cache C1 et dans la deuxième mémoire cache C2 sont des répliques de données de l'application APP. L'entité SERV peut en outre comprendre

l'application APP, située dans une couche d'application de l'entité SERV et mise en œuvre dans l'espace utilisateur, l'application APP comprenant la mémoire d'application C3. L'application APP est configurée pour traiter et répondre à des requêtes d'accès GET_A, SET_B, et en particulier pour gérer des accès concurrents. Par exemple, l'application APP est en mesure de traiter plusieurs requêtes d'écriture SET_B émises par des dispositifs clients CLT différents et portant sur la même donnée D_B, et une requête de lecture GET_A portant sur cette donnée D_B.

- [0087] Dans le mode de réalisation particulier décrit ici, l'entité SERV assure la cohérence entre les données des mémoires caches C1, C2 et les données de l'application APP. Afin d'assurer la cohérence entre ces différentes mémoires C1, C2, C3, l'entité SERV identifie les données D_A1, D_A2 comprises dans les mémoires caches C1, C2 soit comme étant valides D_A1, D_A2, soit comme étant invalides D_A1', D_A2'. Pour un identifiant ID_A, une donnée valide D_A1, D_A2 est nécessairement identique à la donnée D_A3 de l'application APP et peut être utilisée pour répondre à une requête de lecture GET_A comprenant l'identifiant ID_A. Pour un identifiant ID_A, une donnée invalide D_A1', D_A2' n'est pas nécessairement identique à la donnée D_A3 de l'application APP et ne peut pas être utilisée pour répondre à une requête de lecture GET_A comprenant l'identifiant ID_A.
- [0088] La [Fig.2A] représente, sous forme d'ordinogramme, des étapes d'un procédé de traitement de requêtes d'accès à des données selon un mode de réalisation particulier.
- [0089] Dans le mode de réalisation particulier décrit ici, le procédé est mis en œuvre par l'entité SERV connectée au réseau NET. Tel qu'illustré par la [Fig.2A], le procédé comprend les étapes décrites ci-après.
- [0090] Dans la suite de la présente description, les étapes réalisées par la carte réseau NIC sont différenciées des étapes réalisées par le noyau KER. La carte réseau NIC met en œuvre les étapes dont les signes de référence commencent par la lettre N, tandis que le noyau KER met en œuvre les étapes dont les signes de référence commencent par la lettre K.
- [0091] À l'étape N20, la carte réseau NIC intercepte un paquet réseau à destination de l'application APP, le paquet réseau portant une requête de lecture GET_A. La requête GET_A comprend un identifiant ID_A d'une donnée à lire D_A. En particulier, la carte réseau NIC filtre l'ensemble des paquets reçus en provenance du réseau en inspectant leur port de destination afin de traiter uniquement les requêtes destinées à l'application APP.
- [0092] À l'étape N30, la carte réseau NIC détermine si la requête GET_A doit être transmise au noyau KER en fonction de la présence d'une donnée D_A1 correspondant à l'identifiant ID_A dans la première mémoire cache C1 de la carte réseau NIC. Si oui, le procédé se poursuit à l'étape N50, sinon à l'étape N130.

- [0093] À l'étape N50, la carte réseau NIC transmet la requête de lecture GET_A au noyau KER.
- [0094] À l'étape N130, la carte réseau NIC émet une réponse REP_GET_A à la requête de lecture GET_A. La réponse REP_GET_A émise est déterminée en fonction de la présence d'une donnée correspondant à l'identifiant ID_A dans la première mémoire cache C1 de la carte réseau NIC ou dans la deuxième mémoire cache C2 du noyau KER.
- [0095] Selon un mode de réalisation, la carte réseau NIC détermine à l'étape N30 si la première mémoire cache C1 comprend une entrée mémoire pour l'identifiant ID_A. Si la première mémoire cache C1 ne comprend pas d'entrée mémoire pour l'identifiant ID_A, la carte réseau NIC transmet à l'étape N50 la requête GET_A au noyau KER.
- [0096] Selon un mode de réalisation, combiné ou non avec le précédent, la carte réseau NIC détermine à l'étape N30 si la première mémoire cache C1 comprend une donnée D_A1 correspondant à l'identifiant ID_A. Si la première mémoire cache C1 comprend la donnée D_A1 demandée, il n'est pas nécessaire de transmettre la requête GET_A au noyau KER. La carte réseau NIC émet à l'étape N130 la réponse REP_GET_A comprenant la donnée D_A1 lue dans la première mémoire cache C1. Et, si la première mémoire cache C1 ne comprend la donnée D_A1 demandée, la carte réseau NIC transmet à l'étape N50 la requête GET_A au noyau KER.
- [0097] Selon un mode de réalisation, si la requête GET_A est transmise au noyau KER, le noyau KER traite la requête GET_A en fonction de la présence d'une donnée D_A2 correspondant à l'identifiant ID_A dans la deuxième mémoire cache C2. À titre d'exemple, le traitement de la requête GET_A par le noyau KER peut être analogue au traitement mis en œuvre par la carte réseau NIC décrit ci-dessus. Ainsi, dans ce mode de réalisation, la réponse REP_GET_A émise à l'étape N130 est déterminée en fonction de la présence de la donnée D_A2 demandée dans la deuxième mémoire cache C2.
- [0098] Ainsi, et tel que mentionné ci-dessus, l'étape de transmission N50 est réalisée en fonction de la présence d'une donnée D_A1 correspondant à l'identifiant ID_A dans la première mémoire cache C1 ; et la réponse REP_GET_A à la requête GET_A émise par l'entité SERV est déterminée en fonction de la présence d'une donnée D_A1, D_A2 correspondant à l'identifiant ID_A dans la première mémoire C1 ou dans la deuxième mémoire cache C2.
- [0099] La [Fig.2B] représente, sous forme d'ordinogramme, des étapes d'un procédé de traitement de requêtes d'accès à des données selon un mode de réalisation particulier. Tel qu'illustré par la [Fig.2B], le procédé comprend les étapes décrites ci-après.
- [0100] À l'étape N10, la carte réseau NIC reçoit, sur son interface réseau et en provenance d'un dispositif client CLT connecté au réseau NET, un paquet réseau portant une

requête de lecture GET_A, la requête GET_A comprenant un identifiant ID_A d'une donnée à lire D_A.

- [0101] À l'étape N20, la carte réseau NIC intercepte le paquet réseau portant la requête de lecture GET_A.
- [0102] À l'étape N30, la carte réseau NIC détermine si la première mémoire cache C1 comprend une donnée valide D_A1 correspondant à l'identifiant ID_A.
- [0103] Si la première mémoire cache C1 comprend la donnée valide D_A1, à l'étape N40, la carte réseau NIC lit la donnée valide D_A1 dans la première mémoire cache C1. Le traitement de la requête de lecture GET_A par la carte réseau NIC permet d'éviter un transfert de la requête au noyau KER. Le procédé se poursuit à l'étape N130.
- [0104] Si la première mémoire cache C1 ne comprend pas de donnée valide correspondant à l'identifiant ID_A, à l'étape N50, la carte réseau NIC transmet la requête de lecture GET_A au noyau KER.
- [0105] À l'étape K60, le noyau KER détermine si la deuxième mémoire cache C2 comprend une donnée valide D_A2 correspondant à l'identifiant ID_A.
- [0106] Si la deuxième mémoire cache C2 comprend la donnée valide D_A2, à l'étape K70, le noyau KER lit la donnée valide D_A2 dans la deuxième mémoire cache C2. Le traitement de la requête de lecture GET_A par le noyau KER permet d'éviter un transfert de la requête à un niveau supérieur de la pile réseau.
- [0107] À l'étape N130, la carte réseau NIC émet une réponse REP_GET_A à la requête de lecture GET_A. La réponse REP_GET_A peut être émise à destination du dispositif client CLT émetteur de la requête et comprendre la donnée valide D_A1, D_A2 correspondant à l'identifiant ID_A ou un message d'erreur ERR.
- [0108] Selon un mode de réalisation particulier, suite à l'étape K 70, le noyau KER transmet à la carte réseau NIC la réponse REP_GET_A à la requête de lecture GET_A, cette réponse comprenant la donnée valide D_A2 correspondant à l'identifiant ID_A et lue dans la deuxième mémoire cache C2.
- [0109] Par ailleurs, le deuxième niveau de cache mis en œuvre par le noyau KER est particulièrement adapté aux architectures de type multi-cœurs le noyau KER pouvant exploiter les cartes réseaux multi-queues afin de traiter en parallèle plusieurs requêtes sur les différents cœurs.
- [0110] La [Fig.3A] représente schématiquement un exemple d'architecture fonctionnelle d'une entité de traitement de requêtes d'accès à des données selon un mode de réalisation particulier.
- [0111] Comme le montre la [Fig.3A], l'entité SERV comprend une carte réseau NIC et un noyau de système d'exploitation KER. L'entité SERV est connectée au réseau informatique NET par l'intermédiaire de sa carte réseau NIC. Les requêtes d'accès GET_A, SET_B sont reçues par la carte réseau NIC en provenance du réseau NET ; et

les réponses REP_GET_A, REP_SET_B sont émises par la carte réseau NIC à destination du réseau NET. La carte réseau NIC et le noyau KER comprennent les modules décrits ci-après.

- [0112] Le terme module peut correspondre aussi bien à un composant logiciel qu'à un composant matériel ou un ensemble de composants matériels et logiciels, un composant logiciel correspondant lui-même à un ou plusieurs programmes ou sous-programmes d'ordinateur ou de manière plus générale à tout élément d'un programme apte à mettre en œuvre une fonction ou un ensemble de fonctions telles que décrites pour les modules concernés. De la même manière, un composant matériel correspond à tout élément d'un ensemble matériel (ou hardware) apte à mettre en œuvre une fonction ou un ensemble de fonctions pour le module concerné (circuit intégré, carte à puce, carte à mémoire, etc.).
- [0113] Tel qu'illustré par la [Fig.3A], la carte réseau NIC comprend les modules suivants : un module de réception réseau MOD_RX1 pour recevoir et intercepter une requête de lecture GET_A comprenant un identifiant ID_A d'une donnée D_A ; un premier module de détermination MOD_DET1 pour déterminer si la première mémoire cache C1 comprend une donnée valide D_A1 correspondant à l'identifiant ID_A ; un premier module de lecture MOD_RD1 pour lire la donnée valide D_A1 correspondant à l'identifiant ID_A dans la première mémoire cache C1 ; un premier module de transmission MOD_FWD1 pour transmettre la requête de lecture GET_A au noyau KER ; et un module d'émission réseau MOD_TX1 pour transmettre une réponse REP_GET_A à la requête de lecture GET_A. La carte réseau NIC comprend en outre un premier module de réception pour recevoir en provenance du noyau KER une donnée valide D_A2, D_A3.
- [0114] Tel qu'illustré par la [Fig.3A], le noyau KER comprend les modules suivants : un deuxième module de détermination MOD_DET2 pour déterminer si la deuxième mémoire cache C2 comprend une donnée valide D_A2 correspondant à l'identifiant ID_A ; et un deuxième module de lecture MOD_RD2 compris dans le noyau KER pour lire ladite donnée valide D_A2 dans la deuxième mémoire cache C2. Le noyau KER comprend en outre un deuxième module d'émission pour transmettre une réponse REP_GET_A à une requête GET_A à la carte réseau NIC.
- [0115] Selon un mode de réalisation particulier, la carte réseau NIC et le noyau KER sont connectés par l'intermédiaire d'un bus de communication mettant en œuvre le protocole PCIe (acronyme de l'expression anglo-saxonne « Peripheral Component Interconnect Express »).
- [0116] La [Fig.3B] représente schématiquement un exemple d'architecture logicielle et matérielle d'une entité de traitement de requêtes d'accès à des données selon un mode de réalisation particulier.

- [0117] Tel qu'illustré sur la [Fig.3B], l'entité SERV comprend une carte réseau NIC et un noyau de système d'exploitation KER. La carte réseau NIC comprend notamment : une mémoire cache C1 ; une unité de traitement ou processeur PROC1 ; et une mémoire MEM1. Bien entendu, la carte réseau NIC comprend des interfaces et un module de communication pour d'une part échanger des données avec d'autres dispositifs par l'intermédiaire du réseau NET et d'autre part pour échanger des données avec le noyau KER. La carte réseau NIC constitue une interface physique entre l'entité SERV et le réseau NET. La carte réseau NIC permet de préparer, d'envoyer et de contrôler les données circulant sur le réseau NET.
- [0118] Selon un mode de réalisation particulier, la carte réseau NIC est une carte réseau programmable. Ainsi, la carte réseau NIC dispose de l'architecture matérielle d'un ordinateur. La carte réseau NIC comporte, notamment, un processeur PROC1, une mémoire vive RAM1, une mémoire morte MEM1 et une mémoire non volatile ROM1.
- [0119] La mémoire MEM1 de la carte réseau NIC constitue un support d'informations ou d'enregistrement conforme à la présente invention, lisible par ordinateur et par le processeur PROC1 et sur lequel est enregistré un programme d'ordinateur PROG1 conforme à l'invention. Le programme d'ordinateur PROG1 comporte des instructions pour la mise en œuvre des étapes réalisées par la carte réseau NIC d'un procédé de traitement de requêtes d'accès à des données, lorsque le programme d'ordinateur PROG1 est exécuté par le processeur PROC1. Le programme d'ordinateur PROG1 définit les modules fonctionnels représentés par la [Fig.3A] de la carte réseau NIC, qui s'appuient ou commandent les éléments matériels de cette dernière.
- [0120] Tel qu'illustré sur la [Fig.3B], le noyau KER comprend notamment une mémoire cache C2. L'entité SERV comprend une unité de traitement ou processeur PROC2 et une mémoire MEM2 associés au noyau KER. Bien entendu, le noyau KER comprend une interface et un module de communication pour échanger des données avec la carte réseau NIC. En outre, le noyau KER peut comprendre une interface pour échanger des données avec l'application APP.
- [0121] Selon un mode de réalisation particulier, l'entité SERV dispose de l'architecture matérielle d'un ordinateur. Le noyau KER peut ainsi être associé à, notamment, un processeur PROC2, une mémoire vive RAM2, une mémoire morte MEM2 et une mémoire non volatile ROM2.
- [0122] La mémoire MEM2 associée au noyau KER constitue un support d'informations ou d'enregistrement conforme à l'invention, lisible par ordinateur et par le processeur PROC2 et sur lequel est enregistré un programme d'ordinateur PROG2 conforme à l'invention. Le programme d'ordinateur PROG2 comporte des instructions pour la mise en œuvre des étapes réalisées par le noyau KER d'un procédé de traitement de requêtes d'accès à des données selon l'invention, lorsque le programme d'ordinateur

PROG2 est exécuté par le processeur PROC2. Le programme d'ordinateur PROG2 définit les modules fonctionnels représentés par la [Fig.3A] du noyau KER, qui s'appuient ou commandent les éléments matériels de ce dernier.

- [0123] La mémoire MEM2 associée au noyau KER peut en outre constituer un support d'informations ou d'enregistrement conforme à l'invention, lisible par ordinateur et sur lequel est enregistré un programme d'ordinateur PROG12 conforme à l'invention. Le programme d'ordinateur PROG12 comporte des premières instructions pour la mise en œuvre des étapes réalisées par la carte réseau NIC d'un procédé de traitement de requêtes d'accès à des données selon l'invention, lorsque les premières instructions sont exécutées par un premier processeur PROC1. Le programme d'ordinateur PROG12 comporte en outre des deuxièmes instructions pour la mise en œuvre des étapes réalisées par le noyau KER d'un procédé de traitement de requêtes d'accès à des données selon l'invention, lorsque les deuxièmes instructions sont exécutées par un deuxième processeur PROC2. Le programme d'ordinateur PROG12 comprend ainsi des instructions pour la mise en œuvre des étapes d'un procédé de traitement de requête selon l'invention, lorsque le programme d'ordinateur est exécuté par un ordinateur. Le programme PROG12 définit les modules fonctionnels représentés par la [Fig.3A] de la carte réseau NIC et du noyau KER, qui s'appuient ou commandent les éléments matériels de ce dernier.
- [0124] Typiquement, les programmes d'ordinateur selon l'invention peuvent comprendre des instructions de code eBPF, la technologie eBPF étant ainsi utilisée pour développer les différents niveaux de cache. La technologie eBPF permet notamment d'exécuter des instructions de codes dans l'espace noyau.
- [0125] Selon un mode de réalisation particulier, le procédé de traitement de requêtes comprend le chargement ou l'enregistrement du programme d'ordinateur PROG1 dans la carte réseau NIC. Par exemple, le programme d'ordinateur PROG1 peut être chargé dans la carte réseau NIC par l'entité SERV et exécuté par la carte réseau NIC pour mettre en œuvre un procédé de traitement de requêtes tel que proposé.
- [0126] Selon un mode de réalisation particulier, le procédé de traitement de requêtes comprend une étape de chargement ou d'enregistrement du programme d'ordinateur PROG2 dans le noyau KER. Par exemple, le programme d'ordinateur PROG2 peut être chargé dans le noyau KER par l'entité SERV et exécuté par le noyau KER pour mettre en œuvre un procédé de traitement de requêtes tel que proposé. Plus particulièrement, le programme d'ordinateur PROG2 peut être exécuté par les pilotes informatiques de la carte réseau NIC installés dans le noyau KER.
- [0127] La [Fig.4] représente, sous forme d'ordinogramme, des étapes d'un procédé de traitement de requêtes d'accès à des données selon un mode de réalisation particulier.
- [0128] En particulier, le procédé illustré par la [Fig.4] en comparaison du procédé illustré

par la [Fig.2B] comprend les étapes supplémentaires suivantes.

- [0129] Si, suite à l'étape K60 mise en œuvre par le noyau KER, la deuxième mémoire cache C2 ne comprend pas de données valides D_A2 correspondant à l'identifiant ID_A, le noyau KER transmet à l'étape K80 la requête de lecture GET_A à l'application APP.
- [0130] À l'étape K90, le noyau KER reçoit une réponse REP_GET_A à la requête GET_A en provenance de l'application APP, cette réponse comprenant une donnée D_A3 correspondant à l'identifiant ID_A, qui est dite valide par la suite. Le noyau KER intercepte ainsi la donnée valide D_A3 émise par l'application APP en réponse à la requête GET_A.
- [0131] À l'étape N120, la carte réseau NIC reçoit la réponse REP_GET_A en provenance du noyau KER et transmet cette réponse à l'étape N130 décrite précédemment.
- [0132] La [Fig.5A] représente schématiquement un exemple d'architecture fonctionnelle d'une entité de traitement de requêtes d'accès à des données selon un mode de réalisation particulier.
- [0133] Tel qu'illustré par la [Fig.5A], la carte réseau NIC comprend, en comparaison de l'architecture illustrée par la [Fig.3A], les modules supplémentaires suivants : un premier module de réception MOD_RCV1 pour recevoir en provenance du noyau KER une donnée valide D_A2, D_A3 correspondant à l'identifiant ID_A.
- [0134] Tel qu'illustré par la [Fig.5A], le noyau KER comprend, en comparaison de l'architecture illustrée par la [Fig.3A], les modules supplémentaires suivants : un deuxième module de transmission MOD_FWD2 pour transmettre la requête GET_A à l'application APP ; et un deuxième module de réception MOD_RCV2 pour recevoir en provenance de l'application APP une réponse REP_GET_A et intercepter la donnée valide D_A3 correspondant à l'identifiant ID_A comprise dans cette réponse.
- [0135] Tel qu'illustré par la [Fig.5A], l'application APP comprend : une mémoire d'application C3 pour enregistrer les données de l'application APP ; et un module de traitement et de réponse PROC_REP pour traiter et répondre à des requêtes d'accès à des données GET_A, SET_B.
- [0136] Bien entendu, aucune limitation n'est attachée au fait que l'application APP soit intégrée dans l'entité SERV elle-même. L'application APP peut également être déportée à distance et mise en œuvre par une autre entité, cette autre entité étant dotée des composants matériels et logiciels d'un ordinateur, notamment : un processeur, une mémoire vive, une mémoire morte et mémoire non-volatile. L'entité SERV est ainsi située sur un plan intermédiaire entre un dispositif client CLT et l'application APP.
- [0137] La [Fig.5B] représente schématiquement un exemple d'architecture logicielle et matérielle d'une entité de traitement de requêtes d'accès à des données selon un mode de réalisation particulier.
- [0138] Tel qu'illustré sur la [Fig.5B], l'entité SERV comprend une carte réseau NIC, un

noyau KER, et une application APP. En particulier, l'architecture illustrée par la [Fig.5B] comprend en comparaison de l'architecture illustrée par la [Fig.3B] les composants matériels et logiciels supplémentaires suivants.

[0139] Tel qu'illustré sur la [Fig.5B], l'entité SERV comprend une application APP.

L'application APP comprend une mémoire C3 permettant de stocker les données de l'application APP.

[0140] Dans le mode de réalisation particulier décrit ici, la mémoire MEM2 associée au noyau KER constitue un support d'informations ou d'enregistrement conforme à l'invention, lisible par ordinateur et par le processeur PROC2 et sur lequel est enregistré un programme d'ordinateur PROG3 conforme à l'invention. Le programme d'ordinateur PROG3 comporte des instructions pour la mise en œuvre des étapes réalisées par l'application APP d'un procédé de traitement de requêtes d'accès à des données selon l'invention, lorsque le programme d'ordinateur PROG3 est exécuté par le deuxième processeur PROC2. Le programme d'ordinateur PROG3 définit les modules fonctionnels représentés par la [Fig.5A] de l'application APP, qui s'appuient ou commandent les éléments matériels de cette dernière.

[0141] La mémoire MEM2 associée au noyau KER peut en outre constituer un support d'informations ou d'enregistrement conforme à l'invention, lisible par ordinateur et sur lequel est enregistré un programme d'ordinateur PROG123 conforme à l'invention. Le programme d'ordinateur PROG123 comporte les instructions du programme d'ordinateur PROG12 et en outre les instructions du programme d'ordinateur PROG3. Le programme d'ordinateur PROG123 comprend ainsi des instructions pour la mise en œuvre des étapes d'un procédé de traitement de requête selon l'invention, lorsque le programme d'ordinateur est exécuté par un ordinateur. Le programme d'ordinateur PROG123 définit les modules fonctionnels représentés par la [Fig.5A] de la carte réseau NIC, du noyau KER et de l'application APP, qui s'appuient ou commandent les éléments matériels de ce dernier.

[0142] Typiquement, les données de l'application APP comprises dans la mémoire C3 sont stockées dans la mémoire vive RAM2 de l'entité SERV et accédées par l'application APP depuis l'espace utilisateur ; et les données du noyau KER comprises dans la deuxième mémoire cache C2 sont stockées dans mémoire vive RAM2 de l'entité SERV et accédées par le noyau KER depuis l'espace noyau.

[0143] La [Fig.6A] représente, sous forme d'ordinogramme, des étapes d'un procédé de traitement de requêtes d'accès à des données selon un mode de réalisation particulier. En particulier, le procédé illustré par la [Fig.6A] comprend en comparaison de la [Fig.4] les étapes supplémentaires décrites ci-après.

[0144] À l'étape K90, le noyau KER intercepte une réponse REP_GET_A à la requête de lecture GET_A, cette réponse comprenant une donnée valide D_A3. Le noyau KER

peut utiliser la donnée valide D_A3 reçue pour enregistrer une nouvelle donnée dans la mémoire cache C1, C2 ou bien remplacer une donnée enregistrée D_C1, D_C2 dans la mémoire cache C1, C2.

- [0145] À l'étape K100, le noyau KER peut enregistrer dans la deuxième mémoire cache C2 la donnée valide D_A3 reçue de l'application APP. Le noyau KER transmet à destination de la carte réseau NIC la réponse REP_GET_A à la GET_A.
- [0146] À l'étape K110, le noyau KER peut enregistrer dans la première mémoire cache C1 la donnée valide D_A3 reçue de l'application APP.
- [0147] À l'étape N120, la carte réseau NIC reçoit en provenance du noyau KER la réponse REP_GET_A qui est émise à l'étape N130.
- [0148] Dans un mode de réalisation particulier, le noyau KER peut également enregistrer dans la première mémoire cache C1 une donnée valide D_A2 émise en réponse à une requête GET_A. En effet, à l'étape K70, le noyau KER lit la donnée valide D_A2 dans la deuxième mémoire cache C2. Suite à cette étape, le noyau KER peut enregistrer la donnée valide D_A2 dans la première mémoire cache C1 avant de transmettre la réponse REP_GET_A à la requête GET_A à la carte réseau NIC.
- [0149] La [Fig.6B] représente, sous forme d'ordinogramme, des étapes d'un procédé de traitement de requêtes d'accès à des données selon un mode de réalisation particulier. En particulier, le procédé illustré par la [Fig.6B] comprend en comparaison du procédé illustré par la [Fig.6A] les étapes supplémentaires suivantes.
- [0150] Suite à l'étape K90 d'interception, par le noyau KER d'une donnée D_A3 valide correspondant à l'identifiant ID_A, le noyau KER met en œuvre une étape K101.
- [0151] À l'étape K101, le noyau KER détermine si la deuxième mémoire cache C2 comprend une donnée invalide D_A2' correspondant à l'identifiant ID_A.
- [0152] À l'étape K102, si la deuxième mémoire C2 comprend la donnée D_A2', le noyau KER enregistre dans la deuxième mémoire cache C2 la donnée valide D_A3 reçue de l'application APP et correspondant à l'identifiant ID_A.
- [0153] À l'étape K111, le noyau KER détermine si la première mémoire cache C1 comprend une donnée D_A1' correspondant à l'identifiant ID_A.
- [0154] À l'étape K112, si la première mémoire C1 comprend la donnée D_A1', le noyau KER enregistre dans la première mémoire cache C1 la donnée valide D_A3 reçue de l'application APP et correspondant à l'identifiant ID_A.
- [0155] La [Fig.6C] représente, sous forme d'ordinogramme, des étapes d'un procédé de traitement de requêtes d'accès à des données selon un mode de réalisation particulier. En particulier, le procédé illustré par la [Fig.6C] comprend en comparaison du procédé illustré par la [Fig.6A] les étapes supplémentaires suivantes.
- [0156] Suite à l'étape K90 d'interception, par le noyau KER d'une donnée D_A3 valide correspondant à l'identifiant ID_A, le noyau KER met en œuvre au moins une des étapes

K103 et K104.

- [0157] À l'étape K103, le noyau KER compare l'identifiant ID_A à un identifiant ID_C d'une donnée enregistrée D_C2 dans la deuxième mémoire cache C2.
- [0158] À l'étape K104, le noyau KER compare un seuil THR_2 à un compteur CNT_C2 associé à une donnée enregistrée D_C2 dans la deuxième mémoire cache C2 correspondant à un identifiant ID_C.
- [0159] À l'étape K105, en fonction du résultat d'au moins une des comparaisons réalisées aux étapes K103 et K104, le noyau KER enregistre ou non la donnée valide D_A3 reçue de l'application APP et correspondant à l'identifiant ID_A.
- [0160] Selon un mode de réalisation, l'enregistrement dans la mémoire cache C2 est conditionné par les résultats des deux comparaisons décrites ci-dessus et réalisées aux étapes K103, K104. En particulier, la donnée enregistrée D_C2 dans la mémoire cache C2 est remplacée lors de l'enregistrement par la donnée valide D_A3 reçue.
- [0161] En particulier, suite à la réception d'une donnée valide D_A3 en provenance de l'application APP, le noyau KER détermine un haché H_A2 sur la base de l'identifiant ID_A et d'une fonction de hachage H2. Le noyau KER détermine ensuite, à l'étape K103, si une donnée enregistrée D_C2 dans la deuxième mémoire cache C2 partage un haché H_C2 identique au haché H_A2, i.e. $H_C2 = H_A2$. Dans l'affirmative, le noyau KER compare, à l'étape K104, un compteur CNT_C2 associé à la donnée enregistrée D_C2 à un seuil THR_2. Selon un mode de réalisation, le seuil THR_2 est égal à 1. Si le compteur CNT_C2 est inférieur ou égale au seuil THR_2, alors le noyau KER enregistre, à l'étape K105, la donnée valide D_A3 reçue dans la deuxième mémoire cache C2, et en particulier dans la case mémoire stockant la donnée enregistrée D_C2.
- [0162] Selon un mode de réalisation particulier de l'invention, la valeur d'un compteur CNT_C2 est déterminée de la manière suivante. La valeur d'un compteur CNT_C2 varie en fonction du nombre d'accès en lecture et en écriture à la donnée D_C2 dans la deuxième mémoire C2. À l'initialisation des entrées de la mémoire cache C2, les compteurs associés aux entrées sont initialisés à une valeur nulle. Si une donnée enregistrée D_C2' dans la mémoire cache C2 est invalide, alors le compteur CNT_C2 associé est nul ; et si la donnée D_C2 est valide, le compteur associé CNT_C2 est supérieur ou égal à 1. Si le compteur associé CNT_C2 est égal à 1, la donnée enregistrée C_2 est valide dans la mémoire cache C2 mais peut être évincée par une donnée reçue D_A3 en réponse à une requête GET_A. Lorsqu'une donnée D_C2 est enregistrée ou mise à jour dans la mémoire cache C2, le compteur associé CNT_C2 est mis à une valeur maximale MAX_CNT2. De même, lorsqu'une requête de lecture GET_C avec l'identifiant ID_C est interceptée et que la deuxième mémoire cache C2 comprend une donnée valide D_C2 correspondant à l'identifiant ID_C, alors le compteur associé CNT_C2 est mis à la valeur maximale MAX_CNT2.

- [0163] En outre, pour une donnée enregistrée D_{C2} , la mémoire cache C_2 peut comprendre une valeur de concurrent CHA_{C2} permettant d'identifier une donnée concurrente D_D . Ladite valeur de concurrent CHA_{C2} est un haché déterminé sur la base de l'identifiant ID_D de la donnée concurrente D_D et d'une fonction de hachage $G2$, i.e. $CHA_{C2}=G2(ID_D)$. Pour chaque donnée valide D_{A3} reçue en provenance de l'application APP, le noyau KER détermine un haché de concurrent $G2(ID_A)$ sur la base de l'identifiant ID_A et de la fonction de hachage $G2$. Si le compteur CNT_{C2} est supérieur à 1, et que la valeur de concurrent CHA_{C2} est différente du haché de concurrent $G2(ID_A)$, i.e. $CHA_{C2} \neq G2(ID_A)$, alors la valeur de concurrent CHA_{C2} est remplacée par le haché $G2(ID_A)$ associé à la donnée valide D_{A3} reçue, i.e. $G2(ID_A) \rightarrow CHA_{C2}$. Si le compteur CNT_{C2} est supérieur à 1, et que la valeur de concurrent CHA_{C2} est égale au haché de concurrent $G2(ID_A)$, i.e. $CHA_{C2}=G2(ID_A)$, alors le compteur CNT_{C2} est décrémenté, i.e. $CNT_{C2}-1 \rightarrow CNT_{C2}$. Lorsqu'une donnée D_{C2} est enregistrée ou mise à jour dans la mémoire cache $C2$, la valeur de concurrent CHA_{C2} est mis à NULL.
- [0164] De manière analogue, le noyau KER peut mettre en œuvre un enregistrement tel que décrit ci-dessus dans la première mémoire cache $C1$ d'une donnée valide D_{A3} reçue en provenance de l'application APP en réponse à une requête GET_A . Dans ce cas, le noyau KER met en œuvre des étapes K113, K114, et K115 analogues aux étapes K103, K104, et K105 décrites ci-dessus pour enregistrer une donnée valide D_{A3} dans la première mémoire cache $C1$, et en utilisant un seuil THR_1 , une fonction de hachage $G1$ et une valeur maximale MAX_{CNT1} .
- [0165] Ainsi, les différents niveaux de cache peuvent partager des structures de données et des procédés d'enregistrement de données similaires, bien que paramétrables indépendamment. Un des objectifs de ces étapes d'enregistrement est de maintenir dans les mémoires caches $C1$, $C2$ les données fréquemment demandées en évitant que celles-ci ne soient remplacées par des données moins accédées. En outre, ces mécanismes d'enregistrements ont également pour objectif d'être exécutable en un temps minime afin de ne pas impacter négativement la latence de traitement des requêtes.
- [0166] La [Fig.7A] représente schématiquement un exemple de structures de données comprises dans une entité de traitement de requêtes d'accès à des données selon un mode de réalisation particulier.
- [0167] Tel qu'illustré par la [Fig.7A], pour une donnée D_{A1} , la mémoire cache $C1$ comprend dans une de ses entrées : l'identifiant ID_A ; la donnée D_{A1} ; un indicateur de validité V_{A1} ; et un compteur CNT_{A1} .
- [0168] Tel qu'illustré par la [Fig.7A], la validité d'une donnée D_{A1} , D_{A2} dans une des mémoires caches $C1$, $C2$ est caractérisée en utilisant un indicateur V_{A1} , V_{A2} . Typiquement, si la donnée D_{A1} , D_{A2} est valide dans la mémoire cache $C1$, $C2$,

l'indicateur V_A1, V_A2 représentatif de la validité est égal à 1 ; et si la donnée D_A1', D_A2' est invalide dans la mémoire cache C1, C2, l'indicateur V_A1, V_A2 est nul. Dans le présent document, les signes de référence des données valides D_A1, D_A2 sont différenciées des signes de référence des données invalides D_A1', D_A2', ces derniers se terminant par une apostrophe.

- [0169] Selon un mode de réalisation particulier, la validité d'une donnée D_A1, D_A2 dans une des mémoires caches C1, C2 est caractérisée en utilisant le compteur CNT_A1, CNT_A2 associé à la donnée D_A1, D_A2. Typiquement, si la donnée D_A1, D_A2 est valide dans la mémoire cache C1, C2, le compteur CNT_A1, CNT_A2 représentatif de la validité est supérieur ou égal à 1 ; et si la donnée D_A1', D_A2' est invalide dans la mémoire cache C1, C2, le compteur CNT_A1, CNT_A2 est nul.
- [0170] La [Fig.7B] représente schématiquement un exemple de structures de données comprises dans une entité de traitement de requêtes d'accès à des données selon un mode de réalisation particulier.
- [0171] Dans le mode de réalisation décrit ici, la première mémoire cache C1 comprend une table de hachage TAB1. La table de hachage TAB1 permet une association directe clé-valeur, soit ID_A-D_A et permet ainsi retrouver une donnée rapidement, en la cherchant à un emplacement de la table TAB1 correspondant au résultat d'une fonction de hachage H1. Ainsi, le procédé de traitement de requêtes comprend selon ce mode de réalisation particulier une étape de détermination d'un haché H_A1, le haché H_A1 étant calculé à partir de l'identifiant ID_A et de la fonction de hachage H1. Le haché H_A1 permet de retrouver rapidement la donnée D_A1 dans la table TAB1. En particulier, le haché H_A1 peut être réduit à la taille de la table de hachage TAB1 via une opération de modulo pour obtenir l'index de l'entrée comprenant la donnée D_A1 dans la table TAB1.
- [0172] De manière avantageuse, la fonction de hachage H1 utilisée est une fonction de hachage Fowler-Noll-Vo-1A, plus couramment désigné par l'acronyme FNV-1A. Cette fonction permet notamment de calculer le haché H_A1 tout en lisant une requête, permettant ainsi de réduire la latence due aux traitements des requêtes et plus généralement de gagner en performance par rapport à un calcul de haché réalisé a posteriori.
- [0173] Par ailleurs, l'enregistrement d'une donnée D_A dans une table de hachage TAB1 tel que décrite ci-dessus comprend également l'enregistrement de l'identifiant ID_A de cette donnée D_A dans une entrée de la table de hachage TAB1. Les entrées d'une table de hachage TAB1 sont couramment dénommées alvéoles (ou avec les expressions anglo-saxonnes « buckets » ou « slots »).
- [0174] Une telle structure de données offre une complexité en temps constant pour les opérations de recherche, d'insertion ou d'invalidation d'une donnée, ce qui permet de

réduire la latence due aux traitements des requêtes et permet de traiter une charge importante avec un débit de requêtes élevé.

- [0175] La [Fig.7C] représente schématiquement un exemple de structures de données comprises dans une entité de traitement de requêtes d'accès à des données selon un mode de réalisation particulier.
- [0176] Tel qu'illustré par la [Fig.7C], la deuxième mémoire cache C2 peut comprendre une table de hachage TAB2. La table de hachage TAB2 permet une association directe clé-valeur, soit ID_A-D_A. Ainsi, le procédé de traitement de selon ce mode de réalisation particulier comprend une étape de détermination d'un haché H_A2, le haché H_A2 étant calculé à partir de l'identifiant ID_A et d'une fonction de hachage H2. Le haché H_A2 permet de retrouver rapidement la donnée D_A1 dans la table TAB1. De manière avantageuse, la fonction de hachage H2 utilisée est une fonction de hachage Fowler-Noll-Vo-1A, plus couramment désigné par l'acronyme FNV-1A. Par ailleurs, l'enregistrement d'une donnée D_A dans une entrée de la table de hachage TAB2 comprend également l'enregistrement de son identifiant ID_A dans l'entrée.
- [0177] Tel qu'illustré par la [Fig.7C], pour une donnée D_A, la table TAB2 de la deuxième mémoire cache C2 comprend dans une de ses entrées : l'identifiant ID_A ; la donnée D_A2 ; un indicateur de validité V_A2 ; un compteur CNT_A2 ; et une valeur de concurrent CHA_A2.
- [0178] Bien évidemment, la première mémoire cache C1 peut comprendre une table de hachage TAB1 mettant en œuvre une structure de données telle que décrite ci-dessus.
- [0179] Selon un mode de réalisation particulier, les fonctions de hachage H1, H2, G1, et G2 et les valeurs maximales MAX_CNT1 et MAX_CNT2 sont paramétrées de manière différente pour la première mémoire cache C1 et la deuxième mémoire cache C2.
- [0180] En effet, les différents mémoires caches C1, C2 de l'architecture ont toutes deux pour but de stocker les données de l'application APP. Toutefois, chaque niveau de cache est paramétrable indépendamment l'un de l'autre afin de s'adapter aux contraintes matérielles des différents environnements d'exécution. Ce paramétrage permet notamment d'ajuster chaque niveau de cache aux ressources mémoires disponibles. La taille de la première mémoire cache C1 de la carte réseau NIC peut être différente de la taille de la deuxième mémoire cache C2 du noyau KER. Typiquement, la taille de la première mémoire cache C1 de la carte réseau NIC est inférieure à la taille de la deuxième mémoire cache C2 du noyau KER.
- [0181] La [Fig.8A] représente, sous forme d'ordinogramme, des étapes d'un procédé de traitement de requêtes d'accès à des données selon un mode de réalisation particulier.
- [0182] À l'étape N210, la carte réseau NIC reçoit une requête d'accès en écriture SET_B comprenant un identifiant ID_B d'une donnée à écrire D_B. Typiquement, la requête SET_B est émise par un dispositif client CLT à destination de l'entité SERV.

- [0183] À l'étape N240, la carte réseau NIC transmet la requête d'écriture SET_B au noyau KER.
- [0184] À l'étape K270, suite à la réception de la requête d'écriture SET_B en provenance de la carte réseau NIC, le noyau KER transmet la requête d'écriture SET_B à l'application APP.
- [0185] La [Fig.8B] représente, sous forme d'ordinogramme, des étapes d'un procédé de traitement de requêtes d'accès à des données selon un mode de réalisation particulier. En particulier, le procédé illustré par la [Fig.8B] comprend en comparaison du procédé illustré par la [Fig.8A], les étapes supplémentaires suivantes.
- [0186] Suite, à l'étape K270, la transmission par le noyau KER de la requête d'écriture SET_B à l'application APP, le noyau KER met en œuvre une étape K280.
- [0187] À l'étape K280, le noyau KER reçoit une réponse REP_SET_B à la requête d'écriture SET_B comprenant un accusé de réception ACK émis par l'application APP en réponse à la requête d'écriture SET_B. Selon un mode de réalisation, le noyau KER transmet ensuite la réponse REP_SET_B à la carte réseau NIC.
- [0188] À l'étape N290, la carte réseau NIC reçoit la réponse REP_SET_B et la transmet à l'étape N300 à destination du dispositif client CLT émetteur de la requête SET_B.
- [0189] La [Fig.8C] représente, sous forme d'ordinogramme, des étapes d'un procédé de traitement de requêtes d'accès à des données selon un mode de réalisation particulier.
- [0190] À l'étape N210, la carte réseau NIC reçoit une requête d'accès en écriture SET_B comprenant un identifiant ID_B d'une donnée D_B à écrire. L'identifiant ID_B peut être identique ou différent de l'identifiant ID_A reçu à l'étape N10.
- [0191] À l'étape N220, la carte réseau NIC détermine si la première mémoire cache C1 comprend une donnée D_B1 correspondant à l'identifiant ID_B.
- [0192] À l'étape N230, si la première mémoire cache C1 comprend la donnée D_B1, la carte réseau NIC invalide la donnée D_B1' dans la première mémoire cache C1, c'est-à-dire marque la donnée comme étant invalide.
- [0193] À l'étape N240, la carte réseau NIC transmet la requête d'écriture SET_B au noyau KER.
- [0194] À l'étape K250, suite à la réception de la requête d'écriture SET_B en provenance de la carte réseau, le noyau KER détermine si la deuxième mémoire cache C2 comprend une donnée D_B2 correspondant à l'identifiant ID_B compris dans la requête d'écriture SET_B.
- [0195] À l'étape K260, si la deuxième mémoire cache C2 comprend une donnée D_B2 correspondant à l'identifiant ID_B compris dans la requête d'écriture SET_B, le noyau KER invalide la donnée D_B2' dans la deuxième mémoire cache C2.
- [0196] À l'étape K270, le noyau KER transmet la requête d'écriture SET_B à l'application APP.

- [0197] Après qu'une donnée D_B1', D_B2' a été invalidée dans une mémoire cache C1, C2, une requête de lecture GET_B subséquente demandant la donnée D_B ne sera ni traitée par la carte réseau NIC ni par le noyau KER. Plutôt que d'utiliser la requête d'écriture SET_B pour mettre à jour l'entrée correspondante dans les différentes mémoires caches C1, C2, le cache multi-niveaux utilise une donnée valide D_B3 émise par l'application APP en réponse à une requête GET_B et interceptée par le noyau KER ou la carte réseau NIC, ce qui permet de garantir la cohérence des données.
- [0198] La [Fig.9] représente schématiquement des étapes d'un procédé de traitement de requêtes d'accès à des données selon un mode de réalisation particulier. En particulier, la [Fig.9] illustre la hiérarchie des mémoires caches C1, C2 et de la mémoire d'application C3, ainsi que le traitement multi-niveaux des requêtes d'accès GET_A, SET_B.
- [0199] Tel qu'illustré sur la [Fig.9], l'entité SERV reçoit en provenance d'un dispositif client CLT une requête GET_A demandant l'accès en lecture de la donnée D_A, la requête de lecture GET_A comprenant l'identifiant ID_A. La carte réseau NIC de l'entité SERV intercepte la requête GET_A et détermine si une donnée valide D_A1 correspondant à l'identifiant ID_A est enregistrée dans la première mémoire cache C1. Dans l'affirmative, la carte réseau NIC est en mesure de répondre à la requête GET_A et ne transmet pas la requête GET_A au noyau KER. La carte réseau NIC lit la première mémoire cache C1 et transmet au dispositif client CLT une réponse REP_GET_A comprenant la donnée valide D_A1. Dans ce cas, l'entité SERV a traité et répondu à la requête GET_A sans solliciter le noyau KER ni l'application APP. Par conséquent, la latence d'accès aux données est réduite de manière significative et, en outre, l'utilisation des ressources matérielles et logicielles aussi.
- [0200] Tel qu'illustré par la [Fig.9], dans le cas d'une requête de lecture GET_B comprenant un identifiant ID_B, si la première mémoire cache C1 ne comprend pas une donnée valide D_B1 correspondant à l'identifiant ID_B, la carte réseau NIC transmet la requête GET_B au niveau supérieur, le noyau KER. Suite à la réception de la requête GET_B, le noyau KER détermine si une donnée valide D_B2 correspondant à l'identifiant ID_B est enregistrée dans la deuxième mémoire cache C2. Dans l'affirmative, le noyau KER lit la donnée D_B2 et transmet à la carte réseau NIC une réponse REP_GET_B à la requête de lecture GET_B, la réponse REP_GET_B comprenant la donnée valide D_B2. La carte réseau NIC transmet au dispositif client CLT la réponse REP_GET_B. Dans ce cas, l'entité SERV a traité et répondu à la requête GET_B sans solliciter l'application APP, ce qui permet de réduire la latence d'accès aux données et d'économiser des ressources de traitement.
- [0201] Tel qu'illustré par la [Fig.9], dans le cas d'une requête de lecture GET_C comprenant un identifiant ID_C, si la première et la deuxième mémoires caches C1, C2 ne com-

prennent pas de données valides D_C1, D_C2 correspondant à l'identifiant ID_C, la carte réseau NIC transmet la requête GET_C au noyau KER ; et le noyau KER transmet la requête GET_C à l'application APP. Suite à la réception de la requête GET_C, l'application APP traite et répond à la requête GET_C. En réponse à la requête GET_C, l'application APP transmet une réponse REP_GET_C à la requête de lecture GET_C, la réponse REP_GET_C comprenant une donnée D_C3, au noyau KER, qui la transmet à son tour à la carte réseau NIC. Suite à la réception de la réponse REP_GET_C, la carte réseau NIC transmet au dispositif client CLT.

- [0202] Tel qu'illustré par la [Fig.9], dans le cas d'une requête en écriture SET_D, la carte réseau NIC transmet la requête SET_D au noyau KER ; et le noyau KER transmet la requête SET_D à l'application APP. L'application APP traite la requête d'écriture SET_D et enregistre la donnée D_D dans la mémoire d'application C3. La transmission par la carte réseau NIC et le noyau KER des requêtes en écriture SET_D à l'application APP permet notamment d'assurer la cohérence des données D_D entre les différentes mémoires C1, C2, C3. De plus, la carte réseau NIC et le noyau KER peuvent utiliser la requête d'écriture SET_D pour invalider dans les mémoires caches C1, C2 des données D_D1', D_D2' correspondant à l'identifiant ID_D.
- [0203] Un procédé et une entité SERV de traitement de requêtes d'accès à des données tels que proposés présentent les avantages suivants. Le traitement et la réponse à des requêtes de lecture GET par la carte réseau NIC et le noyau KER permettent de réduire de manière significative le temps de traitement des requêtes GET par l'entité, ces requêtes n'étant pas remontées à l'application APP. L'entité SERV est ainsi en mesure de réduire la latence d'accès aux données dans un réseau et d'augmenter le débit des requêtes traitées. En outre, le traitement de requêtes GET, SET par la carte réseau NIC ou le noyau KER est transparent pour l'application APP. La cohérence des données est garantie entre les différentes mémoires C1, C2, C3, seules des données valides étant transmises en réponse aux requêtes GET.
- [0204] Il est à noter que l'ordre dans lequel s'enchaînent les étapes d'un procédé de traitement de requêtes d'accès tel que décrit précédemment en référence, notamment aux dessins ci-joints, ne constitue qu'un exemple de réalisation dépourvu de tout caractère limitatif, des variantes étant possibles. Par ailleurs, les signes de référence ne sont pas limitatifs de l'étendue de la protection, leur unique fonction étant de faciliter la compréhension des revendications.
- [0205] Un homme du métier comprendra que les modes de réalisation et variantes décrits ci-dessus ne constituent que des exemples non limitatifs de mise en œuvre de l'invention. En particulier, l'homme du métier pourra envisager une quelconque adaptation ou combinaison des modes de réalisation et variantes décrits ci-dessus afin de répondre à un besoin bien particulier.

Revendications

[Revendication 1] Procédé de traitement de requêtes d'accès (GET_A) à des données, ledit procédé étant mis en œuvre par une entité (SERV) connectée à un réseau (NET), ladite entité (SERV) comprenant une carte réseau (NIC) et un noyau de système d'exploitation (KER), ledit procédé comprenant des étapes de :

- interception (N20) par la carte réseau (NIC) d'un paquet réseau à destination d'une application (APP) configurée pour traiter et répondre auxdites requêtes d'accès (GET_A), ledit paquet réseau portant une requête de lecture (GET_A) comprenant un identifiant (ID_A) d'une donnée (D_A) ;
 - transmission (N50) par la carte réseau (NIC) de la requête de lecture (GET_A) au noyau (KER) en fonction de la présence d'une donnée (D_A1) correspondant audit identifiant (ID_A) dans une première mémoire cache (C1) de la carte réseau (NIC) ; et
- émission (N130) par la carte réseau (NIC) d'une réponse (REP_GET_A) à la requête de lecture (GET_A), ladite réponse étant déterminée en fonction de la présence d'une donnée (D_A1, D_A2) correspondant audit identifiant (ID_A) dans la première mémoire cache (C1) ou dans une deuxième mémoire cache (C2) du noyau (KER).

[Revendication 2] Procédé selon la revendication 1 caractérisé en ce que :

- la transmission (N50) de la requête (GET_A) par la carte réseau (NIC) au noyau (KER) est réalisée si la première mémoire cache (C1) ne comprend pas une donnée (D_A1) correspondant audit identifiant (ID_A) ; et
- ladite réponse (REP_GET_A) comprend :
 - si la première mémoire cache (C1) comprend une donnée (D_A1) correspondant audit identifiant (ID_A), ladite donnée (D_A1) lue dans la première mémoire cache (C1) ; ou
 - si la deuxième mémoire cache (C2) comprend une donnée (D_A2) correspondant audit identifiant

(ID_A), ladite donnée (D_A2) lue dans la deuxième mémoire cache (C2).

- [Revendication 3] Procédé selon l'une des revendications 1 ou 2 comprenant, si aucune desdites mémoires caches (C1, C2) ne comprend une donnée (D_A1, D_A2) correspondant audit identifiant (ID_A), une étape de :
- transmission (K80) de la requête de lecture (GET_A) à l'application (APP), ladite réponse (REP_GET_A) émise par la carte réseau (NIC) comprenant une donnée (D_A3) correspondant audit identifiant (ID_A), cette donnée (D_A3) étant émise par l'application (APP) en réponse à la requête de lecture (GET_A).
- [Revendication 4] Procédé selon l'une des revendications 1 à 3 caractérisé en ce que :
- la requête de lecture (GET_A) est transmise (N50) par la carte réseau (NIC) au noyau (KER) si la première mémoire cache (C1) ne comprend pas une donnée valide (D_A1) correspondant audit identifiant (ID_A) ;
 - la requête de lecture (GET_A) est transmise (K80) par le noyau (KER) à l'application (APP) si aucune desdites mémoires caches (C1, C2) ne comprend une donnée valide (D_A2) correspondant audit identifiant (ID_A) ; et
 - ladite réponse (REP_GET_A) émise par la carte réseau (NIC) comprend une donnée valide (D_A1, D_A2, D_A3) correspondant audit identifiant (ID_A).
- [Revendication 5] Procédé selon la revendication 3 ou 4 caractérisé en ce qu'il comprend une étape de :
- enregistrement (K100, K110) dans au moins une dite mémoire cache (C1, C2) de ladite donnée (D_A3) reçue de l'application (APP) en réponse à la requête de lecture (GET_A), en fonction dudit identifiant (ID_A).
- [Revendication 6] Procédé selon la revendication 5 caractérisé en que ladite étape d'enregistrement (K105, N125) est réalisée dans une dite mémoire

cache (C1, C2) en fonction d'au moins une des opérations suivantes :

- une comparaison (K103, N123) entre ledit identifiant (ID_A) compris dans la requête de lecture (GET_A) et un identifiant (ID_C) d'une donnée enregistrée (D_C1, D_C2) dans ladite mémoire cache (C1, C2) ;
- une comparaison (K104, N124) entre un seuil (THR_1, THR_2) et un compteur (CNT_C1, CNT_C2) associé à une donnée enregistrée (D_C1, D_C2) dans ladite mémoire cache (C1, C2).

[Revendication 7]

Procédé selon l'une quelconque des revendications 1 à 6 caractérisé en ce qu'au moins une dite mémoire cache (C1, C2) comprend une table de hachage (TAB1, TAB2), et en ce que ledit procédé comprend une étape de :

- détermination d'un haché (H_A1, H_A2) sur la base dudit identifiant (ID_A) compris dans la requête de lecture (GET_A) pour indexer une donnée (D_A1, D_A2) correspondant à l'identifiant (ID_A) dans ladite table (TAB1, TAB2).

[Revendication 8]

Procédé selon l'une quelconque des revendications 1 à 7 caractérisé en ce qu'il comprend une étape de :

- si une dite mémoire cache (C1, C2) comprend (N220, K250) une donnée (D_B1, D_B2) correspondant à un identifiant (ID_B) compris dans une requête d'écriture (SET_B), invalidation (N230, K260) dans ladite mémoire cache (C1, C2) de la donnée (D_B1', D_B2') correspondant à l'identifiant (ID_B) compris dans la requête d'écriture (SET_B).

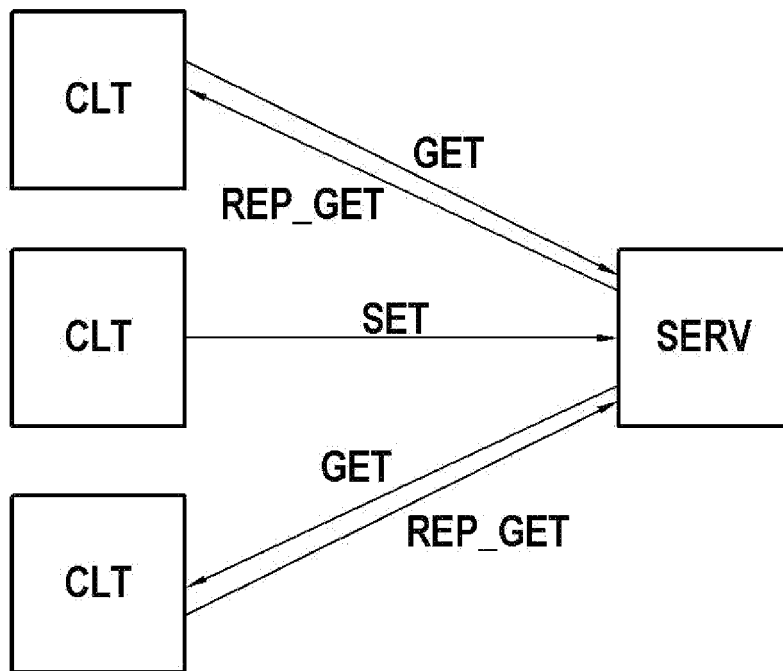
[Revendication 9]

Entité de traitement de requêtes d'accès (GET_A) à des données, ladite entité (SERV) étant connectée à un réseau (NET), ladite entité (SERV) comprenant une carte réseau (NIC) et un noyau de système d'exploitation (KER), la carte réseau (NIC) comprenant une première mémoire cache (C1), le noyau (KER) comprenant une deuxième mémoire cache (C2), ladite entité comprenant :

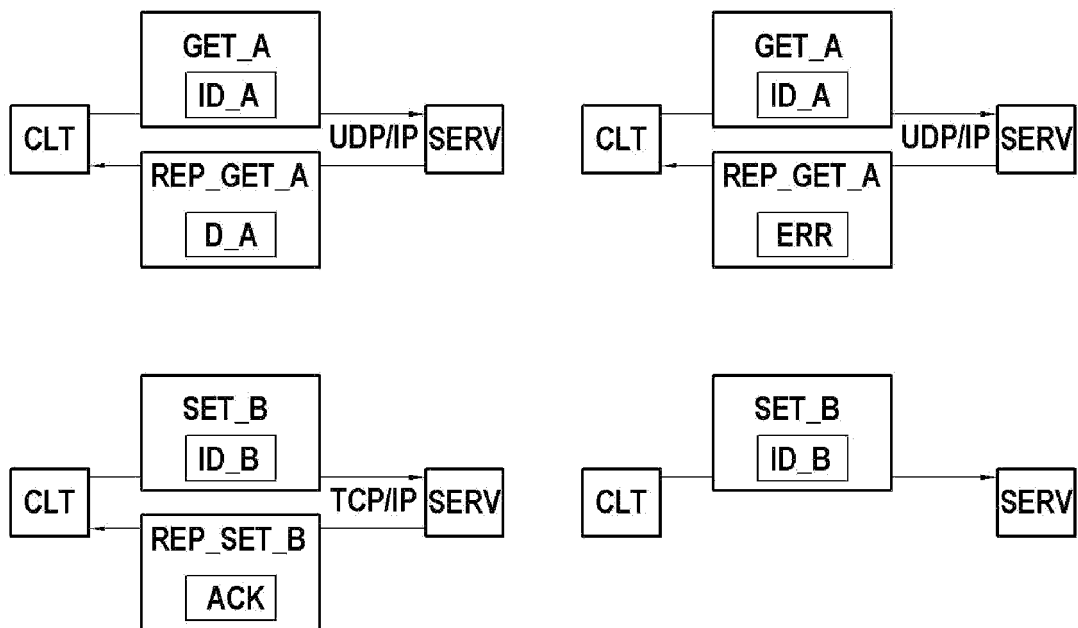
- un module de réception réseau (MOD_RX1) compris dans la carte réseau (NIC) pour intercepter un paquet réseau à destination d'une application (APP) configurée pour traiter et répondre auxdites requêtes d'accès (GET_A), ledit paquet réseau portant une requête de lecture (GET_A) comprenant un identifiant (ID_A) d'une donnée (D_A) ;
- un premier module de détermination (MOD_DET1) compris dans la carte réseau (NIC) pour déterminer la présence dans la première mémoire cache (C1) d'une donnée (D_A1) correspondant audit identifiant (ID_A) ;
- un premier module de transmission (MOD_FWD1) compris dans la carte réseau (NIC) pour transmettre la requête de lecture (GET_A) au noyau (KER) ;
- un deuxième module de détermination (MOD_DET2) compris dans le noyau (KER) pour déterminer la présence dans la deuxième mémoire cache (C2) d'une donnée (D_A2) correspondant audit identifiant (ID_A) ; et
- un module d'émission réseau (MOD_TX1) compris dans la carte réseau (NIC) pour émettre une réponse (REP_GET_A) à la requête de lecture (GET_A).

- [Revendication 10] Entité selon la revendication 9 caractérisé en ce que ladite entité (SERV) est un serveur DNS, lesdites données étant par exemple des adresses IP identifiées par des URLs, ou en ce que ladite entité (SERV) est un serveur de clés, comprenant par exemple des paires clé-valeur.
- [Revendication 11] Programme d'ordinateur comprenant des instructions pour la mise en œuvre des étapes du procédé selon l'une quelconque des revendications 1 à 8 lorsque les instructions sont exécutées par un ou plusieurs processeurs (PROC1, PROC2).
- [Revendication 12] Support d'informations lisible par ordinateur comprenant un programme d'ordinateur selon la revendication 11.

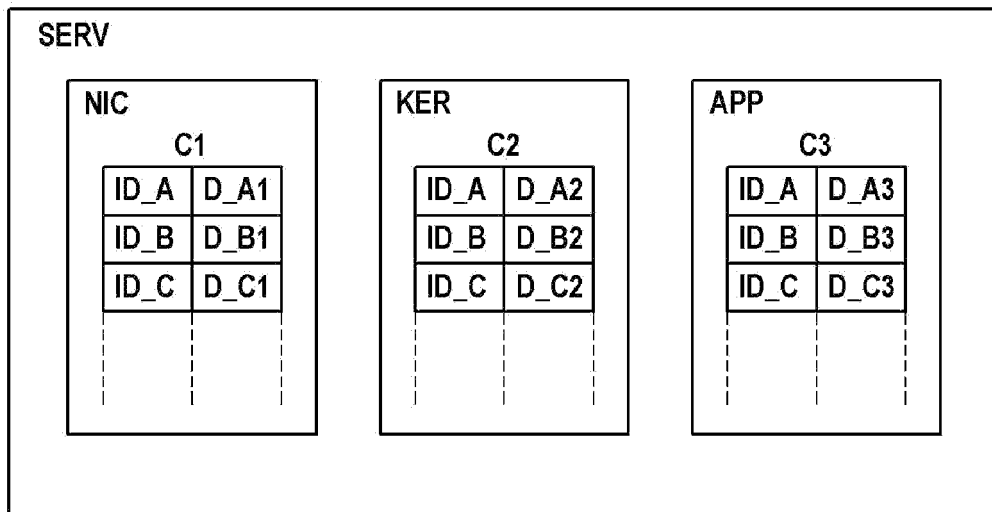
[Fig. 1A]



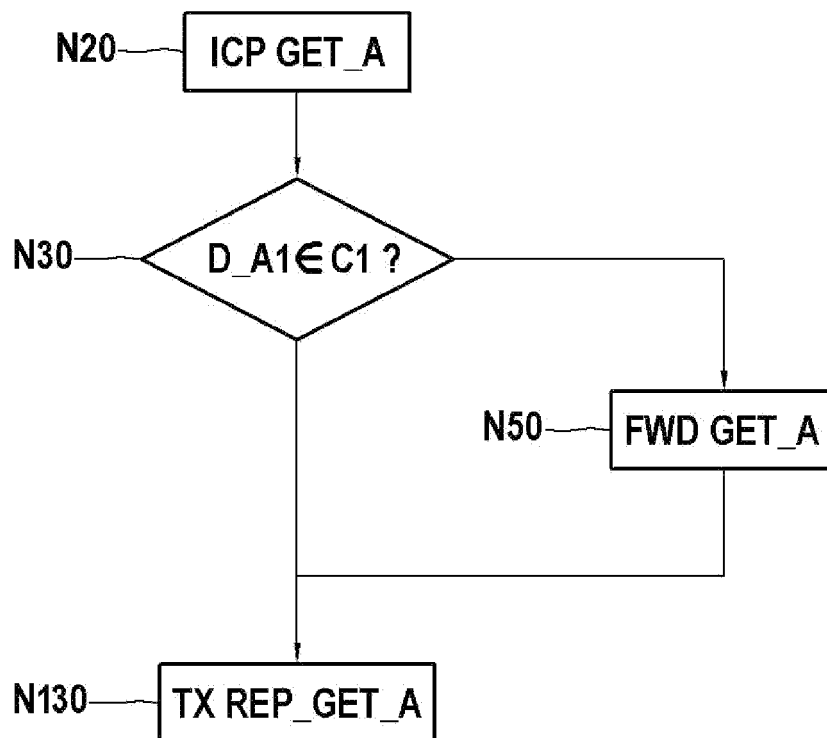
[Fig. 1B]



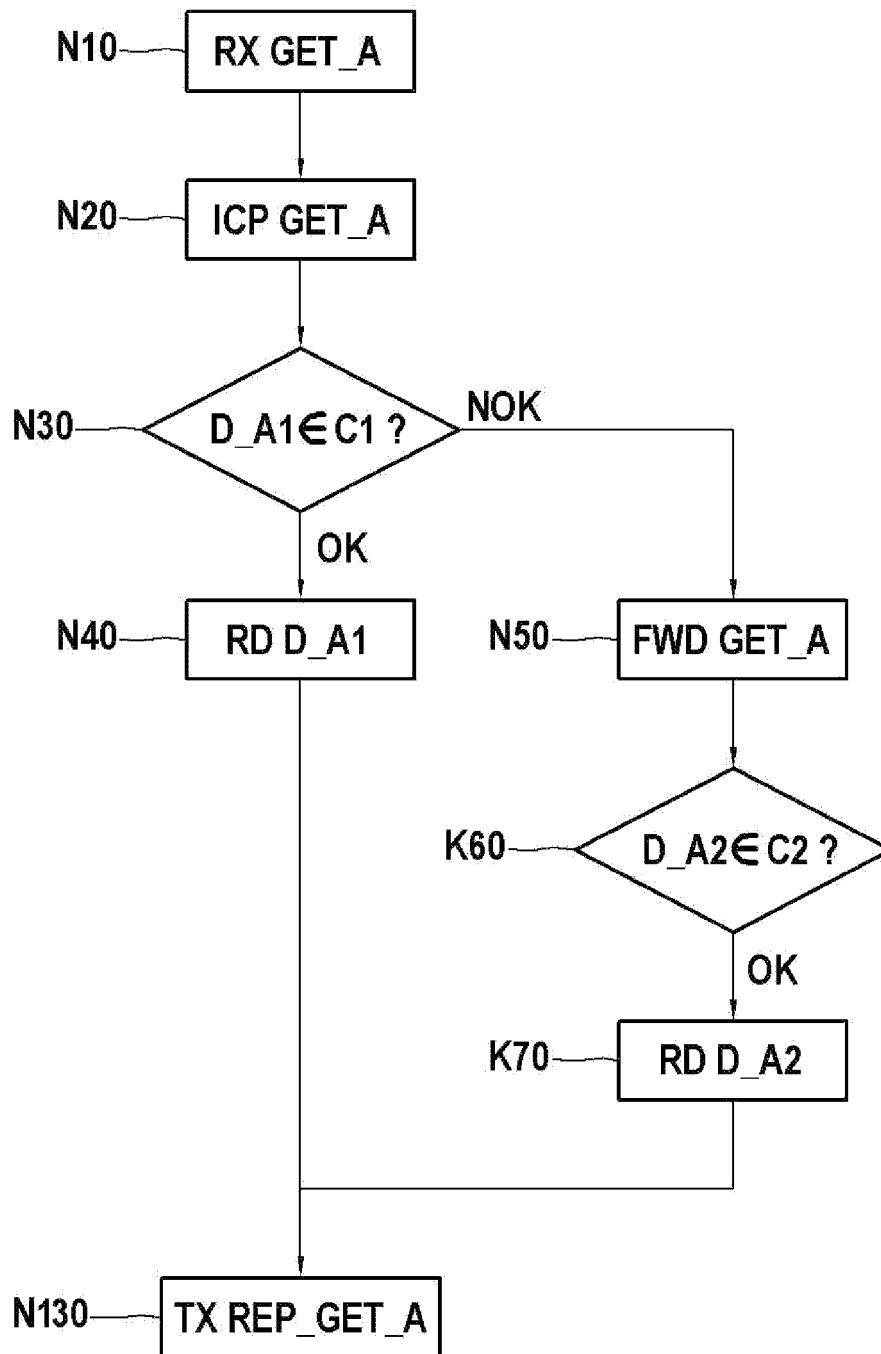
[Fig. 1C]



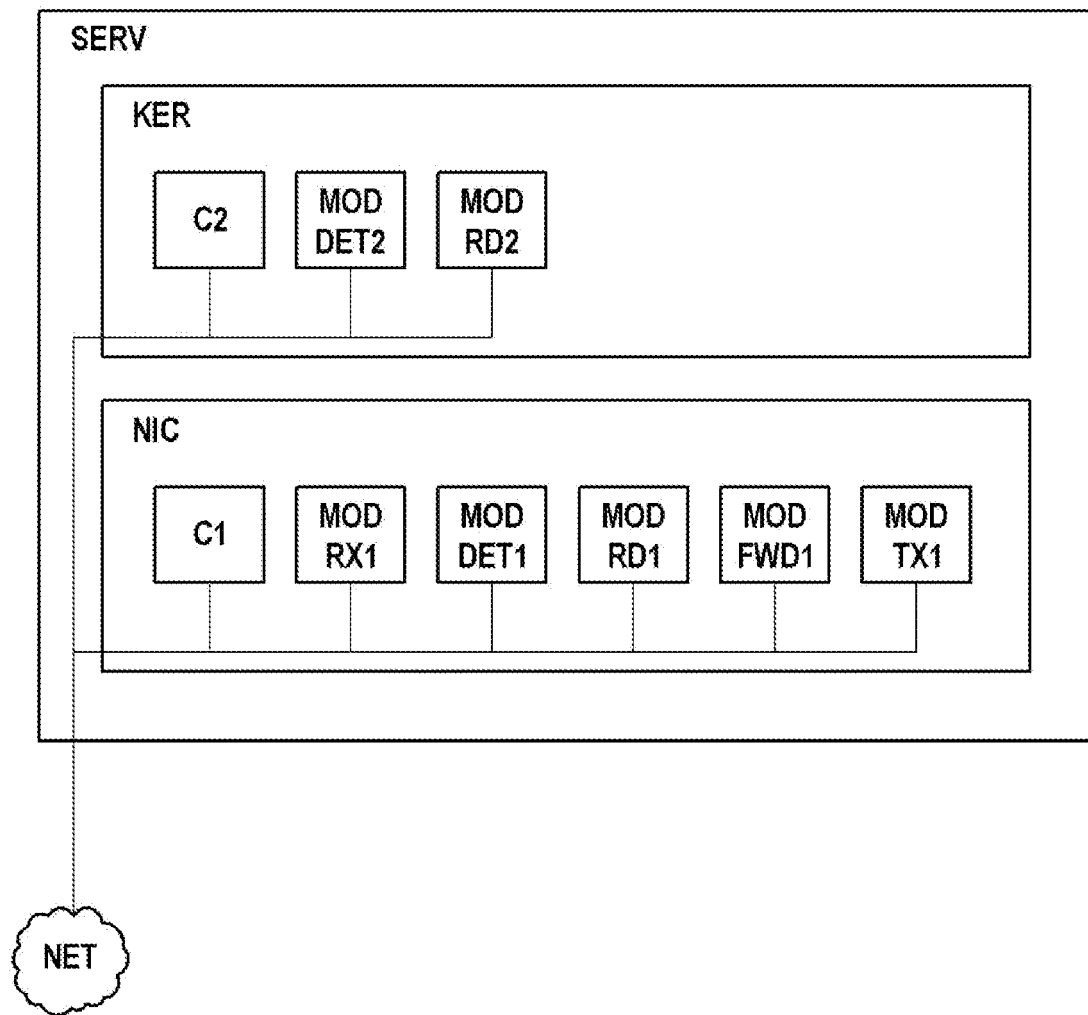
[Fig. 2A]



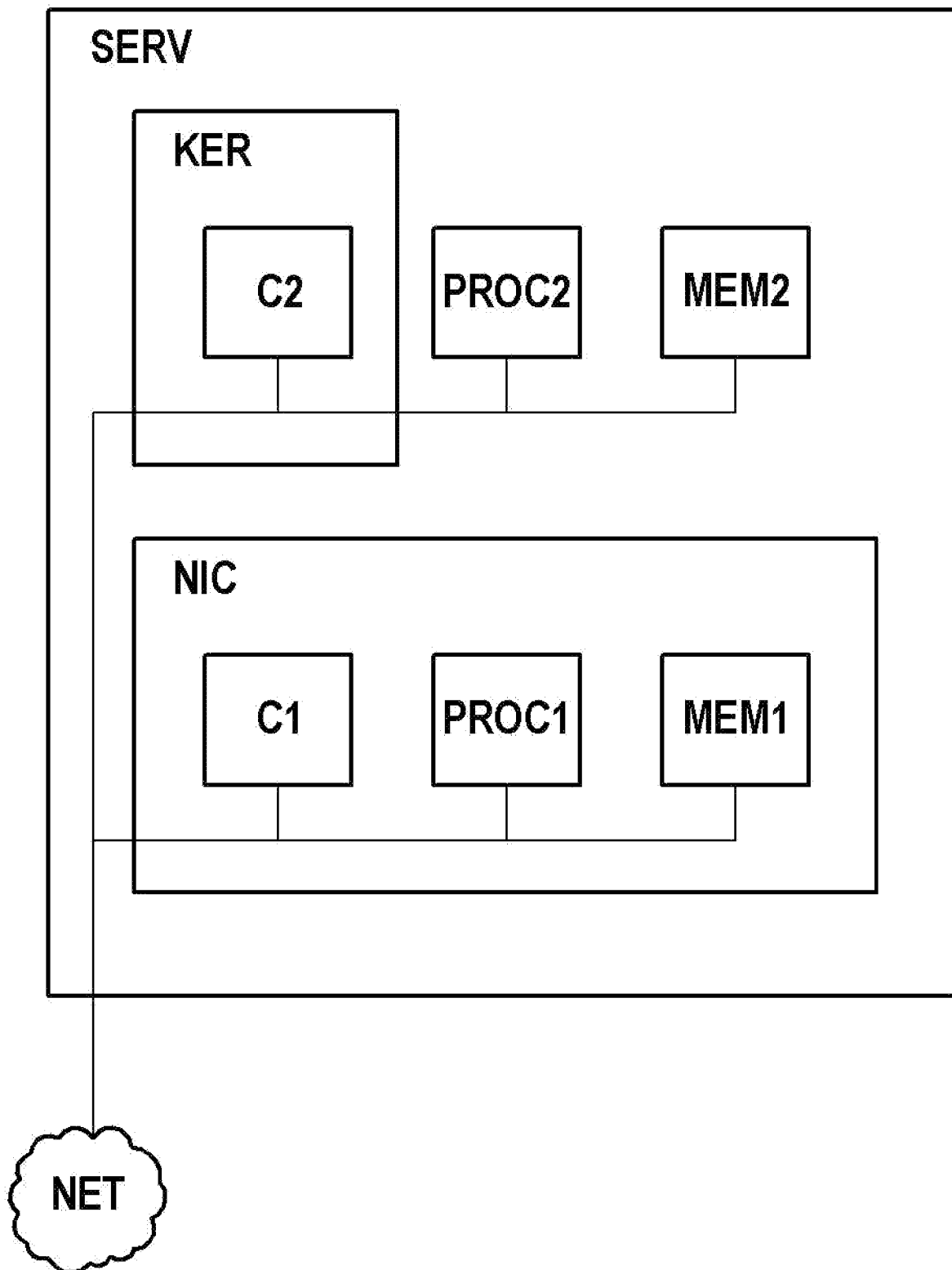
[Fig. 2B]



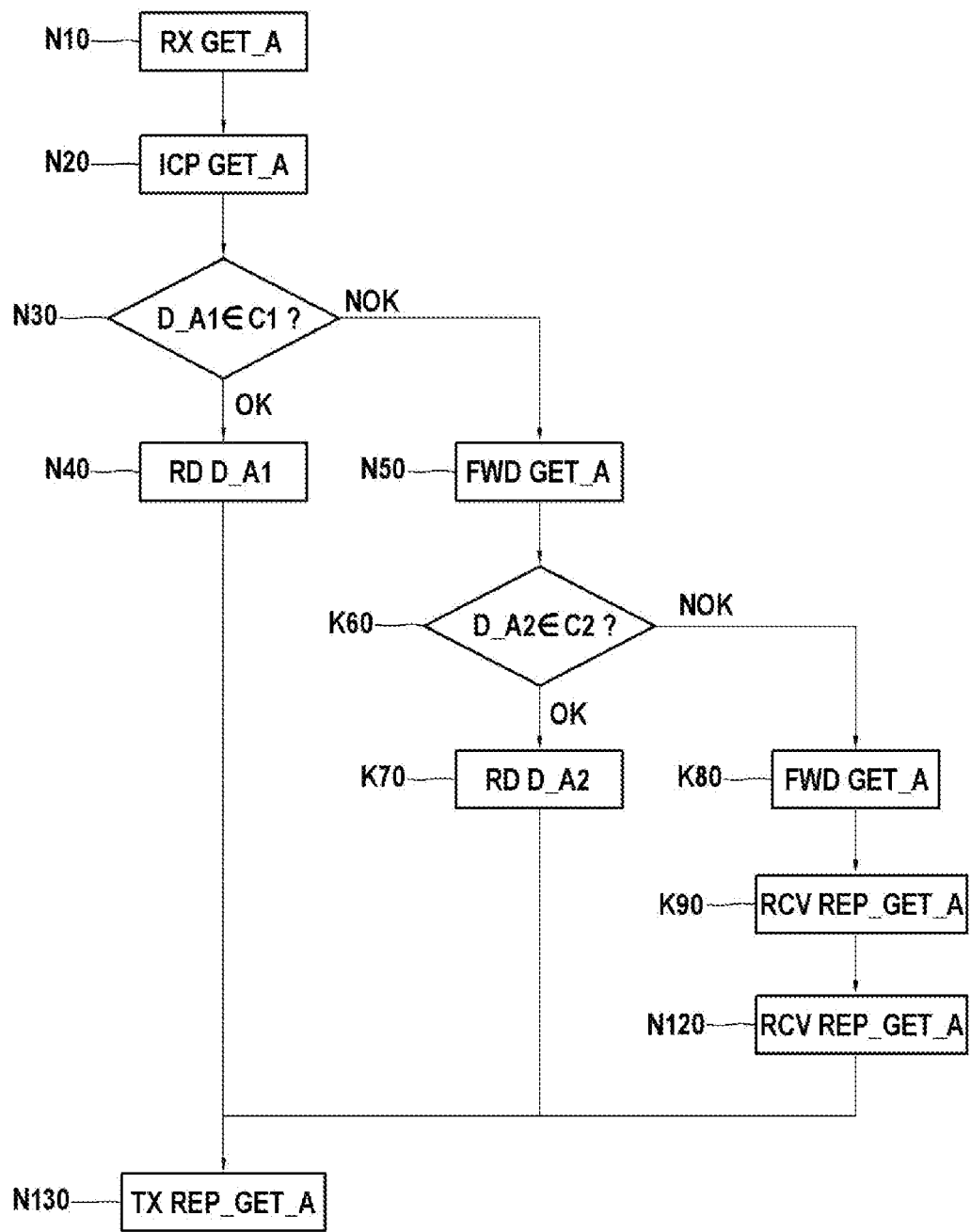
[Fig. 3A]



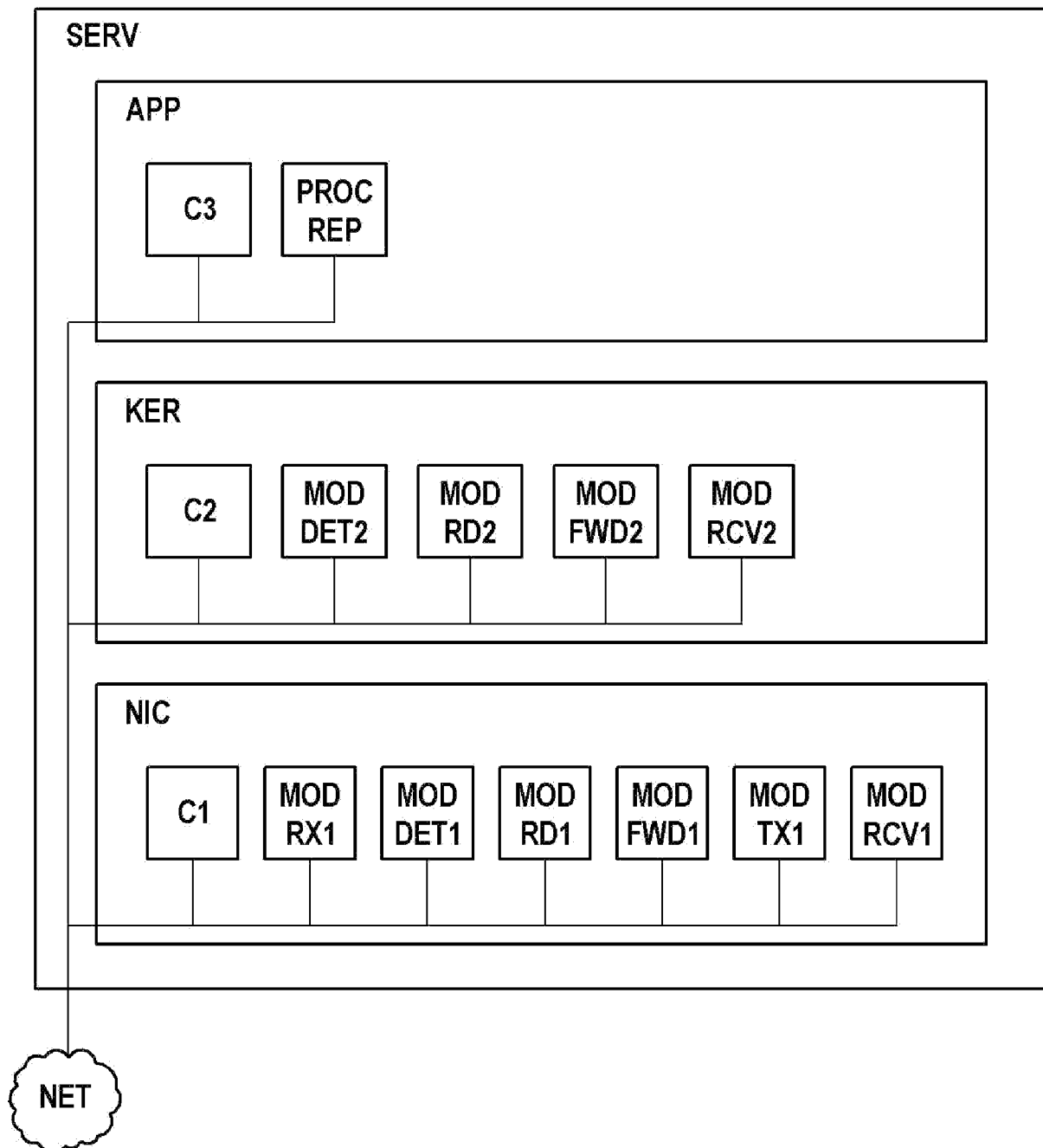
[Fig. 3B]



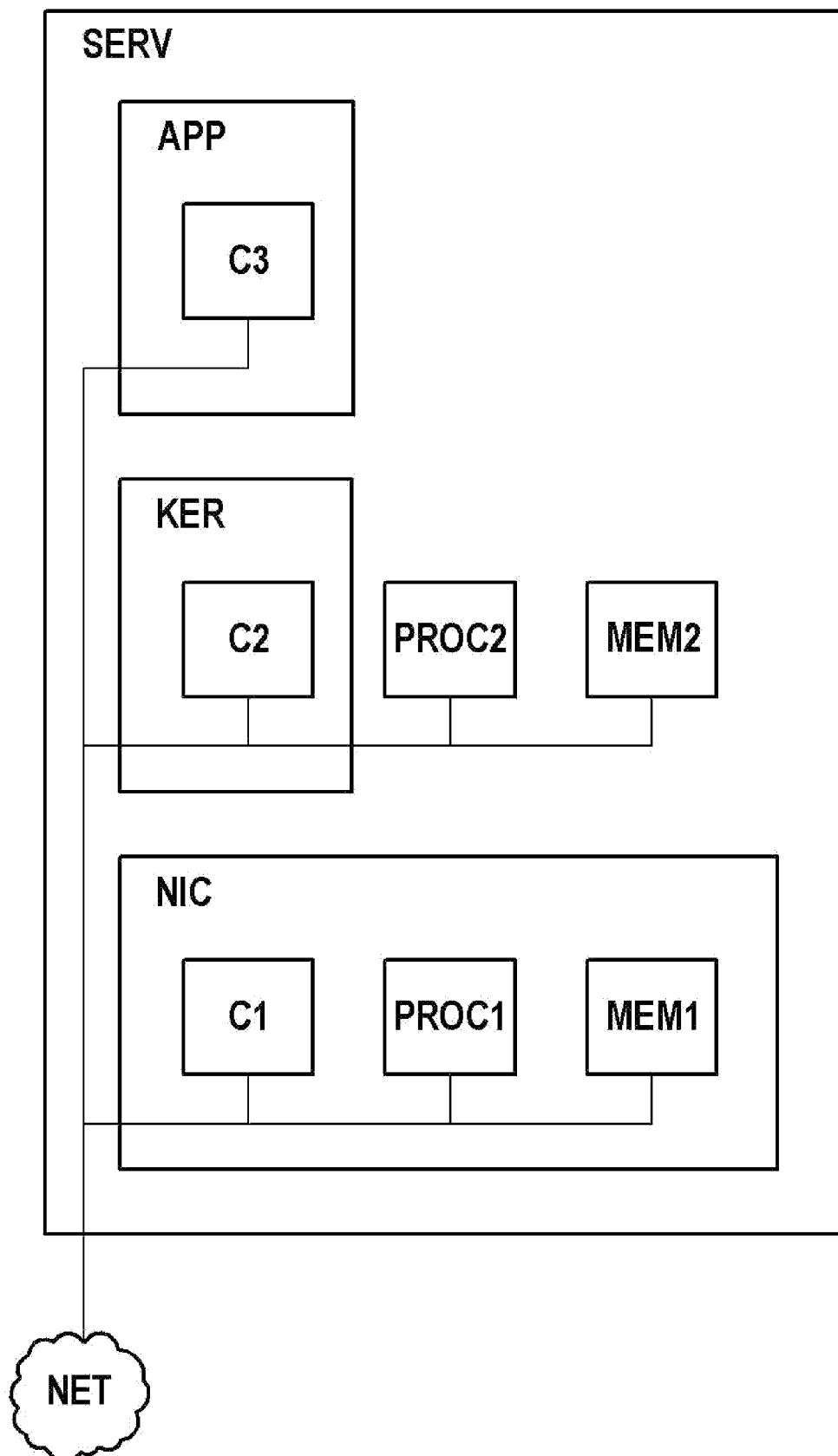
[Fig. 4]



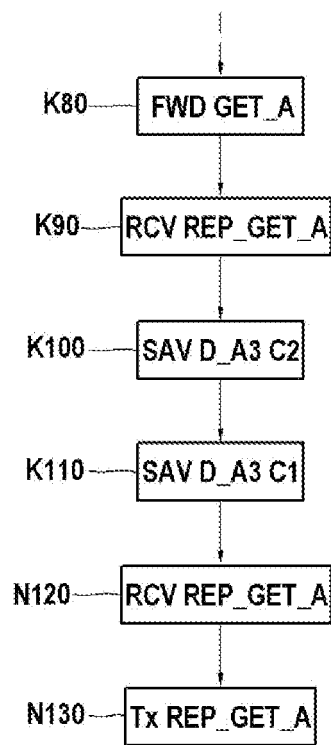
[Fig. 5A]



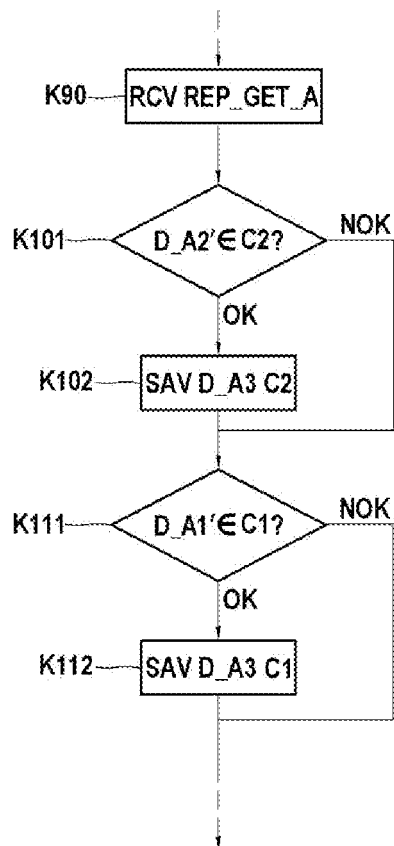
[Fig. 5B]



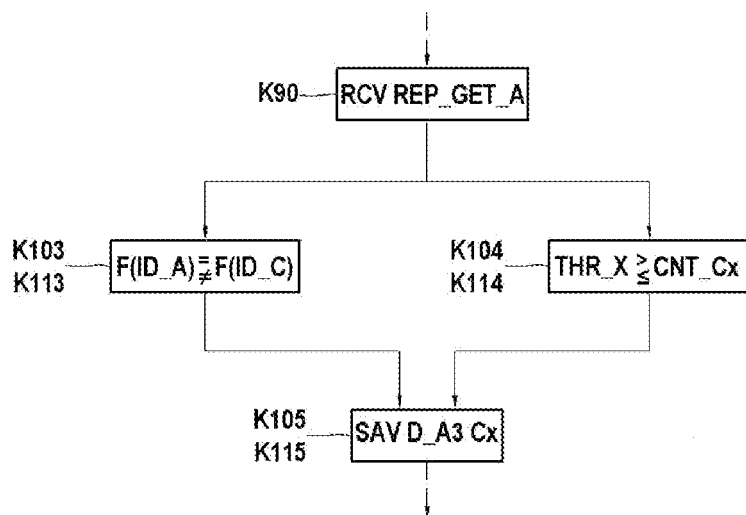
[Fig. 6A]



[Fig. 6B]



[Fig. 6C]



[Fig. 7A]

C1

ID_A	D_A1	V_A1	CNT_A1
ID_B	D_B1	V_B1	CNT_B1

[Fig. 7B]

C1 : TAB1

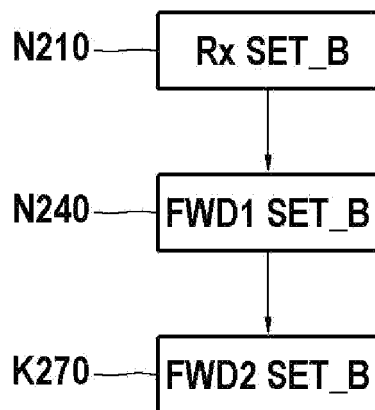
H_A1 →	ID_A	D_A1
H_B1 →	ID_B	D_B1
H_C1 →	ID_C	D_C1

[Fig. 7C]

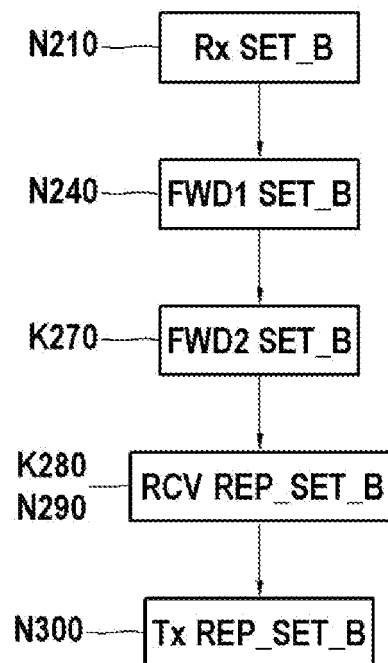
C2 : TAB2

H_A2 →	ID_A	D_A2	V_A2	CNT_A2	CHA_A2
H_B2 →	ID_B	D_B2	V_B2	CNT_B2	CHA_B2
H_C2 →	ID_C	D_C2	V_C2	CNT_C2	CHA_C2

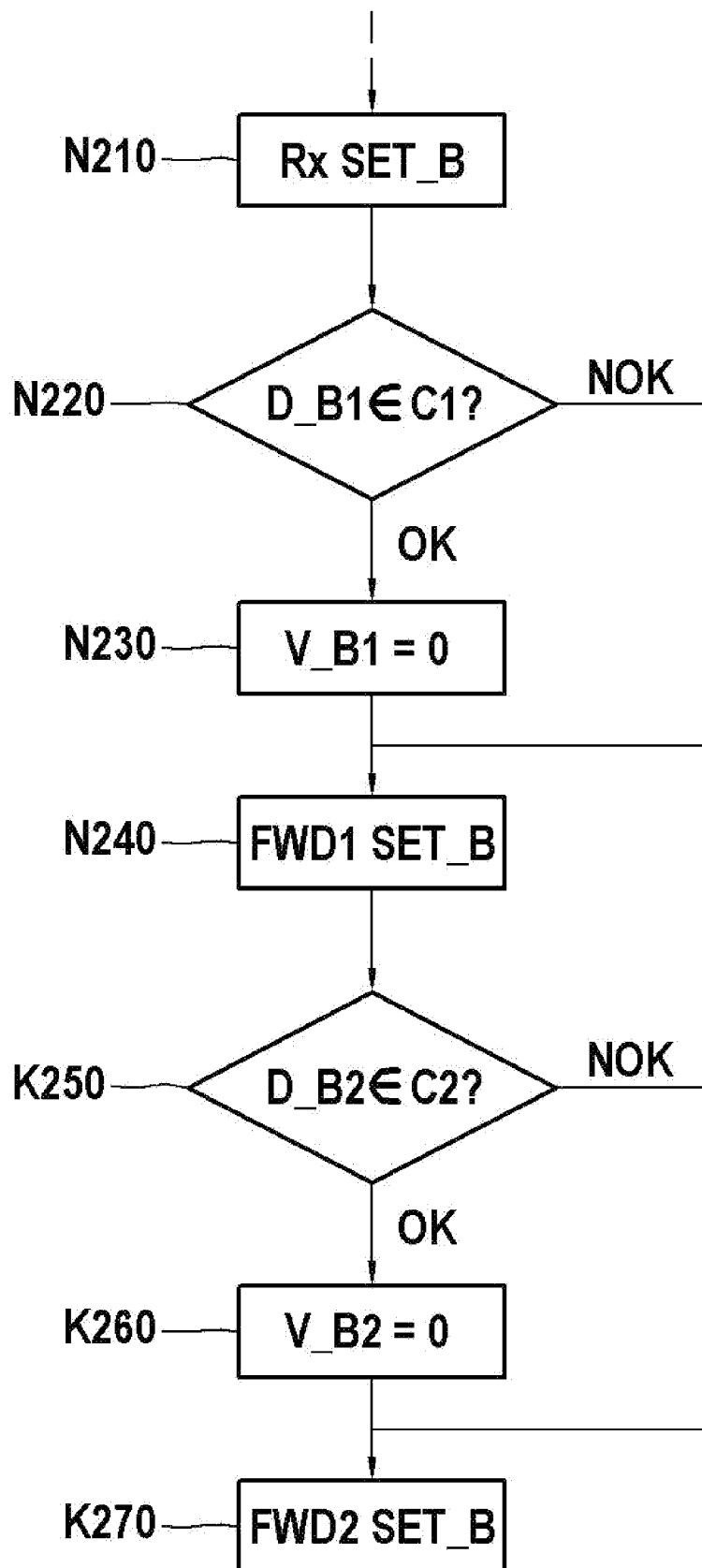
[Fig. 8A]



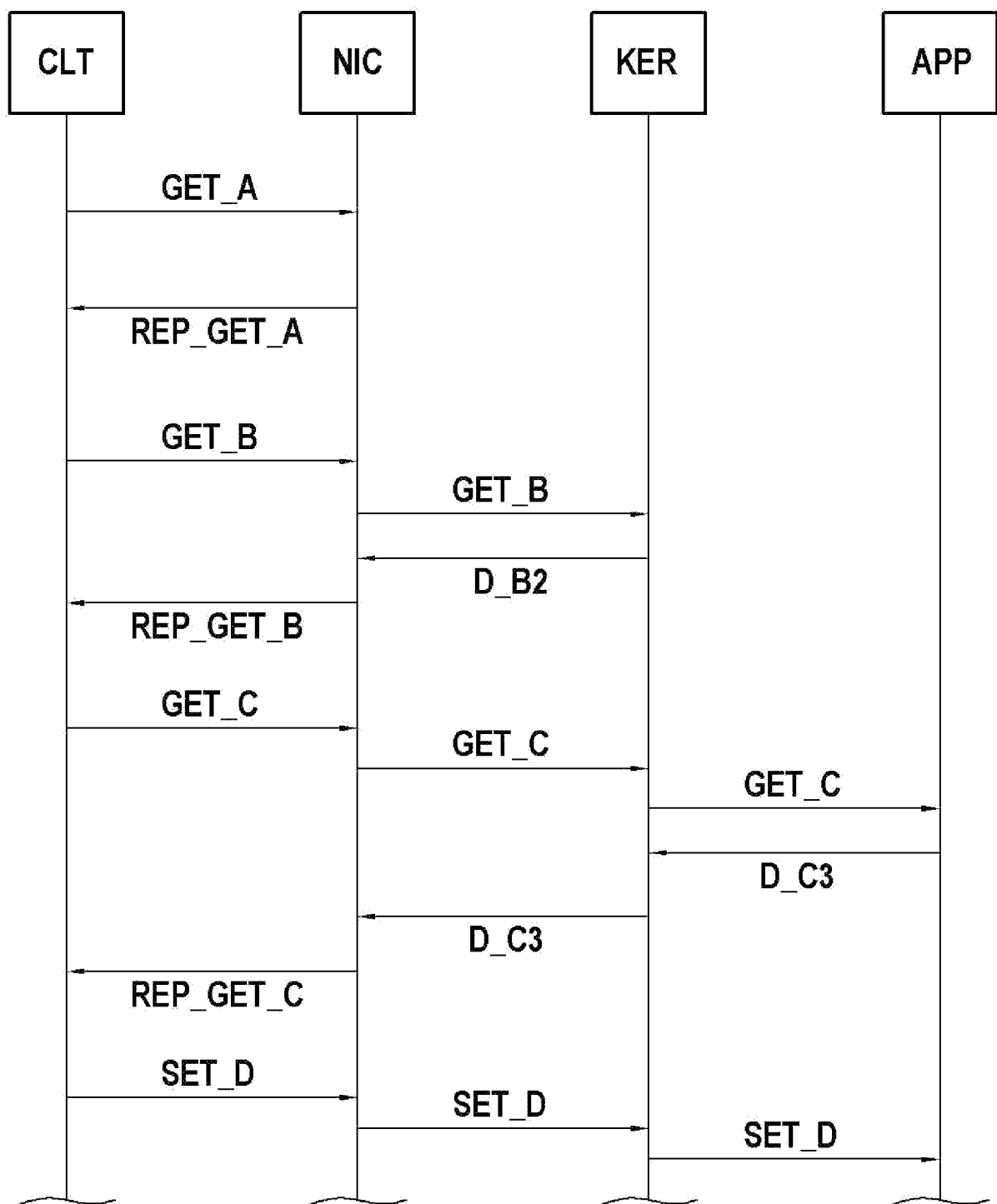
[Fig. 8B]



[Fig. 8C]



[Fig. 9]



RAPPORT DE RECHERCHE PRÉLIMINAIRE

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

N° d'enregistrement
national

FA 905664
FR 2202542

DOCUMENTS CONSIDÉRÉS COMME PERTINENTS		Revendication(s) concernée(s)	Classement attribué à l'invention par l'INPI
Catégorie	Citation du document avec indication, en cas de besoin, des parties pertinentes		
X	US 2020/314011 A1 (DEVAL MANASI [US] ET AL) 1 octobre 2020 (2020-10-01) * figures 1,1a,1b * * alinéa [0030] – alinéa [0044] * * alinéa [0086] * -----	1-12	H04L47/12 H04L47/00
X	TOKUSASHI YUTA ET AL: "LaKe: The Power of In-Network Computing", 2018 INTERNATIONAL CONFERENCE ON RECONFIGURABLE COMPUTING AND FPGAS (RECONFIG), IEEE, 3 décembre 2018 (2018-12-03), pages 1-8, XP033518143, DOI: 10.1109/RECONFIG.2018.8641696 [extrait le 2019-02-13] * section II * -----	1-12	
X	HERBERT BOS ET AL: "CacheCard", ARCHITECTURES FOR NETWORKING AND COMMUNICATIONS SYSTEMS, ACM, 2 PENN PLAZA, SUITE 701 NEW YORK NY 10121-0701 USA, 19 octobre 2009 (2009-10-19), pages 1-10, XP058323579, DOI: 10.1145/1882486.1882491 ISBN: 978-1-60558-630-4 * abrégé * * section 2 * -----	11,12	DOMAINES TECHNIQUES RECHERCHÉS (IPC) H04L
Date d'achèvement de la recherche		Examineur	
13 octobre 2022		De Poy, Iker	
CATÉGORIE DES DOCUMENTS CITÉS X : particulièrement pertinent à lui seul Y : particulièrement pertinent en combinaison avec un autre document de la même catégorie A : arrière-plan technologique 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			

RELATIF A LA DEMANDE DE BREVET FRANÇAIS NO. FR 2202542 FA 905664

La présente annexe indique les membres de la famille de brevets relatifs aux documents brevets cités dans le rapport de recherche préliminaire visé ci-dessus.

Les dits membres sont contenus au fichier informatique de l'Office européen des brevets à la date du **13-10-2022**

Les renseignements fournis sont donnés à titre indicatif et n'engagent pas la responsabilité de l'Office européen des brevets, ni de l'Administration française

US 2020314011 A1 01-10-2020 AUCUN