(19) RÉPUBLIQUE FRANÇAISE

INSTITUT NATIONAL
DE LA PROPRIÉTÉ INDUSTRIELLE

COURBEVOIE

(11) **N° de publication :** ***3 139 216***

(à n'utiliser que pour les commandes de reproduction)

(21) **N° d'enregistrement national :** **22 08639**

(51) Int Cl[8] : ***G 06 F 8/658*** *(2022.01), G 06 F 8/00*

(12) **BREVET D'INVENTION** **B1**

(54) Procédé de génération d'un fichier de mise à jour et dispositif serveur correspondant, procédé de mise à jour et dispositif client correspondant, méthode de mise à jour et système correspondant.

(22) **Date de dépôt :** 29.08.22.

(30) **Priorité :**

(43) **Date de mise à la disposition du public de la demande :** 01.03.24 Bulletin 24/09.

(45) **Date de la mise à disposition du public du brevet d'invention :** 18.10.24 Bulletin 24/42.

(56) **Liste des documents cités dans le rapport de recherche :**

Se reporter à la fin du présent fascicule

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

(○) **Demande(s) d'extension :**

(71) **Demandeur(s) :** *STMICROELECTRONICS (ROUSSET) SAS Société par actions simplifiée (SAS)* — FR.

(72) **Inventeur(s) :** BOUVET Yoann et COUPIGNY Jean-Paul.

(73) **Titulaire(s) :** STMICROELECTRONICS (ROUSSET) SAS Société par actions simplifiée (SAS).

(74) **Mandataire(s) :** Casalonga.

FR 3 139 216 - B1

Description

Titre de l'invention : Procédé de génération d'un fichier de mise à jour et dispositif serveur correspondant, procédé de mise à jour et dispositif client correspondant, méthode de mise à jour et système correspondant.

[0001] Des modes de réalisation et de mise en œuvre concernent les techniques de mise à jour d'un logiciel, notamment la génération d'un fichier de mise à jour par un dispositif serveur, et la mise à jour du logiciel par un dispositif client.

[0002] Les techniques de mise à jour de logiciel peuvent être des fonctionnalités critiques en termes de bande passante de consommation d'énergie pour transmettre le fichier de mise à jour aux dispositifs clients, en particulier pour des cas ayant une bande passante limitée, et visant l'économie d'énergie.

[0003] Ceci est par exemple le cas dans des campagnes de mise à jour par voie hertzienne « OTA » (acronyme des termes anglais usuels « Over The Air ») de dispositifs client du type internet des objets « IoT » (acronyme des termes anglais usuels « Internet of Things ») déployés sur le terrain, où typiquement la bande passante du réseau est limitée et coûteuse et où typiquement les dispositif client IoT tendent à être très économes en énergie.

[0004] Par exemple les réseaux ayant une bande passante limitée et visant l'économie d'énergie, peuvent être du type « LPWAN » (acronyme des termes anglais « Low Power Wide Area Network ») tels que « LoRaWAN », « BLE », « Cat-M1 », « NB-IoT » (respectivement pour « Long Range WAN », « Bluetooth Low Energy », « Machine Type Communication, Category 1 » et « Narrow Band-IoT » en anglais), ou encore du type « Wi-Fi » (le protocole de communication régis par les normes du groupe IEEE 802.11).

[0005] Réduire la quantité de donnée transmises a un effet considérable sur les économies d'énergie lors de la réception du fichier de mise à jour par le dispositif client IoT et aussi sur le coût de la bande passante associée.

[0006] Cela étant, dans l'absolu il est toujours avantageux de limiter la quantité de donnée transmises et la consommation d'énergie, même dans les systèmes de communication n'étant pas particulièrement limités en la matière (par exemple dans les communications du type « LTE », « 4G », « 5G » respectivement pour « Long Term Evolution », « 4th Generation », « 5th Generation » en anglais), pour des raisons d'efficacité et/ou de sobriété dans l'usage des ressources disponibles.

[0007] La [Fig.1] illustre un exemple de technique classique permettant de limiter la quantité de donnée dans un fichier de mise à jour FWupdt, d'une ancienne version d'un logiciel

FW_vN vers une nouvelle version FWvN+1. Différents modules App, UTIL, HAL, SBSFU et différentes fonctions MWfunc1, MWfunc2_vN, …, MWfuncN du logiciel occupent des régions mémoires, représentées par les rectangles respectifs, d'une mémoire.

[0008] Le fichier de mise à jour FWupdt communique les régions mémoires qui ont été modifiées « XX..X » de l'ancienne version à la nouvelle version du logiciel, et pas celles qui sont inchangées « 0 ». Cela étant, si une région mémoire est ajoutée ou retirée dans le volume de la mémoire, ou si la modification d'une fonction MWfunc2_vN+1 modifie la taille de la région mémoire correspondante (plus grande ou plus petite), alors toutes les régions mémoires suivantes sont décalées MWfunc2_vN+1, …, MWfuncN, UTIL, HAL, SBSFU, même si leur contenu n'est pas modifié dans la nouvelle version. En effet, les mécanismes de mappage en mémoire classiques attribuent des emplacements mémoires à des positions successives dans la mémoire, aux fonctions successives du code. Par conséquent, les régions mémoires positionnées successivement après l'emplacement d'une modification sont typiquement toutes décalées, et sont communiquées en tant que régions mémoires modifiées « XX..X » dans le fichier de mise à jour FWupdt.

[0009] D'autre part, les techniques classiques de compression, par exemple typiquement la compression « ZIP » ou « RAR », présentent des limites en performances et nécessitent de munir les dispositifs clients d'un moyen de décompression correspondant. Or, ce type de moyen de décompression peut représenter une relativement grande partie, par exemple 10% à 15%, du code enregistré dans la mémoire, notamment dans les dispositifs IoT les plus simples et les moins onéreux.

[0010] Il existe donc un besoin d'améliorer les techniques de réduction de la quantité de donnée transmises dans un fichier de mise à jour d'un logiciel, et un besoin de réduire les ressources de mémoire.

[0011] Des modes de mise en œuvre et de réalisation proposés ci-dessous permettent d'envoyer un fichier de mise à jour extrêmement petit, ne comportant quasiment que les différences binaires entre les versions du logiciel, d'une manière compatible avec les architectures actuelles et usuelles des logiciels. En outre, la « décompression » du fichier de mise à jour est simple et demande peu de ressource, et l'enregistrement de la mise à jour peut être fait « sur place » directement dans la mémoire, sans nécessiter un espace mémoire « tampon » de la taille du logiciel entier.

[0012] Selon un aspect, il est proposé à cet égard un procédé de génération d'un fichier de mise à jour d'une version antérieure d'un logiciel vers une version ultérieure du logiciel, comprenant :

- une compilation d'un code source générant un fichier binaire de la version ultérieure ;

- une édition de liens attribuant des emplacements mémoires d'une mémoire à des sections du fichier binaire de la version ultérieure, les emplacements mémoires desdites sections de la version ultérieure étant contraints à l'identique des emplacements mémoires des sections correspondantes de la version antérieure ;
- une comparaison entre le fichier binaire de la version ultérieure et un fichier binaire de la version antérieure ; et
- une construction du fichier de mise à jour comportant les différences, par régions mémoires, entre lesdits fichiers binaires comparés.

[0013] On pourra désigner ladite « édition de lien » par le terme « mappage ».

[0014] Ainsi, puisque le mappage des données binaires de la version ultérieure est contraint par le mappage de la version antérieur, alors, en particulier, une modification d'une section n'engendre pas de décalage des sections situées à la suite de cette section modifiée et qui n'ont pas été modifiées, les positions respectives des sections étant contraintes, c'est-à-dire imposées, lors du mappage. En conséquence, seules les régions mémoires du fichier binaire qui correspondent à des portions du code source effectivement modifiées, sont effectivement modifiées après le mappage. Ainsi, seules ces régions mémoires du fichier binaire, sont communiquées dans le fichier de mise à jour.

[0015] Par ailleurs, on entend qu'une section correspond à la taille en mémoire permettant de loger le code d'un objet élémentaire (par exemple une fonction ou un scalaire) issu du code source. Typiquement dans ce cas, une section peut avoir une taille comprise entre 8 bits et n*8 bits (un multiple entier « n » d'un octet de 8 bits) lorsqu'un octet de 8 bits est la taille élémentaire du processeur. Les régions mémoires correspondent quant à elles à la granularité dans laquelle sont communiquées les différences. Dans l'absolu, les régions mémoires peuvent avoir une taille comprise entre 1 bit et une page mémoire entière (c'est à dire la taille maximale accessible dans la mémoire). En pratique, les régions mémoires peuvent avoir une taille de 16 octets, ou plus.

[0016] Selon un mode de mise en œuvre, le fichier de mise à jour ne comporte pas les régions mémoires ne présentant pas de différences entre lesdits fichiers binaires comparés, et dans lequel le fichier de mise à jour comporte une identification des emplacements mémoires, dans le fichier binaire de la version antérieure, des régions mémoires présentant lesdites différences entre lesdits fichiers binaires comparés.

[0017] Cela correspond à une technique de compression dans laquelle le fichier de mise à jour ne comporte pas d'information concernant les régions mémoires ne présentant pas de différence par rapport à la version antérieure.

[0018] Selon un mode de mise en œuvre :

- ladite comparaison entre lesdits fichiers binaires est faite bit-à-bit par un opérateur ou-exclusif ; et
- ladite construction du fichier de mise à jour comporte les résultats de la com-

paraison bit-à-bit par l'opérateur ou-exclusif, pour communiquer lesdites différences entres lesdits fichiers binaires comparés.

[0019] L'opérateur ou-exclusif permet avantageusement d'introduire les différences directement dans la mémoire recevant la mise à jour, sans avoir besoin de stocker une image entière du fichier binaire de la version ultérieure, du fait que l'opérateur ou-exclusif est son propre inverse.

[0020] Selon un mode de mise en œuvre, si le fichier binaire de la version ultérieure comporte une section de plus grande taille en mémoire que l'emplacement mémoire attribué à cette section dans la version antérieure, alors l'édition de liens attribue un nouvel emplacement mémoire, libre dans la version antérieure, à ladite section plus grande, et introduit une instruction d'appel vers le nouvel emplacement mémoire, dans l'emplacement mémoire attribué à cette section dans la version antérieure.

[0021] Ainsi, aucune section située entre l'emplacement mémoire attribué à la section modifiée dans la version antérieure et le nouvel emplacement mémoire n'est décalée ou modifiée, malgré la plus grande taille de la section modifiée.

[0022] Selon un mode de mise en œuvre, si le fichier binaire de la version ultérieure comporte une section de plus petite taille en mémoire que l'emplacement mémoire attribué à cette section dans la version antérieure, alors l'édition de liens attribue le même emplacement mémoire ayant cette même taille et laisse vide l'excédent en taille de l'emplacement mémoire.

[0023] Ainsi, aucune section située après l'emplacement mémoire attribué à la section modifiée dans la version antérieure n'est décalée ou modifiée, malgré la plus petite taille de la section modifiée.

[0024] En particulier dans ce mode de mise en œuvre, la section modifiée pourrait être entièrement effacée, et ainsi avoir une taille nulle en mémoire. Dans ce cas, le mappage attribue le même emplacement mémoire ayant la même taille que dans la version antérieure, en laissant vide cet emplacement mémoire dans la version ultérieure.

[0025] Selon un autre aspect, il est proposé un procédé de mise à jour d'une version antérieure d'un logiciel contenue dans une mémoire vers une version ultérieure du logiciel, comprenant :

- une réception d'un fichier de mise à jour comportant les différences, par régions mémoires, entre un fichier binaire de la version ultérieure et un fichier binaire de la version antérieure ;
- un remplacement, dans la partie de la mémoire contenant le fichier binaire de la version antérieure, des bits des régions mémoires présentant des différences par les bits des régions mémoires du fichier binaire de la version ultérieure.

[0026] En d'autres termes, il est proposé de remplacer les bits du fichier binaires directement dans la mémoire, pour mettre à jour uniquement les différences du point de

vue des données binaires contenues dans la mémoire, et non pas du point de vue « fonctionnel » des objets du code (fonctions et données scalaires) qui ont été modifiés dans la version ultérieure.

[0027] Selon un mode de mise en œuvre :

- ledit fichier de mise à jour ne comporte pas les régions mémoires ne présentant pas de différences entre lesdits fichiers binaires comparés, et comporte une identification des emplacements mémoires, dans le fichier binaire de la version antérieure, des régions mémoires présentant lesdites différences entre lesdits fichiers binaires comparés ;
- ledit remplacement comprend une lecture de ladite identification et un accès sélectif auxdits emplacements mémoire identifiés.

[0028] Ce mécanisme de « décompression », correspondant sommairement à un accès à une adresse de la mémoire, présente l'avantage d'être extrêmement simple à mettre en œuvre, et de ne pas demander de ressource calculatoire tel qu'usuellement dans les mécanismes de décompression algorithmique (par exemple du type « zip » ou « rar »).

[0029] Selon un mode de mise en œuvre :

- ledit fichier de mise à jour comporte le résultat d'une comparaison bit-à-bit par un opérateur ou-exclusif entre le fichier binaire de la version ultérieure et le fichier binaire de la version antérieure, pour communiquer lesdites différences entres lesdits fichiers binaires comparés ;
- ledit remplacement comprend une transformation des bits contenus dans lesdites régions mémoires du fichier binaire de la version antérieure par l'opérateur ou-exclusif avec les bits dudit résultat de la comparaison bit-à-bit par l'opérateur ou-exclusif entre lesdits fichiers binaires.

[0030] Selon un mode de mise en œuvre du procédé de mise à jour d'une version antérieure d'un logiciel vers une version ultérieure du logiciel, la version antérieure ayant été mise à jour à partir d'une version précédente avec le fichier de mise à jour antérieur, et la version ultérieure correspondant à un retour à ladite version précédente :

- le fichier de mise à jour est le fichier mise à jour antérieur ;
- ledit remplacement comprend une transformation des bits contenus dans lesdites régions mémoires du fichier binaire de la version antérieure par l'opérateur ou-exclusif avec les bits desdites régions du résultat de la comparaison du fichier de mise à jour antérieur.

[0031] Selon un autre aspect, il est également proposé une méthode de mise à jour d'une version antérieure d'un logiciel vers une version ultérieure du logiciel, pour au moins un dispositif, comprenant :

- un procédé de génération d'un fichier de mise à jour de la version antérieure d'un logiciel vers la version ultérieure du logiciel tel que défini ci-avant, par un dispositif

serveur ;

- une communication audit au moins un dispositif du fichier de mise à jour, par un réseau de communication ;

- un procédé de mise à jour de version antérieure d'un logiciel vers la version ultérieure du logiciel tel que défini ci-avant, par ledit au moins un dispositif client.

[0032] Par exemple, la communication peut être adaptée pour un réseau de communication du type « LPWAN » (pour « Low Power Wide Area Network » en anglais) tel que « LoRaWAN », « BLE », « Cat-M1 », « NB-IoT » (respectivement pour « Long Range WAN », « Bluetooth Low Energy », « Machine Type Communication, Category 1 » et « Narrow Band-IoT » en anglais), mais aussi du type « Wi-Fi » (définit par la norme IEEE 802.11), du type « LTE 4G/5G » (pour « Long Term Evolution, 4th/5th generation ») ou du type connexion filaire tel qu'Ethernet ou USB.

[0033] Selon un autre aspect, il est également proposé un dispositif serveur, apte à générer un fichier de mise à jour d'une version antérieure d'un logiciel vers une version ultérieure du logiciel, comprenant :

- un compilateur configuré pour compiler un code source de manière à générer un fichier binaire de la version ultérieure ;

- un éditeur de lien configuré pour attribuer des emplacements mémoires d'une mémoire à des sections du fichier binaire de la version ultérieure, de sorte que les emplacements mémoires desdites sections de la version ultérieure sont contraints à l'identique des emplacements mémoires des sections correspondantes de la version antérieure ;

- un compresseur configuré pour :

-- comparer le fichier binaire de la version ultérieure avec un fichier binaire de la version antérieure ; et pour

-- construire le fichier de mise à jour comportant les différences, par régions mémoires, entre lesdits fichiers binaires comparés.

[0034] Selon un mode de réalisation, le compresseur est configuré pour construire le fichier de mise à jour ne comportant pas les régions mémoires ne présentant pas de différences entre lesdits fichiers binaires comparés, et pour construire le fichier de mise à jour comportant une identification des emplacements mémoires, dans le fichier binaire de la version antérieure, des régions mémoires présentant lesdites différences entre lesdits fichiers binaires comparés.

[0035] Selon un mode de réalisation, le compresseur est configuré pour :

- comparer lesdits fichiers binaires bit-à-bit par un opérateur ou-exclusif ; et

- construire le fichier de mise à jour comportant les résultats de la comparaison bit-à-bit par l'opérateur ou-exclusif, pour communiquer lesdites différences entres lesdits fichiers binaires comparés.

[0036] Selon un mode de réalisation, l'éditeur de lien est configuré, si le fichier binaire de la version ultérieure comporte une section de plus grande taille en mémoire que l'emplacement mémoire attribué à cette section dans la version antérieure, pour attribuer un nouvel emplacement mémoire, libre dans la version antérieure, à ladite section plus grande, et introduire une instruction d'appel vers le nouvel emplacement mémoire, dans l'emplacement mémoire attribué à cette section dans la version antérieure.

[0037] Selon un mode de réalisation, l'éditeur de lien est configuré, si le fichier binaire de la version ultérieure comporte une section de plus petite taille en mémoire que l'emplacement mémoire attribué à cette section dans la version antérieure, pour attribuer le même emplacement mémoire ayant cette même taille et laisser vide l'excédent en taille de l'emplacement mémoire.

[0038] Selon un autre aspect, il est également proposé un dispositif client apte à mettre à jour une version antérieure d'un logiciel contenue dans une mémoire vers une version ultérieure du logiciel, comprenant :

- des moyens de réception configurés pour recevoir un fichier de mise à jour comportant les différences, par régions mémoires, entre un fichier binaire de la version ultérieure et un fichier binaire de la version antérieure ;
- des moyens de traitement configurés pour remplacer, dans la partie de la mémoire contenant le fichier binaire de la version antérieure, des bits des régions mémoires présentant des différences par les bits des régions mémoires du fichier binaire de la version ultérieure.

[0039] Selon un mode de réalisation :

- les moyens de réception sont configurés pour recevoir le fichier de mise à jour ne comportant pas les régions mémoires ne présentant pas de différences entre lesdits fichiers binaires comparés, et comportant une identification des emplacements mémoires, dans le fichier binaire de la version antérieure, des régions mémoires présentant lesdites différences entre lesdits fichiers binaires comparés ;
- les moyens de traitement sont configurés pour lire ladite identification et accéder sélectivement auxdits emplacements mémoire identifiés.

[0040] Selon un mode de réalisation :

- les moyens de réception sont configurés pour recevoir fichier de mise à jour comportant le résultat d'une comparaison bit-à-bit par un opérateur ou-exclusif entre le fichier binaire de la version ultérieure et le fichier binaire de la version antérieure, pour communiquer lesdites différences entres lesdits fichiers binaires comparés ;
- les moyens de traitement sont configurés pour transformer les bits contenus dans lesdites régions mémoires du fichier binaire de la version antérieure par l'opérateur ou-exclusif avec les bits dudit résultat de la comparaison bit-à-bit par l'opérateur ou-

exclusif entre lesdits fichiers binaires.

[0041] Selon un mode de réalisation, dans le dispositif client apte à mettre à jour une version antérieure d'un logiciel vers une version ultérieure du logiciel, la version antérieure ayant été mise à jour à partir d'une version précédente avec le fichier de mise à jour antérieur, et la version ultérieure correspondant à un retour à ladite version précédente :

- les moyens de réception sont configurés pour récupérer le fichier mise à jour antérieur en tant que fichier de mise à jour ;
- les moyens de traitement sont configurés pour transformer les bits contenus dans lesdites régions mémoires du fichier binaire de la version antérieure par l'opérateur ou-exclusif avec les bits dudit résultat du fichier de mise à jour antérieur.

[0042] Selon un autre aspect, il est également proposé un système comportant un dispositif serveur tel que défini ci-avant et au moins un dispositif client tel que défini ci-avant, le dispositif serveur étant apte à communiquer ledit fichier de mise à jour de la version antérieure d'un logiciel vers le version ultérieure du logiciel, audit au moins un dispositif client par un réseau de communication, et le dispositif client étant apte à mettre à jour la version antérieure d'un logiciel vers la version ultérieure du logiciel avec ce fichier de mise à jour.

[0043] D'autres avantages et caractéristiques de l'invention apparaîtront à l'examen de la description détaillée de modes de mise en œuvre et de réalisation, nullement limitatifs, et des dessins annexés sur lesquels :

[0044] [Fig.1] précédemment décrite, illustre un cas de l'art antérieur ;

[0045] [Fig.2] ;

[0046] [Fig.3] ;

[0047] [Fig.4] ;

[0048] [Fig.5] ;

[0049] [Fig.6] ;

[0050] [Fig.7] ;

[0051] [Fig.8] ;

[0052] [Fig.9] ;

[0053] [Fig.10] illustrent des modes de mise en œuvre et de réalisation de l'invention.

[0054] La [Fig.2] illustre un procédé de génération d'un fichier de mise à jour d'une version antérieure d'un logiciel vers une version ultérieure du logiciel. Le logiciel est par exemple un micrologiciel (usuellement « firmware » en anglais) d'un dispositifs clients du type IoT, tandis que la génération du fichier de mise à jour peut être mise en œuvre par un dispositif serveur, apte à effectuer des campagnes de mise à jour des logiciels des dispositif clients à distance.

[0055] Le procédé de génération du fichier de mise à jour comprend :

- une étape 310, décrite en détails ci-après en relation avec la [Fig.3], dans laquelle

un code source est compilée, générant un fichier binaire de la version ultérieure ;
- une étape 320, décrite en détails ci-après en relation avec la [Fig.3], dans laquelle une édition de liens, ou « un mappage », est effectuée pour une mémoire des dispositifs clients, de manière à attribuer des emplacements mémoires à des sections du fichier binaire de la version ultérieure, les emplacements mémoires desdites sections de la version ultérieure étant contraints à l'identique des emplacements mémoires des sections correspondantes de la version antérieure ;
- une étape 410, décrite en détails ci-après en relation avec la [Fig.4], dans laquelle le fichier binaire de la version ultérieure et un fichier binaire de la version antérieure sont comparés ; et
- une étape 420, décrite en détails ci-après en relation avec la [Fig.4], dans laquelle le fichier de mise à jour est construit, de sorte que le fichier de mise à jour comporte les différences, par régions mémoires, entre lesdits fichiers binaires comparés.

[0056] On notera que le procédé de génération du fichier de mise à jour permet de minimiser la quantité de données transmises avec une approche « informatique », c'est à dire lors de la compilation et du mappage du code ; contrairement aux techniques classiques de correctifs superposés à l'ancienne version (« patching » en anglais) ou insérés dans des emplacements libres prévus à cet égard dans l'ancienne version (« padding » en anglais), qui sont fait avec une approche de « programmation » lors de l'écriture du code.

[0057] La [Fig.3] illustre l'étape de compilation 310 et l'étape d'édition de liens 320, ou « de mappage » 320, du procédé décrit en relation avec la [Fig.2], par exemple mis en œuvre au sein d'un dispositif serveur. Le dispositif serveur peut comporter à cet égard un moyen BLD_bin apte à la construction du fichier binaire selon les étapes 310 et 320. Le moyen BLD_bin peut être un moyen matériel du type unité de traitement ou processeur, ou un moyen logiciel mis en œuvre par un dispositif matériel pluripotent, tel qu'une unité de traitements ou un processeur.

[0058] D'une part, un compilateur CMPLR est configuré pour compiler un code source vN+1_code de manière à générer un fichier binaire vN+1_bin de la version ultérieure du logiciel FW_vN+1, selon l'étape 310.

[0059] Le code source vN+1_code est écrit dans un langage de programmation « lisible par l'Homme » (.s / .c / .h), tel que le C, C++, Python, ou autres. Le programmeur qui développe le code source vN+1_code de la version ultérieure n'a pas de contrainte particulière à respecter pour obtenir une génération d'un fichier de mise à jour d'une quantité de données minimale.

[0060] Le fichier binaire vN+1_bin, également appelé fichier exécutable, est un fichier dont toute les références ont été résolues et qui peut être placé en mémoire pour être exécuté. Le fichier binaire vN+1_bin est écrit dans un « langage machine », et

comporte des instructions ou fonctions élémentaires, codées en binaire dans des sections de mémoire. Une section du fichier binaire correspond à une entité logique élémentaire contenant une séquence d'octets correspondant à un morceau de données ou de code qui doit être placé à un emplacement physique de la mémoire. Typiquement, une section peut avoir une taille définie en octets, voire d'au moins 4 octets, et par exemple une section peut correspondre à un mot de 32 bits (4 octets) pour coder une instruction de saut ou d'appel vers une adresse mémoire, ainsi que les 32 bits pour identifier ladite adresse.

[0061] D'autre part, un éditeur de lien LNKR est configuré pour effectuer le mappage du fichier binaire vN+1_bin, c'est-à-dire attribuer des emplacements mémoires, dans la mémoire destinée à contenir le logiciel, à chaque section du fichier binaire vN+1_bin généré par le compilateur CMPLR, selon l'étape 320.

[0062] L'éditeur de lien LNKR est en particulier configuré pour effectuer un mappage statique contraint par un fichier de configuration vN_ScttrF. Le fichier de configuration contraignant l'éditeur de lien LNKR correspond au fichier de mappage vN_ScttrF (usuellement « scatter file » en anglais) des sections de la version antérieure du logiciel FW_vN.

[0063] Ainsi, l'éditeur de lien LNKR est configuré de sorte que les emplacements mémoires des sections de la version ultérieure FW_vN+1 sont forcés à être identique aux emplacements mémoires des sections correspondantes de la version antérieure FW_vN.

[0064] En fait, lors de l'étape de mappage 320, toutes les sections du fichier binaire de la version ultérieure vN+1_bin existant déjà dans le fichier binaire de la version antérieure vN_bin sont placées dans le même emplacement mémoire. Les sections nouvelles du fichier binaire de la version ultérieure vN+1_bin peuvent être placées à des emplacements mémoires nouveaux (voir ci-après « #200 » en relation avec la [Fig.6]).

[0065] Par ailleurs, le fichier de mappage vN+1_ScttrF des sections du fichier de la version ultérieure vN+1_bin est extrait par un moyen adapté à cet égard MAP EXTRCT, afin notamment de contraindre le mappage d'une éventuelle future mise à jour à partir de cette version « ultérieure » vN+1.

[0066] La [Fig.4] illustre l'étape de comparaison 410 et l'étape de construction 420 du procédé décrit en relation avec la [Fig.2], par exemple mis en œuvre au sein du dispositif serveur. Le dispositif serveur peut comporter à cet égard un moyen, appelé compresseur, CPRSSR apte à la construction du fichier de mise à jour selon les étapes 410 et 420. Le compresseur CPRSSR peut être un moyen matériel du type unité de traitement ou processeur, ou un moyen logiciel mis en œuvre par un dispositif matériel pluripotent, tel qu'une unité de traitements ou un processeur.

[0067] D'une part, le compresseur CPRSSR est configuré pour comparer, selon l'étape 410,

le fichier binaire de la version ultérieure vN+1_bin, compilé et mappé aux étapes 310 et 320, avec le fichier binaire vN_bin de la version antérieure obtenue antérieurement de la même manière.

[0068] Avantageusement, le moyen de comparaison pour effectuer l'étape 410 est configuré pour effectuer ladite comparaison bit-à-bit par un opérateur ou-exclusif xor, entre lesdits fichiers binaires N+1_bin, vN_bin.

[0069] Le résultat (qu'on appellera « fichier de résultat non compressé ») ΔFW sortant de la comparaison bit-à-bit par l'opérateur ou-exclusif xor est un fichier binaire ayant la taille du fichier binaire le plus grand parmi la version antérieure vN et la version ultérieure vN+1. Chaque bit, dans le fichier de résultat ΔFW, correspond à une position de bits respectifs dans les fichiers binaires comparés vN+1_bin, vN_bin, et la valeur de chaque bit dans le fichier de résultat ΔFW vaut « 0 » lorsqu'à cette position dans les fichiers binaires comparés vN+1, vN, les bits sont identiques ; et vaut « 1 » lorsqu'à cette position dans les fichiers binaires comparés vN+1, vN, les bits sont différents.

[0070] Cela permet en effet de communiquer les différences entres lesdits fichiers binaires comparés, ce qui est suffisant pour mettre à jour tout le fichier binaire à la version ultérieure vN+1_bin, en connaissance de la version antérieure vN_bin.

[0071] En alternative, on notera qu'un mécanisme équivalent d'identification des différences, tel que par l'opérateur non-ou-exclusif fournissant le même résultat dans lequel les « 0 » et les « 1 » sont inversés, pourra également être choisi.

[0072] D'autre part, le compresseur CPRSSR est configuré pour construire, selon l'étape 420, le fichier de mise à jour ΔFW, ΔFW_cpr comportant les différences DAT_diff, par régions mémoires, entre lesdits fichiers binaires comparés vN+1_bin, vN_bin.

[0073] Ainsi, si la comparaison 410 est faite bit-à-bit par un opérateur ou-exclusif xor, alors la construction du fichier de mise à jour ΔFW, ΔFW_cpr comporte avantageusement le fichier de résultat « avant compression » ΔFW obtenus lors de la comparaison bit-à-bit par l'opérateur ou-exclusif xor, ou « fichier de mise à jour non-compressé ΔFW ».

[0074] Cela étant, dans le fichier de mise à jour non-compressé ΔFW, les zéros « 0 », représentatifs d'identités entre les versions, n'ont pas besoin d'être envoyés. Il est ainsi proposé de compresser le fichier de mise à jour ΔFW_cpr en supprimant les zéros « 0 », et en introduisant un en-tête DSCR pour décrire le mappage des différences DAT_diff dans les fichier binaires vN_bin, vN+1_bin. Le compresseur CPRRS peut comporter un moyen de concaténation CONCAT prévu pour générer l'en-tête DSCR et regrouper les groupes de données comportant des différences DAT.

[0075] On se réfère à cet égard à la [Fig.5].

[0076] La [Fig.5] illustre la structure du fichier de mise à jour compressé ΔFW_cpr généré par le moyen de concaténation CONCAT décrit précédemment en relation avec la [Fig.4].

[0077] Dans cet exemple, les différences DAT_diff sont communiqués par régions mémoires, DIFF_1, DIFF_2, …, DIFF_k. Chaque région mémoire DIFF_i, i=[1, 2, …, k], correspond à une portion du fichier de résultat de la comparaison ΔFW, pouvant par exemple être identifié par une adresse de départ Offst_i respective et une longueur Lght_i respective. Les régions mémoires sont choisies pour définir la granularité dans laquelle sont communiquées les différences DIFF_1, DIFF_2, …, DIFF_k. En effet, si un seul bit change dans une région mémoire donnée, alors toute cette région mémoire du fichier de résultat de la comparaison ΔFW est communiqué (c'est-à-dire dans ce cas un seul « 1 » et le reste des « 0 »).

[0078] Dans l'absolu, les régions mémoires peuvent avoir une taille comprise entre 1 bit et une page mémoire entière (c'est à dire la taille maximale accessible dans la mémoire). En pratique, les régions mémoires peuvent avoir une taille définie en octets, et par exemple une taille d'au moins 16 octets.

[0079] Cela permet d'une part de correspondre à une quantité raisonnablement minimale de modifications dans le fichier binaire vN+1_bin se produisant en pratique en cas de modification d'une portion de code raisonnablement minimale ; d'autre part, le seuil sur la taille des régions mémoires, par exemple d'au moins 16 octets, permet de compenser la quantité d'information à ajouter dans l'en-tête DSCR pour identifier lesdites régions mémoires DIFF_1, …, DIFF_k.

[0080] Ainsi, l'en-tête DSCR comporte au moins l'identification des emplacements mémoires, dans le fichier binaire de la version antérieure vN_bin, des régions mémoires présentant lesdites différences DIFF_1, …, DIFF_k.

[0081] Avantageusement, l'en-tête DSCR peut comporter : le nombre « k » de régions mémoires contenant des différences NBR(k) ; la désignation « vN->vN+1 » de la version antérieure vN et de la version ultérieure vN+1 auxquelles s'applique la mise à jour ; une signature Sign_vN, éventuellement tronquée, de la version antérieure du logiciel ; une signature Sign_vN+1, éventuellement tronquée, de la version ultérieure du logiciel ; des couples contenant l'adresses de départ Offst_1, Offst_2, …, Offst_k et la longueur Lgnth_1, Lgnth_2, …, Lgnth_k, pour respectivement chaque région mémoire DIFF_1, DIFF_2, …, DIFF_k. En outre, des champs peuvent être laissés libres dans la structure du fichier de mise à jour compressé ΔFW_cpr, par exemple réservé à d'autres fonctionnalités futures.

[0082] La [Fig.6] illustre, d'un point de vue global, le procédé de génération du fichier de mise à jour ΔFW_cpr et ses avantages, ainsi que des modes de mise en œuvre particuliers et avantageux.

[0083] La version antérieure du logiciel FW_vN et la version ultérieure du logiciel FW_vN+1 sont représentées schématiquement (en [Fig.6] et [Fig.7]) selon les mappages en mémoire respectifs de leurs fichiers binaires, obtenus après compilation

du code source des versions respectives.

[0084] Chaque rectangle App, MWfunc1-MWfuncN, UTIL, HAL, SBSFU, représentent un ou plusieurs emplacement(s) mémoire(s) contenant un module du logiciel App, UTIL, HAL, SBSFU, ou une fonction du logiciel MWfunc1-MWfuncN.

[0085] Les modules et les fonctions sont chacun codés par des objets élémentaires du code binaire, chaque objet étant contenu dans une section de la mémoire, à un emplacement mémoire respectif.

[0086] Etant donné que le mappage des emplacements mémoires des sections de la version ultérieure FW_vN+1 ont tous été contraints, lors des étapes de compilation 310 et mappage 320, à des emplacements identiques aux emplacements des sections correspondantes de la version antérieure FW_vN ; alors les différences DIFF_1, DIFF_2 dans le fichier binaire de la version ultérieur FW_vN+1 ne correspondant qu'à des modifications effectives du code source de la version ultérieur FW_vN+1.

[0087] Le résultat de la comparaison xor de la version ultérieure FW_vN+1 par rapport à la version antérieure FW_vN expose ainsi des différences DIFF_1, DIFF_2, seulement aux endroits du fichier binaire FW_vN+1 correspondant à des modifications effectives du code source de la version ultérieur.

[0088] En effet, la contrainte du mappage du fichier binaire de la version ultérieure FW_vN+1 sur le mappage du fichier binaire de la version antérieure FW_vN permet notamment de ne pas engendrer de décalage de toutes les régions mémoires MWfuncN, UTIL, HAL, SBSFU, positionnées après une fonction modifiée MWfunc2_vN+1 et dont la taille de l'emplacement mémoire correspondant a été modifiée.

[0089] En conséquence, dans les fichiers binaires compilés, la quantité de donnée ayant changé entre la version antérieure FW_vN et la version ultérieure FW_vN+1 est minimisée, et la compression isolant les régions mémoires comportant des différences DIFF_1, DIFF_2 exclusivement permet de générer un fichier de mise à jour extrêmement compact.

[0090] Le fichier de mise à jour ΔFW_cpr est compressé de manière à ne pas comporter les régions mémoires ne présentant pas de différences 0, …, 0 (App, MWfunc1, MWfuncN, UTIL, HAL, SBSFU) entre lesdits fichiers binaires comparés vN+1_bin, vN_bin. Une identification DSCR des emplacements mémoires des régions mémoires présentant lesdites différences DIFF_1, DIFF_2 est communiquée dans le fichier de mise à jour compressé ΔFW_cpr.

[0091] Par ailleurs, afin de permettre toutes les modifications possibles dans le code source, tout en préservant le bénéfice de la contrainte des sections mémoire lors du mappage, il est avantageusement proposé que si le fichier binaire vN+1_bin de la version ultérieure FW_vN+1 comporte une section modifiée MWfunc2_vN+1 de plus grande taille

sz_200 que la taille sz_100 du premier emplacement mémoire #100 attribué à cette section MWfunc2_vN dans la version antérieure FW_vN, alors l'édition de lien LNKR attribue un nouvel emplacement mémoire #200, libre dans la version antérieure FW_vN, à ladite section modifiée MWfunc2_vN+1, plus grande. Parallèlement, le mappage LNKR introduit automatiquement une instruction d'appel ou de saut Call_#200 vers le nouvel emplacement mémoire #200, située dans le premier emplacement mémoire #100.

[0092] Ainsi, on peut modifier la fonction MWfunc2_vN+1 sans aucune contrainte sur la taille sz_200 de la ou des section(s) correspondantes de la version ultérieure FW_vN+1. En outre, la taille de la section de l'instruction d'appel ou de saut Call_#200 est typiquement compatible (c'est-à-dire inférieure ou égale) avec la taille sz_100 d'une section quelconque, contraignant le premier emplacement mémoire #100.

[0093] Dans le même but, si le fichier binaire vN+1_bin de la version ultérieure FW_vN+1 comporte une section modifiée MWfunc2_vN+1 de plus petite taille (non illustré) que la taille sz_100 de l'emplacement mémoire #100 attribué à cette section MWfunc2_vN dans la version antérieure FW_vN, alors le mappage LNKR attribue le même emplacement mémoire #100 ayant cette même taille sz_100, en laissant vide, ou vacant, l'excédent de taille de cet emplacement mémoire.

[0094] Par « laisser vide » ou « laisser vacant », on entend que cette partie de la mémoire n'est pas utilisée pour contenir des informations utiles, cette partie de la mémoire pouvant être par exemple remplie d'une valeur arbitraire, par exemple des zéros « 0 ».

[0095] La [Fig.7] illustre le procédé correspondant de mise à jour de la version antérieure du logiciel FW_vN, contenue dans la mémoire d'un dispositif client, vers une version ultérieure du logiciel FW_vN+1, avec le fichier de mise à jour compressé ΔFW_cpr généré de la façon décrite précédemment en relation avec les figures 2 à 6.

[0096] Le dispositif client comporte ladite mémoire, typiquement une mémoire non-volatile telle qu'un mémoire « Flash » ou « EEPROM » (pour « Electrically Erasable Programmable Read Only Memory » en anglais), dans laquelle est enregistré le fichier binaire de la version antérieure du logiciel FW_vN ; des moyens de réception (non représentés), configurés pour recevoir le fichier de mise à jour ΔFW, avantageusement compressé ΔFW_cpr, tel que décrit ci-avant ; et des moyens de traitement configurés pour remplacer, dans la partie de la mémoire contenant le fichier binaire de la version antérieure FW_vN, les bits des régions mémoires présentant des différences par les bits des régions mémoires correspondantes du fichier binaire de la version ultérieure FW_vN+1.

[0097] Ainsi, l'écriture de la version ultérieure du logiciel FW_vN+1, à partir du fichier de mise à jour ΔFW, ΔFW_cpr, peut être directement faite dans la mémoire en lieu et

place de la version antérieure FW_vN. En d'autres termes, il n'est pas nécessaire de prévoir un espace mémoire au moins aussi grand que la taille du logiciel entier pour stocker le fichier de mise à jour contenant le logiciel entier, avant d'y accéder en exécution.

[0098] Les moyens de traitement sont capables de décompresser le fichier de mise à jour compressé ΔFW_cpr, c'est-à-dire par exemple configurés pour lire ladite identification DSCR et accéder sélectivement auxdits emplacements mémoire identifiés.

[0099] Les moyens de traitement peuvent également être configurés à cet égard pour « reconstruire » le fichier des résultats de la comparaison ΔFW, en remplissant de zéros « 0 » (c'est à dire le terme neutre de l'opérateur du remplacement des bits en mémoire, l'opérateur ou-exclusif dans l'exemple ci-dessous) les régions mémoires qui ne sont pas identifiées dans l'en-tête DSCR.

[0100] Dans le cas où ladite construction du fichier de mise à jour ΔFW, ΔFW_cpr est faite avec les résultats de la comparaison bit-à-bit par l'opérateur ou-exclusif xor, alors la mise à jour du logiciel peut avantageusement tirer parti de la propriété de l'opérateur xor selon laquelle « A xor B = C <=> B = C xor A », c'est-à-dire la propriété selon laquelle l'opérateur xor est son propre inverse.

[0101] En effet, étant donné que le fichier de mise à jour ΔFW a été généré par : FW_vN xor FW_vN+1 = ΔFW, alors la version ultérieure FW_vN+1 est obtenue, à partir de la version antérieure FW_vN et du fichier de mise à jour ΔFW, par : FW_vN+1 = FW_vN xor ΔFW.

[0102] Plus particulièrement, dans l'exemple décrit ci-avant en relation avec la [Fig.6] :

DIFF_1 xor MWfunc2_vN = Call_#200 (pour le premier emplacement mémoire #100) ; et

DIFF_2 xor 0…0 = MWfunc2_vN+1 (pour le nouvel emplacement mémoire #200).

[0103] La [Fig.8] illustre un autre avantage de la propriété de l'opérateur ou-exclusif xor selon laquelle il est son propre inverse (c'est-à-dire « A xor B xor B = A »), dans un cas où la version ultérieure correspondant à un retour à une version précédente FW_vN, en place avant la dernière mise à jour FW_vN+1 effectuée.

[0104] Ainsi, dans ce cas du procédé de mise à jour d'une version antérieure d'un logiciel vers une version ultérieure du logiciel, la version antérieure « FW_vN+1 » a été mise à jour à partir d'une version précédente « FW_vN » avec le fichier de mise à jour antérieur ΔFW_cpr (vN->vN+1), par exemple telle que décrit précédemment en relation avec la [Fig.7], et la version ultérieure correspondant à un retour à ladite version précédente « FW_vN ».

[0105] Le fichier de mise à jour antérieur ΔFW_cpr (vN->vN+1), permet, grâce aux propriétés l'opérateur ou-exclusif xor, de revenir directement à ladite version précédente FW_vN, en appliquant le mécanisme de remplacement décrit ci-avant sans

distinction.

[0106] En détails, FW_vN+1 xor ΔFW = FW_vN ; et
DIFF_1 xor Call_#200= MWfunc2_vN ; et
DIFF_2 xor MWfunc2_vN+1 = 0…0.

[0107] La [Fig.9] illustre un autre avantage de la propriété de l'opérateur ou-exclusif, permettant, à partir d'un fichier de mise à jour d'une version FW_v1 vers une version FW_v2, et d'un fichier de mise à jour d'une version FW_v2 vers une version FW_v3, de créer immédiatement le fichier de mise à jour pour la version FW_v1 vers la version FW_v3.

[0108] En effet, par construction : ΔFW(v1->v3) = ΔFW(v1->v2) xor ΔFW(v2->v3).

[0109] Cela permet en pratique de concevoir, par le dispositif serveur, des fichiers de mise à jour permettant de rattraper des ratés pour certains dispositifs clients dans une campagne de mise à jour passée, d'une manière bénéficiant à la fois d'une grande simplicité et d'être adaptés aux cas particuliers.

[0110] La [Fig.10] illustre schématiquement un système comportant un dispositif serveur SRV et au moins un dispositif client DEV, aptes à communiquer ensemble via un réseau de communication NTW.

[0111] Le dispositif serveur SRV est apte à générer le fichier de mise à jour ΔFW_cpr de la version antérieure d'un logiciel FW_vN vers la version ultérieure du logiciel FW_vN+1, selon le procédé décrit précédemment en relation avec les figures 2 à 6, et est apte à communiquer ledit fichier de mise à jour ΔFW_cpr aux dispositif clients par l'intermédiaire du réseau de communication NTW.

[0112] Les dispositifs clients DEV sont aptes à mettre à jour la version antérieure d'un logiciel FW_vN contenu dans leurs mémoires, vers la version ultérieure du logiciel FW_vN+1 avec le fichier de mise à jour ΔFW_cpr ainsi communiqué.

[0113] Par exemple, le réseau de communication NTW peut être avantageusement un réseau relativement limité en bande passante et débit de données, par exemple les réseaux du type « LPWAN » (pour « Low Power Wide Area Network » en anglais) tel que « LoRaWAN », « BLE », « Cat-M1 », « NB-IoT » (respectivement pour « Long Range WAN », « Bluetooth Low Energy », « Machine Type Communication, Category 1 » et « Narrow Band-IoT » en anglais).

[0114] Cela étant, le réseau de communication NTW peut aussi ne pas être particulière limité en matière de bande passante et débit de données, par exemple les réseaux du type « Wi-Fi » (définit par la norme IEEE 802.11), du type « LTE 4G/5G » (pour « Long Term Evolution, 4th/5th generation ») ou du type connexion filaire tel qu'Ethernet ou USB.

Revendications

[Revendication 1] Procédé de génération d'un fichier de mise à jour d'une version antérieure d'un logiciel (FW_vN) vers une version ultérieure du logiciel (FW_vN+1), comprenant :
- une compilation (CMPLR) d'un code source (vN+1_code) générant un fichier binaire (vN+1_bin) de la version ultérieure (FW_vN+1) ;
- une édition de liens (LNKR) attribuant des emplacements mémoires d'une mémoire à des sections du fichier binaire (vN+1_bin) de la version ultérieure (FW_vN+1), les emplacements mémoires desdites sections de la version ultérieure (FW_vN+1) étant contraints à l'identique (vN_ScttrF) des emplacements mémoires des sections correspondantes de la version antérieure (FW_vN) ;
- une comparaison, faite bit-à-bit par un opérateur ou-exclusif (xor), entre le fichier binaire (vN+1_bin) de la version ultérieure (FW_vN+1) et un fichier binaire (vN_bin) de la version antérieure (FW_vN) ; et
- une construction du fichier de mise à jour (ΔFW, ΔFW_cpr) comportant les résultats de la comparaison bit-à-bit par l'opérateur ou-exclusif (xor), pour communiquer les différences (DAT_diff), par régions mémoires (Offst_1-Lgth_1, …, Offst_k-Lgth_k), entre lesdits fichiers binaires comparés (vN+1_bin, vN_bin).

[Revendication 2] Procédé selon la revendication 1, dans lequel le fichier de mise à jour (ΔFW_cpr) est exempt des régions mémoires exemptes de différences (0, …, 0) entre lesdits fichiers binaires comparés (vN+1_bin, vN_bin), et dans lequel le fichier de mise à jour (ΔFW_cpr) comporte une identification (DSCR) des emplacements mémoires, dans le fichier binaire (vN_bin, vN+1_bin) de la version antérieure (FW_vN), des régions mémoires présentant lesdites différences (DIFF_1, …, DIFF_k) entre lesdits fichiers binaires comparés (vN+1_bin, vN_bin).

[Revendication 3] Procédé selon l'une des revendications 1 ou 2, dans lequel, si le fichier binaire (vN+1_bin) de la version ultérieure (FW_vN+1) comporte une section (MWfunc2_vN+1) de plus grande taille en mémoire (sz_200) que l'emplacement mémoire (#100 ; sz_100) attribué à cette section (MWfunc2_vN) dans la version antérieure (FW_vN), alors l'édition de liens (LNKR) attribue un nouvel emplacement mémoire (#200 ; sz_200), libre dans la version antérieure (FW_vN), à ladite section plus grande (MWfunc2_vN+1), et introduit une instruction d'appel (Call_#200) vers le nouvel emplacement mémoire (#200), dans

l'emplacement mémoire (#100 ; sz_100) attribué à cette section (MWfunc2_vN) dans la version antérieure.

[Revendication 4] Procédé selon l'une des revendications 1 à 3, dans lequel, si le fichier binaire (vN+1_bin) de la version ultérieure (FW_vN+1) comporte une section (MWfunc2_vN+1) de plus petite taille en mémoire que l'emplacement mémoire (#100 ; sz_100) attribué à cette section (MWfunc2_vN) dans la version antérieure, alors l'édition de liens (LNKR) attribue le même emplacement mémoire ayant cette même taille (#100 ; sz_100) et laisse vide l'excédent en taille de l'emplacement mémoire (#100 ; sz_100).

[Revendication 5] Procédé de mise à jour d'une version antérieure d'un logiciel (FW_vN) contenue dans une mémoire vers une version ultérieure du logiciel (FW_vN+1), comprenant :

- une réception d'un fichier de mise à jour (ΔFW, ΔFW_cpr) comportant le résultat d'une comparaison bit-à-bit par un opérateur ou-exclusif (xor) entre le fichier binaire de la version ultérieure (vN+1_bin) et le fichier binaire de la version antérieure (vN_bin), pour communiquer les différences (DAT_diff), par régions mémoires (Offst_1-Lgth_1, …, Offst_k-Lgth_k), entre un fichier binaire (vN+1_bin) de la version ultérieure (FW_vN+1) et un fichier binaire (vN_bin) de la version antérieure (FW_vN) ;
- un remplacement, dans la partie de la mémoire contenant le fichier binaire (vN_bin) de la version antérieure (FW_vN), des bits des régions mémoires présentant des différences par les bits des régions mémoires du fichier binaire (vN+1_bin) de la version ultérieure (FW_vN+1), le remplacement comprenant une transformation des bits contenus dans lesdites régions mémoires du fichier binaire de la version antérieure (vN_bin) par l'opérateur ou-exclusif (xor) avec les bits dudit résultat de la comparaison bit-à-bit par l'opérateur ou-exclusif (xor) entre lesdits fichiers binaires (vN+1_bin, vN_bin).

[Revendication 6] Procédé selon la revendication 5, dans lequel :

- ledit fichier de mise à jour (ΔFW_cpr) est exempt des régions mémoires exemptes de différences (0, …, 0) entre lesdits fichiers binaires comparés (vN+1_bin, vN_bin), et comporte une identification (DSCR) des emplacements mémoires, dans le fichier binaire (vN_bin, vN+1_bin) de la version antérieure (FW_vN), des régions mémoires présentant lesdites différences (DIFF_1, …, DIFF_k) entre lesdits fichiers binaires comparés (vN+1_bin, vN_bin) ;

- ledit remplacement comprend une lecture de ladite identification (DSCR) et un accès sélectif auxdits emplacements mémoire identifiés.

[Revendication 7] Procédé, selon l'une des revendications 5 ou 6, de mise à jour d'une version antérieure d'un logiciel (FW_vN+1) vers une version ultérieure du logiciel (FW_vN), la version antérieure (FW_vN+1) ayant été mise à jour à partir d'une version précédente (FW_vN) avec le fichier de mise à jour antérieur (ΔFW, ΔFW_cpr), et la version ultérieure (FW_vN) correspondant à un retour à ladite version précédente (FW_vN), dans lequel :
- le fichier de mise à jour (ΔFW, ΔFW_cpr) est le fichier mise à jour antérieur ;
- ledit remplacement comprend une transformation des bits contenus dans lesdites régions mémoires du fichier binaire de la version antérieure par l'opérateur ou-exclusif (xor) avec les bits desdites régions du résultat de la comparaison du fichier de mise à jour antérieur.

[Revendication 8] Méthode de mise à jour d'une version antérieure d'un logiciel (FW_vN) vers une version ultérieure du logiciel (FW_vN+1), pour au moins un dispositif, comprenant :
- un procédé de génération d'un fichier de mise à jour (ΔFW, ΔFW_cpr) de la version antérieure d'un logiciel (FW_vN) vers la version ultérieure du logiciel (FW_vN+1) selon l'une des revendications 1 à 4, par un dispositif serveur (SRV) ;
- une communication audit au moins un dispositif du fichier de mise à jour (ΔFW, ΔFW_cpr), par un réseau de communication (NTW) ;
- un procédé de mise à jour de version antérieure d'un logiciel (FW_vN) vers la version ultérieure du logiciel (FW_vN+1) selon l'une des revendications 5 à 7, par ledit au moins un dispositif client (DEV).

[Revendication 9] Dispositif serveur, apte à générer un fichier de mise à jour d'une version antérieure d'un logiciel (FW_vN) vers une version ultérieure du logiciel (FW_vN+1), comprenant :
- un compilateur (CMPLR) configuré pour compiler un code source (vN+1_code) de manière à générer un fichier binaire (vN+1_bin) de la version ultérieure (FW_vN+1) ;
- un éditeur de lien (LNKR) configuré pour attribuer des emplacements mémoires d'une mémoire à des sections du fichier binaire (vN+1_bin) de la version ultérieure (FW_vN+1), de sorte que les emplacements mémoires desdites sections de la version ultérieure (FW_vN+1) sont contraints à l'identique (vN_ScttrF) des emplacements mémoires des

sections correspondantes de la version antérieure (FW_vN) ;
- un compresseur (CPRSSR) configuré pour :
-- comparer bit-à-bit par un opérateur ou-exclusif (xor), le fichier binaire (vN+1_bin) de la version ultérieure (FW_vN+1) avec un fichier binaire (vN_bin) de la version antérieure (FW_vN) ; et pour
-- construire le fichier de mise à jour (ΔFW, ΔFW_cpr) comportant les résultats de la comparaison bit-à-bit par l'opérateur ou-exclusif (xor), pour communiquer les différences (DAT_diff), par régions mémoires, entre lesdits fichiers binaires comparés (vN+1_bin, vN_bin).

[Revendication 10] Dispositif serveur selon la revendication 9, dans lequel le compresseur est configuré pour construire le fichier de mise à jour (ΔFW_cpr) exempt des régions mémoires exemptes de différences (0, …, 0) entre lesdits fichiers binaires comparés (vN+1_bin, vN_bin), et pour construire le fichier de mise à jour (ΔFW_cpr) comportant une identification (DSCR) des emplacements mémoires, dans le fichier binaire (vN_bin, vN+1_bin) de la version antérieure (FW_vN), des régions mémoires présentant lesdites différences (DIFF_1, …, DIFF_k) entre lesdits fichiers binaires comparés (vN+1_bin, vN_bin).

[Revendication 11] Dispositif serveur selon l'une des revendications 9 ou 10, dans lequel l'éditeur de lien (LNKR) est configuré, si le fichier binaire (vN+1_bin) de la version ultérieure (FW_vN+1) comporte une section (MWfunc2_vN+1) de plus grande taille en mémoire (sz_200) que l'emplacement mémoire (#100 ; sz_100) attribué à cette section (MWfunc2_vN) dans la version antérieure (FW_vN), pour attribuer un nouvel emplacement mémoire (#200 ; sz_200), libre dans la version antérieure (FW_vN), à ladite section plus grande (MWfunc2_vN+1), et introduire une instruction d'appel (Call_#200) vers le nouvel emplacement mémoire (#200), dans l'emplacement mémoire (#100 ; sz_100) attribué à cette section (MWfunc2_vN) dans la version antérieure.

[Revendication 12] Dispositif serveur selon l'une des revendications 9 à 11, dans lequel l'éditeur de lien (LNKR) est configuré, si le fichier binaire (vN+1_bin) de la version ultérieure (FW_vN+1) comporte une section (MWfunc2_vN+1) de plus petite taille en mémoire que l'emplacement mémoire (#100 ; sz_100) attribué à cette section (MWfunc2_vN) dans la version antérieure, pour attribuer le même emplacement mémoire ayant cette même taille (#100 ; sz_100) et laisser vide l'excédent en taille de l'emplacement mémoire (#100 ; sz_100).

[Revendication 13] Dispositif client apte à mettre à jour une version antérieure d'un logiciel (FW_vN) contenue dans une mémoire, vers une version ultérieure du logiciel (FW_vN+1), comprenant :
- des moyens de réception configurés pour recevoir un fichier de mise à jour (ΔFW, ΔFW_cpr) comportant comportant le résultat d'une comparaison bit-à-bit par un opérateur ou-exclusif (xor) entre le fichier binaire de la version ultérieure (vN+1_bin) et le fichier binaire de la version antérieure (vN_bin), pour communiquer les différences (DAT_diff), par régions mémoires (Offst_1-Lgth_1, …, Offst_k-Lgth_k), entre un fichier binaire (vN+1_bin) de la version ultérieure (FW_vN+1) et un fichier binaire (vN_bin) de la version antérieure (FW_vN) ;
- des moyens de traitement configurés pour remplacer, dans la partie de la mémoire contenant le fichier binaire (vN_bin) de la version antérieure (FW_vN), des bits des régions mémoires présentant des différences par les bits des régions mémoires du fichier binaire (vN+1_bin) de la version ultérieure (FW_vN+1), en transformant les bits contenus dans lesdites régions mémoires du fichier binaire de la version antérieure (vN_bin) par l'opérateur ou-exclusif (xor) avec les bits dudit résultat de la comparaison bit-à-bit par l'opérateur ou-exclusif (xor) entre lesdits fichiers binaires (vN+1_bin, vN_bin).

[Revendication 14] Dispositif client selon la revendication 13, dans lequel :
- les moyens de réception sont configurés pour recevoir le fichier de mise à jour (ΔFW_cpr) exempt des régions mémoires exemptes de différences (0, …, 0) entre lesdits fichiers binaires comparés (vN+1_bin, vN_bin), et comportant une identification (DSCR) des emplacements mémoires, dans le fichier binaire (vN_bin, vN+1_bin) de la version antérieure (FW_vN), des régions mémoires présentant lesdites différences (DIFF_1, …, DIFF_k) entre lesdits fichiers binaires comparés (vN+1_bin, vN_bin) ;
- les moyens de traitement sont configurés pour lire ladite identification (DSCR) et accéder sélectivement auxdits emplacements mémoire identifiés.

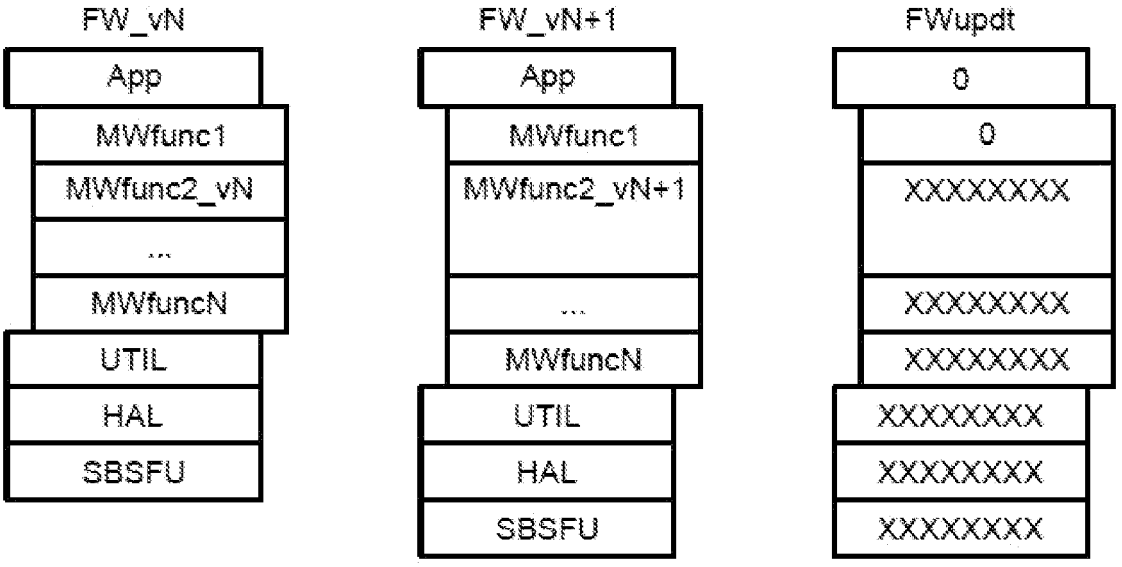
[Revendication 15] Dispositif client, selon l'une des revendications 13 ou 14, apte à mettre à jour une version antérieure d'un logiciel (FW_vN+1) vers une version ultérieure du logiciel (FW_vN), la version antérieure (FW_vN+1) ayant été mise à jour à partir d'une version précédente (FW_vN) avec le fichier de mise à jour antérieur (ΔFW, ΔFW_cpr), et la version ul-

térieure (FW_vN) correspondant à un retour à ladite version précédente (FW_vN), dans lequel :
- les moyens de réception sont configurés pour récupérer le fichier mise à jour antérieur en tant que fichier de mise à jour (ΔFW, ΔFW_cpr) ;
- les moyens de traitement sont configurés pour transformer les bits contenus dans lesdites régions mémoires du fichier binaire de la version antérieure (vN_bin) par l'opérateur ou-exclusif (xor) avec les bits dudit résultat du fichier de mise à jour antérieur.

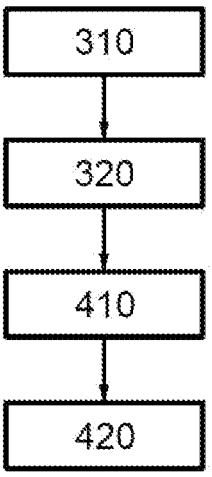
[Revendication 16] Système comportant un dispositif serveur (SRV) selon l'une des revendications 9 à 12, et au moins un dispositif client (DEV) selon l'une des revendications 13 à 15, le dispositif serveur étant apte à communiquer ledit fichier de mise à jour (ΔFW_cpr) de la version antérieure d'un logiciel (FW_vN) vers le version ultérieure du logiciel (FW_vN+1), audit au moins un dispositif client (DEV) par un réseau de communication (NTW), et le dispositif client (DEV) étant apte à mettre à jour la version antérieure d'un logiciel (FW_vN) vers la version ultérieure du logiciel (FW_vN+1) avec ce fichier de mise à jour (ΔFW_cpr).

[Fig. 1]

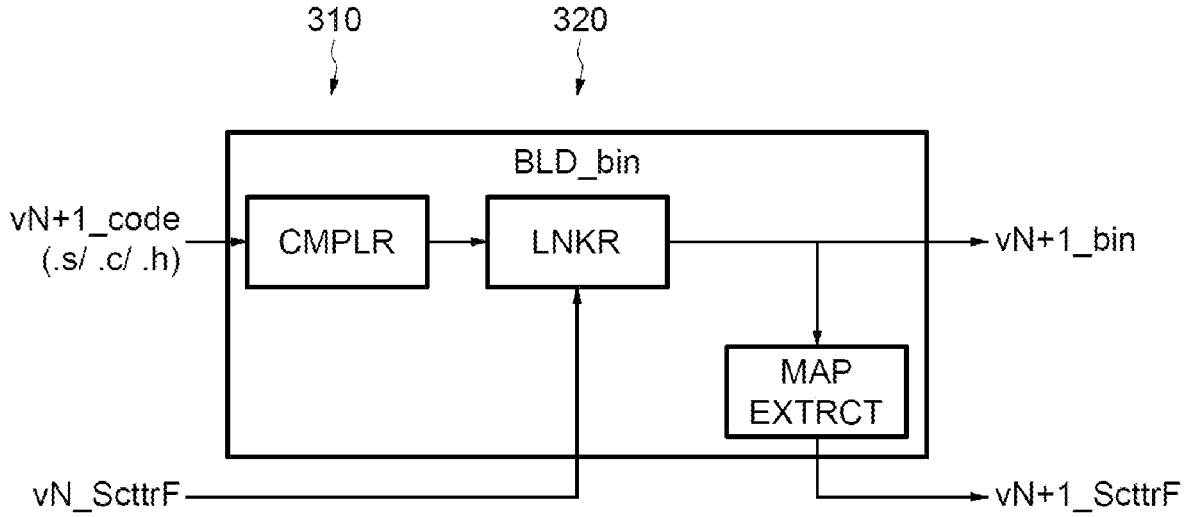
FIG.1
ART ANTERIEUR



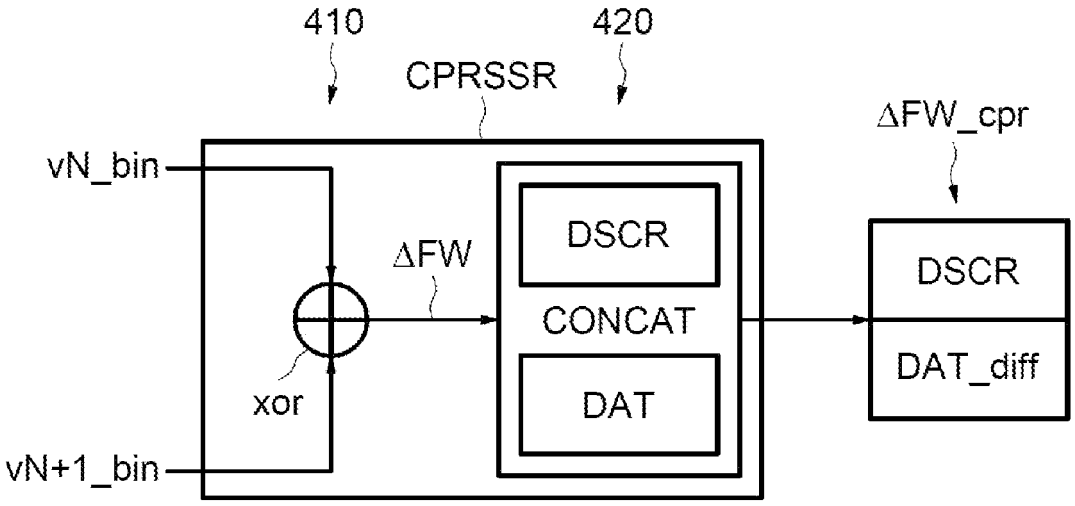
[Fig. 2]



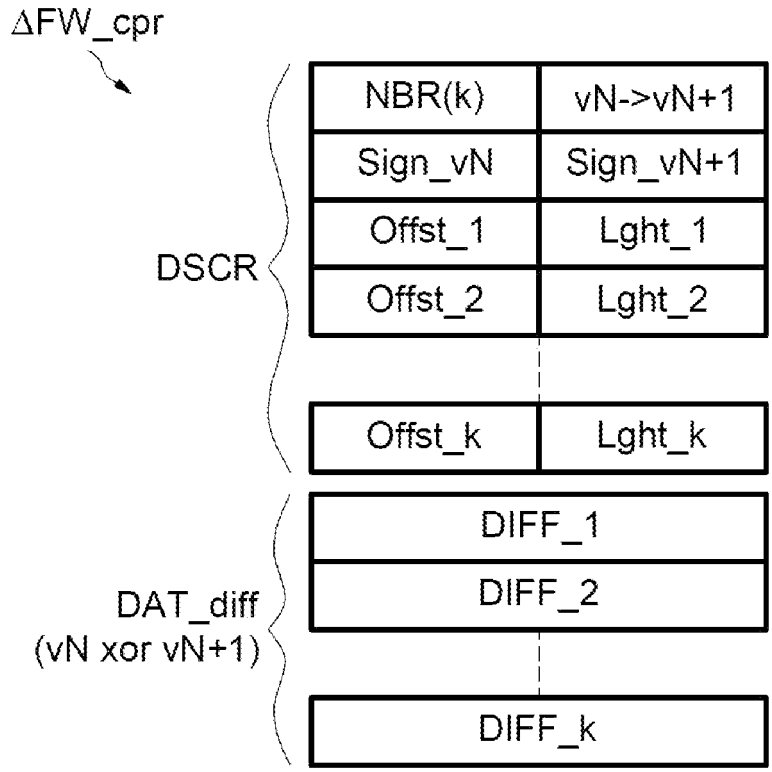
[Fig. 3]


310
320
BLD_bin
vN+1_code
(.s/ .c/ .h)
CMPLR
LNKR
vN+1_bin
MAP
EXTRCT
vN_ScttrF
vN+1_ScttrF

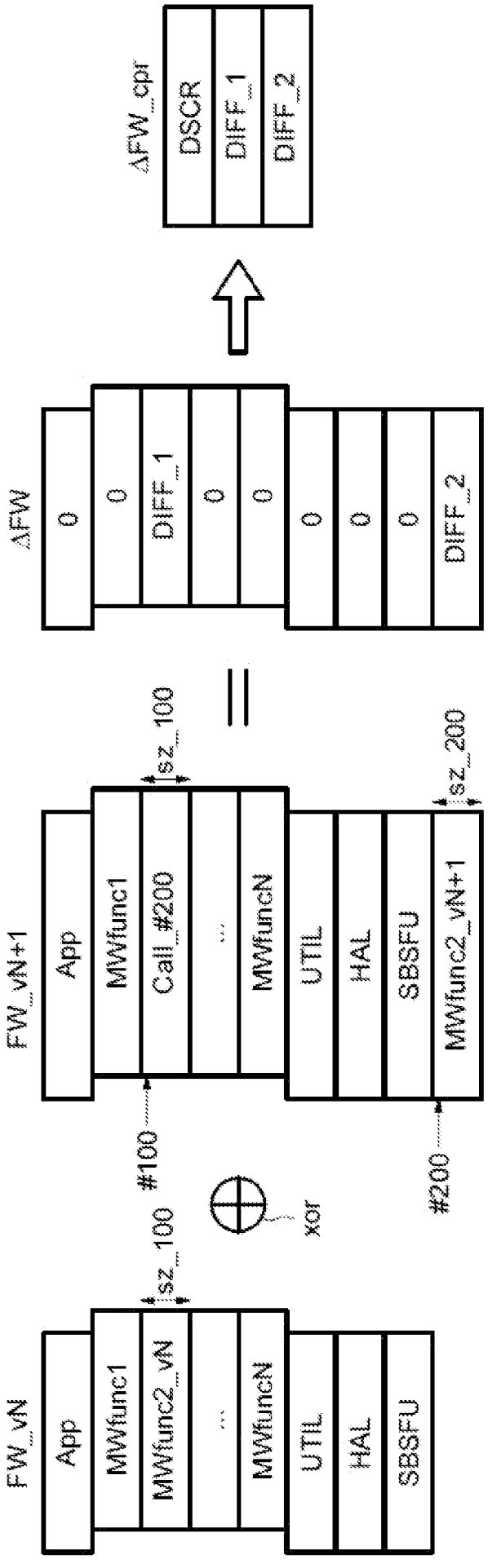
[Fig. 4]


410
420
CPRSSR
ΔFW_cpr
vN_bin
ΔFW
DSCR
CONCAT
DAT
DSCR
DAT_diff
xor
vN+1_bin

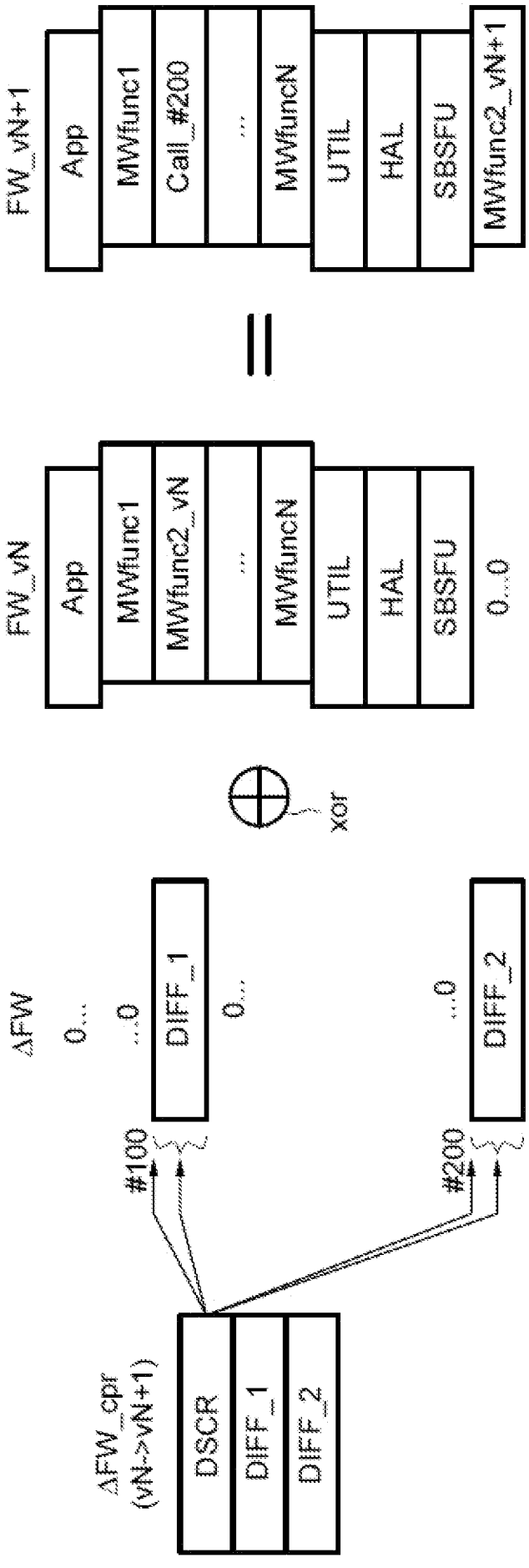
[Fig. 5]


ΔFW_cpr
DSCR
NBR(k)
vN->vN+1
Sign_vN
Sign_vN+1
Offst_1
Lght_1
Offst_2
Lght_2
Offst_k
Lght_k
DAT_diff
(vN xor vN+1)
DIFF_1
DIFF_2
DIFF_k

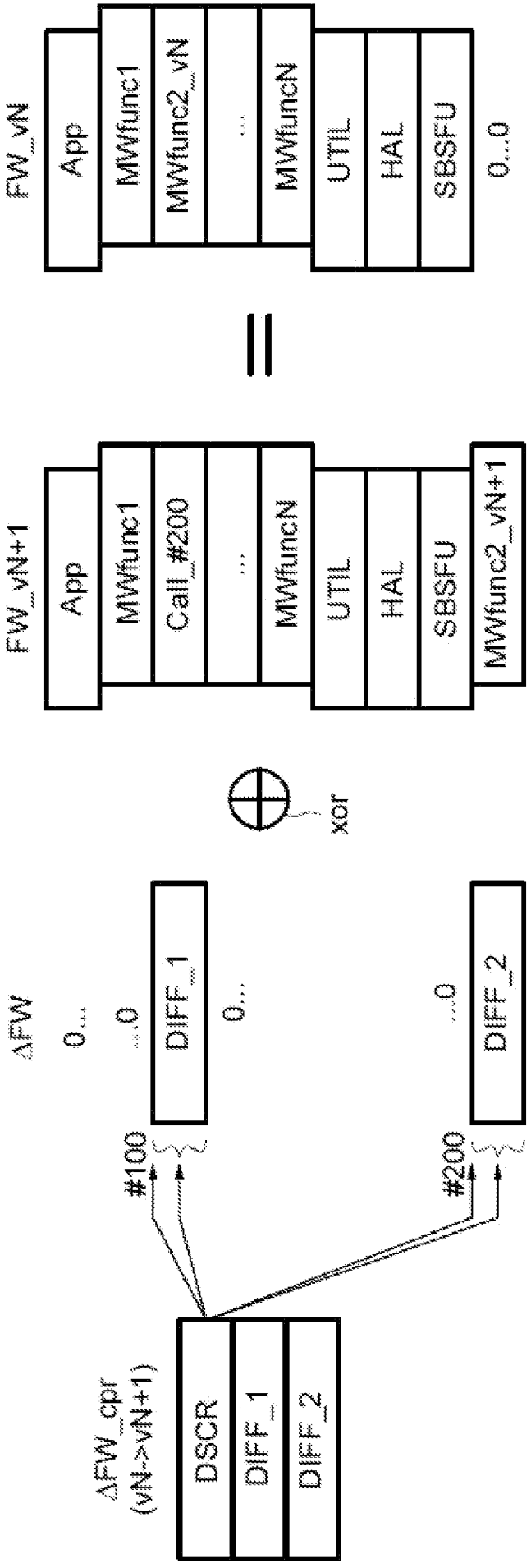
[Fig. 6]


FW_vN
App
MWfunc1
MWfunc2_vN
...
MWfuncN
UTIL
HAL
SBSFU
sz_100
#100
xor
FW_vN+1
App
MWfunc1
Call_#200
...
MWfuncN
UTIL
HAL
SBSFU
MWfunc2_vN+1
sz_100
sz_200
#200
ΔFW
0
0
DIFF_1
0
0
0
0
0
DIFF_2
ΔFW_cpr
DSCR
DIFF_1
DIFF_2

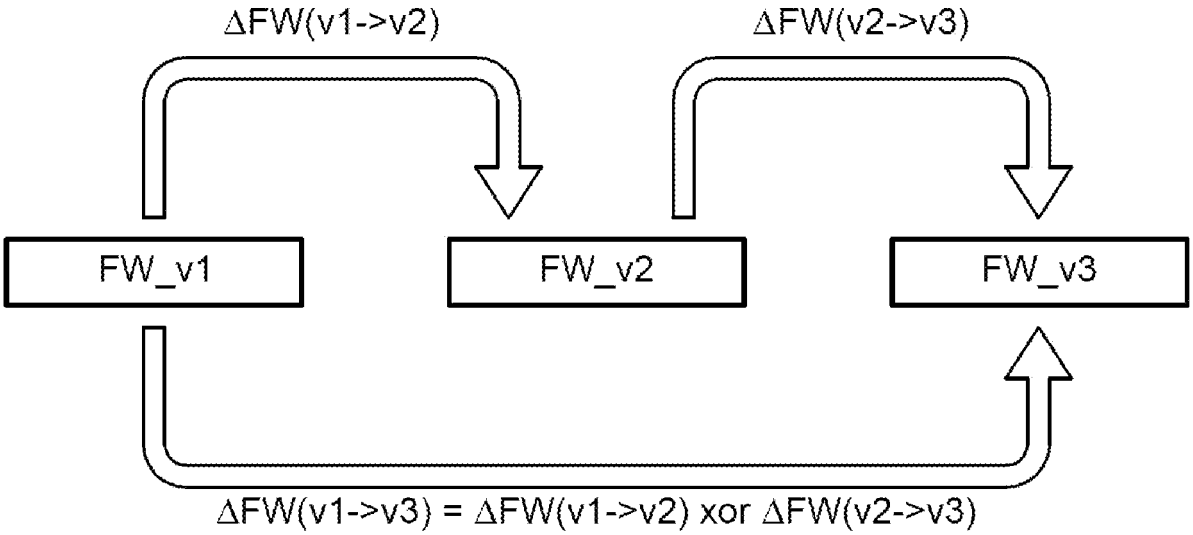
[Fig. 7]


FW_vN+1
App
MWfunc1
Call_#200
...
MWfuncN
UTIL
HAL
SBSFU
MWfunc2_vN+1
=
FW_vN
App
MWfunc1
MWfunc2_vN
...
MWfuncN
UTIL
HAL
SBSFU
0...0
xor
ΔFW
0...
...0
DIFF_1
0...
...0
DIFF_2
#100
#200
ΔFW_cpr
(vN->vN+1)
DSCR
DIFF_1
DIFF_2

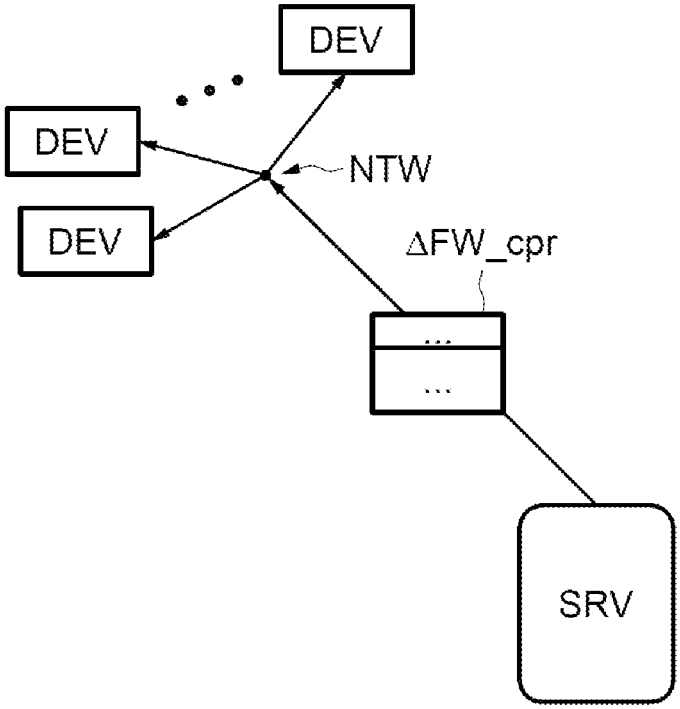
[Fig. 8]


FW_vN
App
MWfunc1
MWfunc2_vN
...
MWfuncN
UTIL
HAL
SBSFU
0...0
=
FW_vN+1
App
MWfunc1
Call_#200
...
MWfuncN
UTIL
HAL
SBSFU
MWfunc2_vN+1
xor
ΔFW
0...
...0
DIFF_1
0...
...0
DIFF_2
#100
#200
ΔFW_cpr
(vN->vN+1)
DSCR
DIFF_1
DIFF_2

[Fig. 9]


ΔFW(v1->v2)
ΔFW(v2->v3)
FW_v1
FW_v2
FW_v3
ΔFW(v1->v3) = ΔFW(v1->v2) xor ΔFW(v2->v3)

[Fig. 10]


DEV
DEV
DEV
NTW
ΔFW_cpr
...
...
SRV

RAPPORT DE RECHERCHE

articles L.612-14, L.612-53 à 69 du code de la propriété intellectuelle

OBJET DU RAPPORT DE RECHERCHE

L'I.N.P.I. annexe à chaque brevet un "RAPPORT DE RECHERCHE" citant les éléments de l'état de la technique qui peuvent être pris en considération pour apprécier la brevetabilité de l'invention, au sens des articles L. 611-11 (nouveauté) et L. 611-14 (activité inventive) du code de la propriété intellectuelle. Ce rapport porte sur les revendications du brevet qui définissent l'objet de l'invention et délimitent l'étendue de la protection.
Après délivrance, l'I.N.P.I. peut, à la requête de toute personne intéressée, formuler un "AVIS DOCUMENTAIRE" sur la base des documents cités dans ce rapport de recherche et de tout autre document que le requérant souhaite voir prendre en considération.

CONDITIONS D'ETABLISSEMENT DU PRESENT RAPPORT DE RECHERCHE

[x] Le demandeur a présenté des observations en réponse au rapport de recherche préliminaire.

☐ Le demandeur a maintenu les revendications.

[x] Le demandeur a modifié les revendications.

☐ Le demandeur a modifié la description pour en éliminer les éléments qui n'étaient plus en concordance avec les nouvelles revendications.

☐ Les tiers ont présenté des observations après publication du rapport de recherche préliminaire.

☐ Un rapport de recherche préliminaire complémentaire a été établi.

DOCUMENTS CITES DANS LE PRESENT RAPPORT DE RECHERCHE

La répartition des documents entre les rubriques 1, 2 et 3 tient compte, le cas échéant, des revendications déposées en dernier lieu et/ou des observations présentées.

☐ Les documents énumérés à la rubrique 1 ci-après sont susceptibles d'être pris en considération pour apprécier la brevetabilité de l'invention.

[x] Les documents énumérés à la rubrique 2 ci-après illustrent l'arrière-plan technologique général.

☐ Les documents énumérés à la rubrique 3 ci-après ont été cités en cours de procédure, mais leur pertinence dépend de la validité des priorités revendiquées.

☐ Aucun document n'a été cité en cours de procédure.

1. ELEMENTS DE L'ETAT DE LA TECHNIQUE SUSCEPTIBLES D'ETRE PRIS EN CONSIDERATION POUR APPRECIER LA BREVETABILITE DE L'INVENTION

NEANT

2. ELEMENTS DE L'ETAT DE LA TECHNIQUE ILLUSTRANT L'ARRIERE-PLAN TECHNOLOGIQUE GENERAL

EP 1 755 039 A1 (ERICSSON TELEFON AB L M [SE]) 21 février 2007 (2007-02-21)

EP 0 472 812 A1 (LANDIS & GYR BETRIEBS AG [CH]) 4 mars 1992 (1992-03-04)

EP 1 956 482 A1 (ERICSSON TELEFON AB L M [SE]) 13 août 2008 (2008-08-13)

EP 1 569 102 B1 (ERICSSON TELEFON AB L M [SE]) 28 avril 2010 (2010-04-28)

US 9 268 552 B1 (KIISKILA MARKO [US] ET AL) 23 février 2016 (2016-02-23)

CARL VON PLATEN ET AL: "Feedback linking",
ACM SIGPLAN NOTICES, ASSOCIATION FOR COMPUTING MACHINERY, US,
vol. 41, no. 7, 14 juin 2006 (2006-06-14),
pages 2-11, XP058195660,
ISSN: 0362-1340, DOI:
10.1145/1159974.1134653

3. ELEMENTS DE L'ETAT DE LA TECHNIQUE DONT LA PERTINENCE DEPEND DE LA VALIDITE DES PRIORITES

NEANT