(19) RÉPUBLIQUE FRANÇAISE

INSTITUT NATIONAL DE LA PROPRIÉTÉ INDUSTRIELLE

PARIS

(11) **N° de publication :** **3 006 527**

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

(21) **N° d'enregistrement national :** **13 54889**

(51) Int Cl[8] : ***H 04 L 12/24*** *(2013.01), H 04 L 12/16*

(12) DEMANDE DE BREVET D'INVENTION A1

(22) **Date de dépôt :** 29.05.13.

(30) **Priorité :**

(43) **Date de mise à la disposition du public de la demande :** 05.12.14 Bulletin 14/49.

(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) :** *UNIVERSITE DE PAU ET DES PAYS DE L'ADOUR Etablissement public* — FR.

(72) **Inventeur(s) :** DALMAU MARC et ROOSE PHILIPPE.

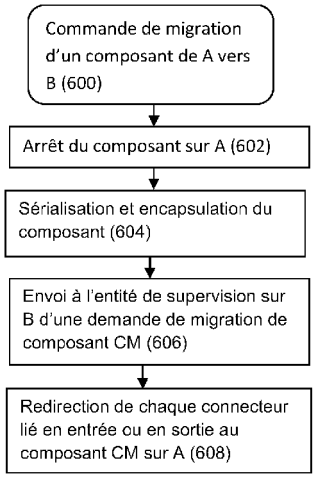
(73) **Titulaire(s) :** UNIVERSITE DE PAU ET DES PAYS DE L'ADOUR Etablissement public.

(74) **Mandataire(s) :** MARKS & CLERK FRANCE Société en nom collectif.

(54) MIGRATION DE COMPOSANTS APPLICATIFS.

(57) L'invention propose un système de supervision d'applications s'exécutant sur un ensemble de dispositifs électroniques reliés entre eux par un ou plusieurs réseaux. Chaque dispositif comprend une entité de supervision locale, les entités de supervision coopérant entre elles pour contrôler les applications s'exécutant sur les dispositifs électroniques. Chaque application comprend un ensemble de composants applicatifs, et chaque composant applicatif est encapsulé dans un conteneur par l'entité de supervision du dispositif hébergeant le composant. Les composants sont reliés entre eux par des connecteurs. L'entité de supervision d'un dispositif donné, dit dispositif source, est configurée pour exécuter les étapes suivantes, en réponse à la réception d'une commande de migration d'un composant vers un dispositif cible :

- arrêter le composant (602), l'arrêt du composant interrompant l'arrivée de données dans les connecteurs d'entrée du composant,
-sérialiser et encapsuler les propriétés du composant dans un conteneur de l'entité de supervision (604),
-envoyer un message de demande migration à l'entité de supervision du dispositif cible, ledit message comprenant le composant sérialisé et encapsulé (606), et
- rediriger les connecteurs (608) du composant en fonction de l'état des connexions de chaque connecteur sur le dispositif source.



FR 3 006 527 - A1

Migration de composants applicatifs

Domaine technique

La présente invention concerne généralement les dispositifs électroniques mobiles et en particulier un procédé et un système de supervision de composants applicatifs.

Art antérieur et problème technique

Les récentes avancées technologiques de ces dernières années ont mis l'accent sur la démocratisation des réseaux sans-fil et sur la miniaturisation des appareils de communication. Actuellement, il existe sur le marché une multitude de dispositifs électroniques personnels de plus en plus légers, compacts, mobiles et dotés de divers moyens de communication sans-fil, tels que les téléphones portables, les téléphones mobiles intelligents (« smartphones » en langue anglo-saxonne), les tablettes, les ordinateurs portables ou encore les capteurs. Ces dispositifs concentrent des fonctionnalités complexes et très diversifiées : téléphonie, messagerie instantanée, navigation Internet, système de localisation (GPS acronyme pour « Global Positioning System »), lecteur audio, etc.

Ces dispositifs font l'objet d'une demande grandissante pour des services de plus en plus riches et personnalisés. Le défi est de pouvoir proposer des applications pour ces dispositifs qui s'adaptent tant aux souhaits des utilisateurs qu'à l'environnement physique dans lequel ils opèrent.

Les dispositifs électroniques mobiles ont la capacité de pouvoir rendre compte non seulement de leur environnement matériel et logiciel mais aussi, avec l'arrivée de périphériques tels que les capteurs sans fils ou les capteurs intégrés aux téléphones portables, de pouvoir mesurer des grandeurs physiques liées à l'environnement du dispositif ou au dispositif lui-même, telles que la température, la pression ou encore la vitesse de déplacement. L'intégration des données issues de tels dispositifs dans les applications peut permettre de proposer aux utilisateurs des services mieux adaptés à leur situation courante. Cependant, ces dispositifs possèdent des caractéristiques (autonomie énergétique, mobilité, ressources limitées) qui nécessitent l'adaptation des applications ainsi que des services rendus par celles-ci pour assurer un fonctionnement correct pendant une durée suffisante, et une continuité de service en cas d'indisponibilité d'un périphérique du dispositif électronique, par exemple lorsqu'un périphérique du dispositif électronique ne possède plus suffisamment de batterie. En l'absence de continuité de service, tous les services offerts par le périphérique cessent alors de fonctionner impliquant une rupture de service. Il peut en résulter un confort d'utilisation limité pour l'utilisateur, notamment dans le cas où des applications sont en cours d'exécution.

On connaît des systèmes de supervision sensibles au contexte qui réagissent aux changements de contexte pour fournir à l'utilisateur des services adaptés à la situation. De tels systèmes

peuvent reposer sur différents types d'adaptations : adaptation de contenu, adaptation de présentation, adaptation de comportement, adaptation structurelle, adaptation de déploiement.

Ainsi, des systèmes de supervision existent pour adapter le contenu des applications qui s'exécutent sur les dispositifs électroniques en fonction du contexte, comme par exemple la solution décrite dans Christoph Dorn, Richard N. Taylor - Co-adapting human collaborations and software architectures - ICSE 2012 Proceedings of the 2012 International Conference on Software Engineering – pp. 1277-1280. En fonction de la situation, les données peuvent être modifiées pour ne présenter à l'utilisateur que celles qui sont pertinentes à sa situation.

D'autres systèmes connus sont configurés pour adapter la présentation dans le domaine des IHM (acronyme pour Interface Homme Machine). En fonction du statut hiérarchique de l'utilisateur, l'interface de l'application présentera ou non une information et proposera ou non une fonctionnalité, comme par exemple la solution décrite dans l'article de Y. Gabillon, G. Calvary, H. Fiorino, intulé « Composition d'Interfaces Homme-Machine en contexte : approche par planification automatique », Revue TSI. Vol. 30, 2011.

Dans d'autres systèmes encore, il est prévu une adaptation de comportement qui repose sur l'adaptation des fonctionnalités fournies par un composant ou un service en fonction du contexte, comme par exemple la solution décrite dans l'article de Peyman Oreizy, Nenad Medvidovic, Richard N. Taylor, intitulé « Runtime Software Adaptation: Framework, Approaches, and Styles », ICSE Companion '08 Companion of the 30th international conference on Software engineering, pp. 899-910. Dans d'autres approches encore, les systèmes de supervision réalisent une adaptation structurelle qui vise à modifier la composition de l'application et/ou les connexions entre les différents composants dans le but d'obtenir une application dont le comportement reste inchangé. Ce type d'adaptation est le plus couramment utilisé dans le domaine des applications distribuées basées composants.

On connaît également des systèmes de supervision qui adaptent le déploiement en fonction du contexte, comme proposé par exemple dans l'article de Ning Gui, Vincenzo De Florio, Hong Sun, Chris Blondia intitulé « Toward architecture-based context-aware deployment and adaptation », The Journal of Systems and Software 84 (2011) 185–197 – Elsevier, 2011. De tels systèmes mettent en œuvre des déploiements qui prennent en compte les propriétés des périphériques supportant l'application. Ce type d'adaptation est souvent utilisé pour faire face aux problèmes engendrés par les limitations matérielles des dispositifs mobiles et contraints, massivement utilisés de nos jours.

Les solutions existantes reposant sur l'adaptation de contenu, l'adaptation de présentation et l'adaptation de fonctionnalité sont essentiellement tournés vers l'utilisateur. Le contenu et les fonctionnalités sont adaptés en fonction de ses préférences et la présentation est adaptée en fonction de son statut par exemple. Les adaptations de structure et de déploiement conviennent

particulièrement aux contraintes matérielles et aux contraintes réseaux. Toutefois, les fonctionnalités restent inchangées malgré les changements de contexte.

Les solutions reposant sur l'adaptation structurelle et l'adaptation de déploiement permettent d'adapter les fonctionnalités en fonction du contexte. Une application est alors représentée par un assemblage de composants qu'il est possible de modifier par des opérations élémentaires telles que l'ajout, la suppression de composants ainsi que de connexions entre ces composants. Ces opérations élémentaires agissent sur la structure de l'application.

Parmi les solutions basées sur une adaptation structurelle en fonction du contexte, l'article de O. Riva, T. Nadeem,C. Borcea, L. Iftode, Context-Aware Migratory Services, in Ad Hoc Networks IEEE Transactions on Mobile Computing, Vol. 6, No. 12, December 2007, propose un modèle de service permettant aux réseaux adhoc de fournir des services capables de s'adapter au contexte afin d'offrir une continuité de service au client. Un service de migration supervise les services et réagit en déclenchant des migrations de services chaque fois qu'un noeud n'est plus capable de supporter l'exécution du service, provoquant la poursuite du service sur un autre nœud. L'aspect migration est rendu invisible aux applications clientes par l'utilisation d'un unique terminal virtuel. On connaît également une approche basée sur des composants légers pour concevoir des services web composites, qui est décrite dans l'article de V. Maurin, N. Dalmasso, B. Copigneaux, S. Lavirotte, G. Rey, J. Y. Tigli, Simply engine-wcomp : plate-forme de prototypage rapide pour l'informatique ambiante basée sur une approche orientée services pour dispositifs réels/virtuels, David Menga and Florence Sedes, editors, UbiMob, volume 394 of ACM International Conference Proceeding Series, pages 83-86. ACM, 2009. Cette solution permet de construire des applications sous forme de graphes de services web basés sur le concept de conteneur. D'autre part, elle fournit un intergiciel basé sur le concept d'Aspects d'Assemblage permettant d'adapter les services web. Une telle solution permet la réutilisation des services et par suite une extensibilité et une communication basée sur les événements, ce qui garantit une forte réactivité du système. Un autre avantage de cette solution est qu'elle permet la mobilité des applications (paradigme services web) et apporte une flexibilité au niveau de la structure qu'il est possible d'adapter (paradigme composant).

Le projet MUSIC (R. Rouvoy, P. Barone, Y. Ding, F. Eliassen, S. Hallsteinsen, J. Lorenzo, A.Mamelli, and U. Scholz. MUSIC: Middleware Support for Self-Adaptation in Ubiquitous and Service-Oriented Environments - Book on Software Engineering for Self-Adaptive Systems (SEfSAS). LNCS 5525 –2009) fournit un intergiciel permettant la reconfiguration d'applications mobiles et sensibles au contexte. Le processus d'adaptation défini dans MUSIC repose sur les principes de l'adaptation par planification (« planning based adaptation » en langue anglo-saxonne). L'adaptation par planification fait référence à la capacité de reconfiguration d'une application en réponse aux changements de contexte en exploitant les connaissances de sa composition et des métadonnées de Qualité de Service associées à chacun des services la constituant.

Dans l'article de D. Ayed, C. Taconet, G. Bernard, and Y. Berbers. Cadecomp, Context-aware deployment of component-based applications", J. Network and Computer Applications, 31(3) 2008, un intergiciel est proposé pour le déploiement sensible au contexte des applications basées composants. Cet intergiciel étend les services de déploiement existants en y intégrant les capacités d'adaptation nécessaires au domaine des applications mobiles et des périphériques contraints. Il propose un déploiement automatique à la volée et sensible au contexte : une application est installée au moment de son accès et désinstallée juste après la fin de son utilisation. Les applications sont considérées comme une collection de composants distribués sur le réseau et reliés entre eux via des ports. Le déploiement est défini selon cinq paramètres : l'architecture de l'application, le placement des instances des composants, le choix de leur implémentation, les propriétés des composants et leurs dépendances. Cet intergiciel repose sur un modèle de données permettant de décrire le contexte qui agit sur le déploiement et de définir des contrats de déploiement qui associent à chaque situation de contexte toutes les variations possibles des paramètres de déploiement. Le contexte modélise essentiellement les caractéristiques des instances des composants. Ces informations sont collectées lors de la spécification et du développement par le producteur du composant. Elles permettent de spécifier des contraintes sur le placement des composants ainsi que sur les connexions, obligatoires ou optionnelles.

La plateforme à service OSGi (acronyme pour « Open Services Gateway initiative ») implémente un modèle de composant (appelé « Bundle », terme anglo-saxon signifiant groupement). Ces derniers possèdent un cycle de vie leur permettant d'être arrêté, démarrés, mis à jour et désinstallés à chaud. Le service appelé « registry » (terme anglo-saxon signifiant « registre ») permet d'enregistrer des modèles de composant (bundle) en tant que services et d'en détecter l'apparition ou la suppression d'autres. OSGi est basé sur la découverte de services. Toutefois, la plateforme OSGi ne supporte pas la migration des composants entre périphériques et est basée sur la découverte de services.

Les solutions existantes proposent ainsi différentes approches d'adaptation structurelle des applications (canevas logiciel, intergiciel, plateforme d'exécution) dans lesquelles la répartition de charge est soit statique, soit planifiée et non contextualisée. Aucune des approches proposées ne permet une supervision décentralisée, entièrement dynamique et transparente des composants applicatifs sur un ensemble de dispositifs mobiles, pouvant être de nature différente et connectés par tous types de réseaux, en fonction du contexte. Par ailleurs, aucune des solutions existantes ne propose un tel système de supervision capable de réaliser de la migration de composants logiciels en environnement mobile.

Définition générale de l'invention

A cet effet, l'invention propose un système de supervision d'applications s'exécutant sur un ensemble de dispositifs électroniques reliés entre eux par un ou plusieurs réseaux. Chaque dispositif comprend une entité de supervision locale, les entités de supervision coopérant entre

elles pour contrôler les applications s'exécutant sur les dispositifs électroniques. Chaque application comprend un ensemble de composants applicatifs, chaque composant applicatif étant encapsulé dans un conteneur par l'entité de supervision du dispositif hébergeant le composant, et les composants étant reliés entre eux par des connecteurs. Avantageusement, l'entité de supervision d'un dispositif donné, dit dispositif source, est configurée pour exécuter les étapes suivantes, en réponse à la réception d'une commande de migration d'un composant vers un dispositif cible :

- arrêter le composant, l'arrêt du composant interrompant l'arrivée de données dans les connecteurs d'entrée du composant,

-sérialiser et encapsuler les propriétés du composant dans un conteneur de l'entité de supervision,

-envoyer un message de demande migration à l'entité de supervision du dispositif cible, ledit message comprenant le composant sérialisé et encapsulé, et

- rediriger les connecteurs du composant en fonction de l'état des connexions de chaque connecteur sur le dispositif source.

Selon une caractéristique de l'invention, en réponse au message de demande de migration du composant, l'entité de supervision du dispositif cible est configurée pour déterminer si le code exécutable associé au composant est disponible sur le dispositif cible.

L'entité de supervision du dispositif cible peut charger le code exécutable associé au composant pour désencapsuler et dé-sérialiser les propriétés du composant en utilisant un chargeur de code, et démarrer le composant, si le code associé au composant est disponible sur le dispositif cible.

En particulier, l'entité de supervision du dispositif cible peut envoyer un message de demande du code associé au composant vers les entités de supervision d'un ensemble de dispositifs, si le code associé au composant n'est pas disponible sur le dispositif cible, tandis que l'entité de supervision du dispositif cible est apte à charger le code associé au composant pour désencapsuler et dé-sérialiser les propriétés du composant en utilisant un chargeur de code, et démarrer le composant, en réponse à la réception dudit code depuis l'une au moins des entités de supervision de l'ensemble d'entités de supervision.

En particulier, l'ensemble d'entités de supervision comprend les entités de supervision des dispositifs voisins accessibles par diffusion si le dispositif cible supporte l'envoi par diffusion de message.

Le message de demande de code peut être envoyé à un proxy prédéfini si le dispositif cible ne supporte l'envoi de message par diffusion, et le proxy est apte à relayer le message de demande de code par diffusion, l'ensemble d'entités de supervision comprenant les entités de supervision des dispositifs auxquels est diffusé le message relayé par le proxy.

En variante, lorsque l'entité de supervision du dispositif cible maintient un service de nom de domaine pour mémoriser les informations relatives aux entités de supervision avec lesquelles elle

communique, l'ensemble d'entité de supervision est déterminé à partir des informations maintenues dans le service de nom de domaine, si le dispositif cible ne supporte l'envoi de message par diffusion.

Le code associé au composant peut comprendre un ensemble de classes, let chargeur de code étant alors un chargeur de classes associé au composant et créé localement sur le dispositif en réponse à la création de la première instance du composant.

En particulier, le code associé au composant peut être maintenu dans un fichier de code de type JAR.

Selon un aspect de l'invention, le lien entre le chargeur de classe et le composant peut être enregistré dans le conteneur du composant.

Selon un autre aspect de l'invention, la redirection des connecteurs par l'entité de supervision sur le dispositif source est mise en œuvre de manière indépendante par rapport au démarrage du composant migré par l'entité de supervision sur le dispositif cible.

Lorsque le composant sur le dispositif source était lié à un connecteur distribué entre le dispositif source et le dispositif cible, l'entité de supervision sur le dispositif source peut rediriger le connecteur distribué en créant un connecteur interne sur le dispositif cible.

En variante, lorsque le composant sur le dispositif source était lié à un connecteur interne lié au composant sur le dispositif source, l'entité de supervision sur le dispositif source peut rediriger le connecteur interne en créant un connecteur distribué sur le dispositif cible et le dispositif source, si le dispositif cible et le dispositif source ont des réseaux de communication compatibles, ou d'un connecteur relai entre le dispositif cible et le dispositif source, si le dispositif cible et le dispositif source n'ont pas de réseaux de communication compatibles.

Lorsque le composant sur le dispositif source était lié à un connecteur distribué ou à un connecteur relai entre le dispositif source et un dispositif donné distinct du dispositif source et du dispositif cible, l'entité de supervision sur le dispositif source est apte à rediriger le connecteur distribué en créant un connecteur distribué sur le dispositif cible et le dispositif donné si le dispositif cible et le dispositif donné ont des réseaux de communication compatibles, ou d'un connecteur relai entre le dispositif cible et le dispositif donné, si le dispositif cible et le dispositif source n'ont pas de réseaux de communication compatibles.

La création d'un connecteur distribué ou relai peut comprendre la synchronisation des parties du connecteur distribué ou relai par un échange de messages d'acquittement.

La synchronisation des parties du connecteur distribué ou relai peut comprendre la création d'interfaces logicielles de communication entre les parties de connecteurs, lesdites interfaces logicielles étant utilisées pour le transfert de données sur ledit connecteur distribué ou relai.

Selon un autre aspect de l'invention, l'entité de supervision sur chaque dispositif est apte à contrôler chaque connecteur hébergé sur le dispositif en utilisant un conteneur encapsulant le connecteur.

L'entité de supervision du dispositif cible est apte à communiquer avec le conteneur du composant pour l'arrêter jusqu'à ce qu'un connecteur lui soit connecté.

Selon une caractéristique de l'invention, les données échangées entre les composants peuvent être encapsulées dans une classe, une donnée reçue par un composant hébergé sur un dispositif étant désencapsulée et traitée par l'entité de supervision si le dispositif dispose de la classe du composant.

Selon un autre caractéristique de l'invention, le message de demande de migration peut comprendre des informations relatives à l'état des entrées et sorties du composant.

L'invention propose en outre un procédé de supervision d'applications s'exécutant sur un ensemble de dispositifs électroniques reliés entre eux par un ou plusieurs réseaux. Chaque dispositif comprenant une entité de supervision locale, les entités de supervision coopérant entre elles pour contrôler les applications s'exécutant sur les dispositifs électroniques, chaque application comprenant un ensemble de composants applicatifs, chaque composant applicatif étant encapsulé dans un conteneur par l'entité de supervision du dispositif hébergeant le composant, et les composants étant reliés entre eux par des connecteurs. Le procédé comprend, en réponse à la réception par l'entité de supervision d'un dispositif donné, dit dispositif source, d'une commande de migration d'un composant du dispositif source vers un dispositif cible :

- arrêter le composant sur le dispositif source, l'arrêt du composant interrompant l'arrivée de données dans les connecteurs d'entrée du composant,
-sérialiser et encapsuler les propriétés du composant dans un conteneur de l'entité de supervision,
-envoyer un message de demande de migration à l'entité de supervision du dispositif cible, ledit message comprenant le composant sérialisé et encapsulé, et
- rediriger les connecteurs du composant en fonction de l'état des connexions de chaque connecteur sur le dispositif source.

Le système de supervision selon l'invention permet ainsi de migrer un composant de manière transparente entre les dispositifs sans que les composants qui communiquent avec ce composant n'aient besoin d'en être informés. Ainsi, les composants liés à un composant migré peuvent continuer à fonctionner, sans être impactés.

La migration de composant selon l'invention offre en outre une solution dynamique et transparente au partage de ressources en environnement mobile ou contraint, en particulier pour les

applications liées nécessitant de répartir les charges afin d'économiser de l'énergie, de répartir les débits réseaux, d'utiliser des capteurs déportés, de déplacer une exécution, etc.

Brève description des dessins

D'autres caractéristiques et avantages de l'invention apparaîtront à la lecture de la description détaillée qui suit et des figures des dessins annexés dans lesquels:

- La figure 1 est un schéma représentant l'architecture générale du système de supervision selon une forme de réalisation de l'invention ;

- La figure 2 représente des exemples de dispositifs hébergeant des composants contrôlés par le système de supervision, selon des modes de réalisation de l'invention ;

- La figure 3 représente la structure générale de communication entre composants applicatifs selon une forme de réalisation de l'invention ;

- La figure 4 est un schéma d'un modèle de composant selon une forme de réalisation de l'invention;

- La figure 5 illustre le cycle de vie d'un Composant, selon certaines formes de réalisation de l'invention;

- La figure 6 est un organigramme représentant les étapes mises en œuvre par l'entité de supervision sur un dispositif source pour migrer un composant vers un dispositif cible, selon certaines formes de réalisation de l'invention ;

- La figure 7 est un organigramme représentant les étapes mises en œuvre par l'entité de supervision sur un dispositif cible pour démarrer un composant migré depuis un dispositif cible, selon certaines formes de réalisation de l'invention ;

- La figure 8 illustre le chargement de classe mis en œuvre par l'entité de supervision sur un dispositif cible pour charger les classes associées à un composant migré depuis un dispositif cible, selon certaines formes de réalisation de l'invention ;

- La figure 9 montre les interactions entre des connecteurs et un composant auquel ils sont reliés, selon un exemple de réalisation de l'invention ;

- La figure 10 représente un diagramme d'états d'une unité d'entrée, selon une forme de réalisation de l'invention ;

- La figure 11 est un schéma structurel d'un modèle de Connecteur supporté par le système de supervision selon une forme de réalisation de l'invention ;

- La figure 12 représente un connecteur Interne, selon une forme de réalisation de l'invention ;

- La figure 13 représente un connecteur distribué, selon une forme de réalisation de l'invention;

- La figure 14 représente un connecteur relai du système de supervision selon une forme de réalisation de l'invention;

- La figure 15 est un tableau illustrant la redirection des connecteurs, selon certaines formes de réalisation de l'invention ;

- La figure 16 est un organigramme représentant la création d'un connecteur distribué selon une forme de réalisation de l'invention;

- La figure 17 est un organigramme représentant la suppression d'un connecteur distribué selon une forme de réalisation de l'invention;

- La figure 18 est un organigramme représentant la création d'un connecteur distribué selon une forme de réalisation de l'invention entre deux dispositifs, lorsque les dispositifs ont des moyens de communication incompatibles;

- La figure 19 illustre la communication entre les entités de supervision ;

- La figure 20 est un schéma représentant le mécanisme de synchronisation des connecteurs selon les modes de réalisation de l'invention ; et

- La figure 21 représente l'architecture du noyau du système de supervision.

Description détaillée

La figure 1 représente un système de supervision d'applications 10 mettant en œuvre le procédé de chargement dynamique selon les formes de réalisation de l'invention. Le système de supervision 10 représenté est un système dynamique et décentralisé pour contrôler les applications d'un ensemble de dispositifs électroniques 5 connectés entre eux par des réseaux de communication, par exemple de type WIFI, ou satellite (GSM, 3G, etc.). Les dispositifs électroniques (encore désignés par les expressions « dispositif hôtes » ou « hôtes » ou « machines » dans la suite de la description) peuvent comprendre tout type de dispositif électronique, notamment des dispositifs électroniques mobiles, comme des ordinateurs personnels tels que PC1 et PC2), des téléphones mobiles intelligents (appelé « Smartphone » en langue anglo-saxonne) tels que T1 et T2, des tablettes informatiques telles que TI1, etc. Les réseaux de communication supportés par les dispositifs électroniques 5 peuvent comprendre plusieurs types de réseaux.

Selon cette approche décentralisée, le système de supervision 10 comprend un ensemble d'entités de supervision 6 (encore appelée « entités de supervision locales »), chacune hébergée sur un dispositif électronique respectif 5. Les entités de supervision coopèrent entre elles pour superviser de manière dynamique les composants applicatifs qui s'exécutent sur l'ensemble des dispositifs 5 en fonction du contexte et des ressources. Ces entités de supervision contrôlent dynamiquement le cycle de vie des composants qui peut comprendre la création d'un composant sur un dispositif, la suppression d'un composant d'un dispositif 5 ou la migration d'un composant applicatif entre un dispositif source et un dispositif cible, notamment pour tenir compte du contexte d'exécution et des ressources matérielles. Les entités locales de supervision 6 peuvent en outre être configurées pour collecter des informations sur l'utilisation des ressources matérielles des dispositifs électroniques, telle que la batterie, la mémoire ou le CPU (acronyme pour l'expression anglo-saxonne « Central Processing Unit »), et/ou le contexte d'exécution, représenté par exemple par le réseau, les besoins des utilisateurs, ou les règles d'usage de l'application, etc. Les entités de supervision locales 6 peuvent alors déclencher dynamiquement des actions de reconfiguration des composants applicatifs qui s'exécutent sur les dispositifs électroniques 5 de manière transparente pour l'utilisateur selon une approche décentralisée (aucun serveur central n'est requis). Une telle reconfiguration permet le déploiement et le redéploiement dynamique d'applications et peut notamment impliquer la création, la suppression ou la migration de composants. Les composants (C1,C2, C3) peuvent ainsi être migrés de dispositif en dispositif, de manière indépendante, quel que soit le type du dispositif source et du dispositif d'accueil en fonction du contexte d'exécution et/ou des ressources matérielles, tout en poursuivant leur exécution sur chaque dispositif qui les accueille (redémarrage à chaud).

Le système de supervision 10 selon l'invention permet notamment d'assurer une continuité de service en cas d'indisponibilité d'un dispositif électronique 5. En particulier, en cas de dysfonctionnement d'un dispositif, les entités de supervision 6 peuvent fournir à l'application tout ce qu'elle attend tout en pérennisant au maximum son exécution et en anticipant des situations critiques qui peuvent être liées par exemple à la durée de vie d'une batterie ou le dépassement de la bande passante disponible. Le système de supervision 10 permet notamment de répartir la charge de l'application sur des dispositifs électroniques 5 voisins, et d'optimiser la répartition des composants de l'application sur les dispositifs électroniques voisins, lorsque le dispositif 5 sur lequel s'exécute l'application est confronté à des problèmes de ressources, comme des déconnexions dues au déchargement de la batterie.

La figure 2 montre des exemples de dispositifs 5, de types différents, exécutant des applications contrôlées par le système de supervision selon l'invention (non représenté sur la figure). Une application peut être composée d'un ou de plusieurs services et chaque service peut être réalisé par un ou plusieurs assemblages de composants (représentés par les carrés grisés) reliés par des connecteurs (représentés par les lignes fléchées). L'état d'une application est ainsi constitué par l'ensemble des états des composants, des dispositifs 5, des connecteurs entre les composants et de l'environnement. Les entités de supervision 6 sont configurées pour recueillir de telles données

afin de les traiter et de déclencher des commandes de reconfiguration qui conviennent. En réponse à ces commandes de reconfigurations, les entités de supervision peuvent coopérer entre elles pour modifier l'architecture générale de l'application (pouvant impliquer une modification de la répartition des composants sur l'ensemble des dispositifs impliqués dans l'application), en migrant des composants (carrés grisés) d'un dispositif 5 à un autre selon un procédé de migration, et/ou en remplaçant un ou plusieurs composants par d'autres.

Le système de supervision 10 selon l'invention permet ainsi le déploiement dynamique et la reconfiguration dynamique d'applications sur tout un ensemble de dispositifs pouvant inclure tout type de dispositif électronique 5 (ordinateur portable, tablette informatique, téléphones intelligents, etc.), quel que soit le réseau de communication qu'il supporte.

La figure 3 représente la structure générale des applications 3 contrôlées par le système de supervision 10. Des applications modulaires 3 à base de composants logiciels distribués peuvent être exécutées sur les dispositifs 5. La modularité qui en résulte permet de proposer des solutions ad hoc, reconfigurables à chaud, qui garantissent la continuité des applications et leur pérennité dans le temps.

Chaque application 3 selon l'invention comprend un ensemble de fonctionnalités 30 interconnectées. Chaque fonctionnalité est elle-même constituée d'un ensemble de composants logiciels 302 reliés par des connecteurs 303. Ces fonctionnalités 30 peuvent être réalisées de différentes manières à partir d'assemblages de composants 302 différents. Le système de supervision 10 dispose donc de plusieurs décompositions fonctionnelles correspondant aux diverses configurations de l'architecture.

Pour pouvoir s'adapter dynamiquement, chaque application 3 a une capacité de réflexivité qui lui permet d'avoir une connaissance d'elle-même. Cette capacité de réflexivité lui permet de remplacer un service défectueux ou inadapté au contexte courant par un autre service.

A cet effet, le système de supervision 10 est configuré pour acquérir la connaissance de l'application en cours ainsi que le contexte de l'application de manière dynamique et transparente.

Le système de supervision peut être utilisé par exemple pour superviser une application de prise de notes à destination des jardiniers d'un parc botanique équipé chacun de dispositifs 5, afin de leur permettre de réaliser un suivi des plantes et de leurs évolutions (prises de photos, prises de notes, localisation, etc.). L'application peut être hébergée sur des téléphones intelligents 5 (smartphones) permettant aux jardiniers de prendre des notes géolocalisées, écrites ou orales. Chaque plante est accompagnée d'un panneau identificateur par code barre de type code QR (acronyme pour l'expression anglo-saxonne « Quick Response », signifiant littéralement « réponse rapide »), la lecture de ces codes QR facilitant la désignation des plantes dans les notes. Pour des raisons pratiques (position du preneur de notes par rapport au panneau), la lecture des codes QR

peut être déléguée à un autre jardinier. Il est également possible de permettre à un autre jardinier de compléter une prise de notes.

Lorsqu'un jardinier arrive dans le parc et dispose d'un téléphone intelligent hébergeant une entité de supervision locale 6 en cours d'exécution, un composant d'IHM (Interface Homme Machine) peut lui être déployé, offrant 3 boutons : Edition/Sélection d'une note, enregistrement d'une note vocale, activation de la géolocalisation.

S'il sélectionne le premier bouton, un composant C1 de choix de note écrite ou orale est déployé lui permettant de sélectionner une note déjà présente tandis qu'un composant C2 d'accès à la base de données des notes est déployé sur le serveur central du parc. Ces deux composants C1 et C2 sont reliés entre eux par des connecteurs. Si le jardinier choisit une note écrite, un composant d'édition C3 est déployé. Lorsqu'il souhaite introduire une lecture (scan) de code QR dans sa note, la liste des dispositifs des autres jardiniers présents dans le parc lui est proposée. Il peut alors choisir de scanner lui-même la note ou de déléguer cette tâche à l'un de ses collègues. Dans ce dernier cas, un composant d'alerte (vibreur et message) C4 est déployé sur le terminal de ce collègue afin de l'avertir de la demande. Si le collègue accepte, ce composant d'alerte C4 est remplacé par un composant C5 de lecture de code QR qui est relié par connecteur au composant de prise de notes C6 du premier jardinier. Dès que la lecture du code a été faite et le résultat inséré dans la note, le composant de lecture (scan) et le connecteur sont automatiquement supprimés.

Si le jardinier a mis en marche la géolocalisation, lorsqu'il sauvegarde sa note sur le serveur du parc, elle pourra être automatiquement géolocalisée. Lors d'une consultation ultérieure de cette note, cette géolocalisation ainsi que la date seront accessibles.

Une application est ainsi constituée de composants logiciels 302 lies par des connecteurs 303. L'architecture d'une application est conçue pendant l'exécution et peut être modifiée pendant que l'application s'exécute sans qu'il soit nécessaire de l'arrêter (reconfiguration à chaud).

Les entités de supervision 6 coopèrent entre elles pour ajouter, supprimer, connecter, déconnecter et/ou ou migrer les composants de chaque application. En complément, une interface utilisateur peut être prévue pour permettre la gestion des entités de supervision sur chaque dispositif impliqué dans une application donnée.

Les applications supervisées par le système de supervision 10 sont ainsi réparties sur l'ensemble des dispositifs 5. Toutefois, contrairement aux approches classiques, elles ne sont pas dépendantes de l'existence de serveurs centraux. Chaque dispositif 5 peut ainsi dialoguer directement avec les autres dispositifs par des liens entre les composants qui sont établis par les entités de supervision 10.

Le système de supervision 10 peut ainsi migrer un composant indépendamment des composants qui communiquent avec lui. Les composants liés peuvent alors continuer à fonctionner sans être impacté.

La migration d'un composant d'un dispositif source vers un dispositif cible nécessite de faire suivre les liaisons réseaux (via les connecteurs) qui sont établies depuis le composant déplacé et vers ce composant et de conserver l'état du composant pour pouvoir le redémarrer à chaud sur le dispositif où il est migré (dispositif cible). Or, le dispositif cible peut être d'un type différent du dispositif source et la migration peut nécessiter de changer d'interface réseau (par exemple, pour passer du wifi vers de la 3G). Par ailleurs, pour assurer un fonctionnement dynamique du système de supervision, les opérations liées à migration du composant doivent se faire de manière totalement transparente pour les composants en relation avec le composant migré.

La figure 4 illustre les interactions entre les composants 302 et des connecteurs 303 selon l'invention. Le système de supervision 10 forme une plateforme de supervision qui est distribuée sur les dispositifs 5 de manière à avoir connaissance des composants 302 et des connecteurs 303 déployés. Elle est en outre configurée pour récupérer les informations de contexte que ceux-ci peuvent lui transmettre. En fonction de ces informations, le système de supervision 10 détermine si des reconfigurations doivent être mises en œuvre, impliquant le déploiement et le redéploiement dynamique de composant.

Les entités de supervision 6 utilise des conteneurs 305 pour surveiller le fonctionnement des composants 302 et la circulation des flux de données entre les composants 302 connectés par les connecteurs 303, sur le dispositif associé. Les conteneurs 305 sont configurés pour recueillir les informations entre les entités 302 et 303. Ils permettent en outre de gérer l'hétérogénéité matérielle et logicielle des dispositifs 5 ainsi que la mobilité des dispositifs.

La figure 4 montre plus précisément un modèle de conteneur 305 de composant 302 de type Composant Métier. Dans la suite de la description, les termes « composants métier », « composants applicatifs », « composants logiciels », ou simplement « composants » pourront être utilisés de manière similaire pour désigner un composant. Selon ce modèle de conteneur 305, la logique métier contenue dans un composant est séparée de la supervision gérée par un conteneur. Le Composant 302 peut recevoir plusieurs flux de données en entrée et produire plusieurs flux de données en sortie. Par ailleurs, chaque flux de sortie peut être dupliqué. Le conteneur 305 représenté encapsule un seul composant 302 et peut implémenter un ensemble de propriétés non-fonctionnelles, telles que la gestion du cycle de vie, la récupération des informations de qualité de service et la gestion des communications. Le conteneur 305 associé à un composant sur un dispositif est créé par l'entité de supervision 6 qui accueille le composant, tandis que le code associé au composant peut être chargé dynamiquement.

Le conteneur 305 possède trois types unités :

- Une ou plusieurs unités d'entrée (UE) désignées par la référence 41 pour recevoir les flux de donnée en entrée,
- Une ou plusieurs unités de sortie (US) désignées par la référence 42 pour recevoir les flux de donnée en sortie, et
- Une Unité de contrôle (UC) désignées par la référence 40.

Les unités UE d'entrée 41 et les unités US de sortie 42 peuvent être reliées à un ou à plusieurs connecteurs 303. Ces unités 41 et 42 permettent au Composant 302 de lire et d'écrire des données provenant d'autres composants ou à destination d'autres composants. Le composant 302 peut ainsi lire des données via les unités d'entrée 41, effectuer son traitement et écrire les résultats dans les unités de sortie 42 US. L'entité de supervision 6 du dispositif hébergeant le composant peut utiliser l'unité de contrôle 40 (UC) pour superviser le conteneur 305 de composant. Chaque conteneur 305 de composant 302 peut en effet enregistrer son unité de contrôle 40 comme un service auprès de l'entité de supervision du dispositif hébergeant le composant, afin de permettre à l'entité de supervision 6 de contrôler les différentes phases du cycle de vie du composant.

L'unité de contrôle 40 contrôle les éléments du conteneur 305 de composant. Par exemple, le comportement du composant peut être contrôlé par l'unité de contrôle 40 au moyen d'une méthode d'initialisation de composant (appelée init()), d'une méthode de suppression de composant (appelée « destroy ») ou encore d'une méthode d'activation de composant (appelée « Run_BC »). Les unités d'entrée et de sortie 40 et 41 contrôlent la circulation des données dans le conteneur 305 et donnent au composant des accès à ses flux d'entrée et de sortie.

La figure 5 illustre le cycle de vie d'un composant 302. Les composants associés à une application donnée sont initialement écrits par le concepteur de l'application. L'entité de supervision 6 du dispositif accueillant un composant encapsule le composant 302 dans un conteneur 305 qui en contrôlera le cycle de vie. Le cycle de vie d'un composant peut correspondre à l'appel de méthodes surchargées. Ce cycle de vie d'un composant est généralement similaire à celui d'une « applet » (terme désignant un logiciel qui s'exécute dans la fenêtre d'un navigateur web), d'un MIDlet (acronyme pour « Mobile Information Device applet » et désignant un programme installé dans un dispositif d'information mobile) ou d'une activité du système d'exploitation Android.

Le cycle de vie d'un composant 302 correspond aux trois types d'actions mises en œuvre par les entités de supervision 6: la création d'un composant sur une machine hôte, la suppression d'un composant d'un composant sur une machine hôte et la migration d'un composant entre deux machines hôte reliées en réseau.

La création d'un composant 302 comprend l'instanciation d'un objet de la classe associée au composant, l'encapsulation de cet objet dans un conteneur 305 puis la connexion de ses flux d'entrée et de sortie.

La suppression d'un composant 302 comprend l'arrêt du composant encapsulé puis la suppression de son conteneur 305. Ses flux d'entrée/sortie peuvent rester en attente d'un nouveau composant ou d'être à leur tour supprimés.

La migration d'un composant est mise en œuvre lorsqu'un composant 302 s'exécutant sur un hôte A doit être migré vers un hôte B. L'entité de supervision 6 hébergée sur l'hôte A arrête alors le composant au niveau de l'hôte A comme lors d'une suppression, puis envoie le composant à l'entité de supervision sur l'hôte B, en utilisant le procédé de migration, selon certaines formes de réalisation de l'invention.

Lorsqu'un composant 302 est créé, il passe de l'état « non-présent », désigné par la référence 50, à l'état « Initialisé », désigné par la référence 51, où il exécute une méthode d'initialisation appelée « Init ». Il passe ensuite dans l'état « Actif », désigné par la référence 52, où il exécute une méthode d'exécution appelée « run_BC ». Un composant peut également passer directement de l'état « non-présent » (50) à l'état « actif » (51) lorsqu'il est reçu sur le dispositif électronique 5 suite à une migration. Dans ce cas le composant est démarré à chaud dans le même état que celui où il se trouvait précédemment sur le dispositif source depuis lequel il a été migré, de sorte qu'aucune phase d'initialisation n'est requise. Lors d'appels de fonctions de l'interface de programmation API (acronyme pour l'expression anglo-saxonne « Application Programming Interface »), comprenant par exemple des fonctions de type lecture, écriture, services du système de supervision, le composant peut être mis dans un état « Bloqué », désigné par la référence 53. Depuis l'état Actif 52 et l'état Bloqué 53, le composant 302 peut être arrêté et mis dans l'état « Détruit » (état 54) où il exécute une méthode appelée « Destroy ». Un composant 302 peut passer de l'état actif 52 à l'état détruit 54 s'il est en fin d'activité ou a été migré vers un autre dispositif. Un composant peut notamment passer de l'état « bloqué » 53 à l'état « détruit » 54 sur un dispositif donné 5, lors d'une migration vers un autre dispositif électronique 5. Les transitions entre états sont provoquées par des exceptions. Une exception désigne une interruption de l'exécution d'un programme en réponse à un évènement spécifique pouvant entraîner un changement d'état.

Selon une caractéristique de l'invention, deux classes d'exception peuvent être utilisées :

- Une première classe appelée « StopBCException » qui représente une exception qui peut être utilisée lors de tentatives de lecture ou d'écriture ou lors d'accès aux services du système de supervision (passage de l'état actif 52 à l'état bloqué 53).
- Une deuxième classe appelée « InterruptedException » qui peut être utilisée pour provoquer des changements d'état lorsqu'un composant 302 est dans l'état bloqué 53 sur un sémaphore ou est suspendu.

Après l'exécution de la méthode « Destroy » (dans l'état « détruit » 54) ou après un laps de temps prédéfini, si l'exécution de la méthode « Destroy » ne se termine pas, le composant 302 peut être détruit ou migré. Lors d'une migration du composant 302, ses propriétés sont sérialisées et sur le nouveau dispositif électronique d'accueil 5, il est directement mis dans l'état Actif 52.

Le procédé de migration selon les formes de réalisation de l'invention est mis en œuvre par le système de supervision 10 pour déplacer un composant en cours d'exécution afin qu'il poursuive son exécution sur un autre dispositif, sans perturber l'exécution du composant migré (redémarrage à chaud) ni l'exécution des composants liés au composant. Le procédé de migration s'appuie notamment sur une coopération entre les entités de supervision 6 et des échanges internes entre les entités de supervision 6 impliquées dans la migration et le conteneur du composant 302 qui fait l'objet de la migration.

La figure 6 est un organigramme des étapes mises en œuvre par l'entité de supervision d'un dispositif source A pour migrer un composant CM vers un dispositif cible B.

En réponse à une commande de migration du composant du dispositif A vers le dispositif B (600), l'entité de supervision locale 6 sur le dispositif A arrête le composant CM sur le dispositif A, à l'étape 602. La commande d'arrêt du composant peut être transmise à l'unité de contrôle 40 du composant. Une telle commande a pour effet d'arrêter les composants CM. Les connecteurs liés au composant CM peuvent continuer à fonctionner. Toutefois, les connecteurs 303 reliés à des sorties du composant ne reçoivent plus de données de la part du composant, en réponse à son arrêt. Les connecteurs reliés aux entrées du composant CM conservent les données qu'elles reçoivent et qu'elles ne peuvent plus transmettre au composant dans des mémoires tampons.

A l'étape 604, le composant CM est sérialisé et encapsulé dans une classe spécifique par l'entité de supervision sur le dispositif source. La sérialisation du composant comprend la sérialisation des propriétés du composant. Les propriétés sérialisées sont ensuite encapsulées dans un conteneur de l'entité de supervision 6 du dispositif source. La lecture des propriétés par l'entité de supervision sur le dispositif cible B ne pourra se faire que si le dispositif B dispose du code des classes de ces propriétés pour pouvoir désencapsuler les propriétés du composants et les affecter au composant.

A l'étape 606, l'entité de supervision locale 6 envoie à l'entité de supervision sur le dispositif B un message de demande de migration du composant CM vers le dispositif B. Le message de demande de migration peut comprendre des informations liées au composant, notamment le nom de la classe du composant, ainsi que le composant sérialisé et encapsulé. Le message de demande de migration peut comprendre en outre l'état des entrées et sorties du composant CM. Cet état peut comprendre une ou plusieurs listes des connecteurs qui sont reliés à chacune des entrées du composant CM et à chacune des sorties du composant CM. L'état du composant transmis dans le message pourra être utilisé par l'entité de supervision locale 6 du dispositif cible pour reconstituer les connexions du composant en réponse à sa migration.

A l'étape 608, l'entité de supervision locale 6 sur le dispositif A communique avec d'autres entités de supervision pour rediriger les connecteurs qui étaient liés au composant migré. Plus précisément, les connecteurs 303 d'entrée et de sortie du composant CM sur le dispositif source sont redirigés de façon à ce qu'ils aboutissent désormais sur le dispositif B et non plus sur le

dispositif A. La redirection d'un connecteur 303 dépend notamment du type de connecteur et de sa configuration avant la migration du composant.

Le procédé de migration permet ainsi l'envoi des propriétés du composant et la redirection des connecteurs qui lui étaient liés vers un dispositif cible. La création et l'encapsulation de ce composant sur le dispositif cible est ensuite pris en charge par l'entité de supervision 6 du dispositif cible, à partir d'un fichier de code exécutable associé au composant. Par conséquent, le composant qui s'exécute sur le dispositif d'arrivée (dispositif cible) peut être différent de celui qui s'exécutait sur le dispositif de départ (dispositif source) dès lors qu'il a les mêmes propriétés que le composant de départ sur le dispositif source. Ainsi par exemple un composant qui affiche un texte sur le dispositif source A (de type PC par exemple) peut, après migration, devenir un composant qui lit le texte à haute voix par synthèse de parole sur le dispositif cible B (par exemple téléphone intelligent). Ces deux composants ont ainsi un rôle identique (présenter un contenu textuel à l'utilisateur) qu'ils réalisent de deux façons différentes.

Par ailleurs, les étapes du procédé de migration peuvent être mises en œuvre par l'entité de supervision sur le dispositif source indépendamment de l'état d'avancement de la migration sur le dispositif cible. Ainsi, la redirection des connecteurs peut être mise en œuvre par l'entité de supervision du dispositif source même si le composant n'a pas été encore créé sur le dispositif cible.

Dans une forme de réalisation préférée de l'invention, la partie applicative (comprenant notamment le code des classes des composants et des objets échangés) n'est pas résidente sur les dispositifs électroniques mobiles pour limiter la surcharge des dispositifs 5. Le code exécutable associé au composant 302 est alors chargé dynamiquement lors de la création de la première instance du composant sur un dispositif et supprimé lors de la suppression de sa dernière instance sur le composant, tandis que les conteneurs 305 de composant sont créés par l'entité de supervision 6 hébergée sur le dispositif. Aussi, pour ne pas occuper inutilement de la place en mémoire, le code associé au composant migré sur le dispositif A peut être supprimé lors de la migration du composant, si aucune autre instance du composant 302 ne l'utilise. Par ailleurs, pour pouvoir démarrer le composant, l'entité de supervision 6 du dispositif cible B doit disposer du code associé au composant migré.

Selon une forme de réalisation préférée de l'invention, un composant 302 comprend plusieurs classes et peut inclure des ressources (par exemple images, sons,..) et des bibliothèques.

Le code associé à un composant peut alors se présenter sous la forme d'un fichier de code. En particulier, pour les dispositifs 5 dépendant de la machine virtuelle JAVA, le fichier de code de composant est un fichier de code binaire Java appelé fichier JAR (fichier interprétable par le système d'exploitation). Un tel fichier de code peut comprendre les informations suivantes :

- Le code binaire (« byte code » en langue anglo-saxonne) de chaque classe nécessaire au fonctionnement de ce composant 302 (incluant des bibliothèques) et des classes des objets échangés entre composants;

- Les ressources utilisées par le composant (par exemple, images, fichiers, etc.);

- Les bibliothèques utilisées par le composant;

- Le nom de la classe du composant peut être inclus par exemple dans un fichier de type « manifest ». Le nom de la classe peut être utilisé par les entités de supervision locales 6 pour retrouver un fichier de code associé à un composant migré (si le dispositif est le dispositif d'accueil du composant ou s'il en reçoit la demande depuis l'entité de supervision d'un autre dispositif).

La suite de la description sera faite en référence à une telle structure de fichier de code à titre d'exemple non limitatif.

La figure 7 est un organigramme des étapes mises en œuvre par l'entité de supervision 6 sur le dispositif B pour traiter le message de demande de migration reçu de l'entité de supervision du dispositif A.

En réponse à la demande de migration reçue de l'entité de supervision du dispositif A (700), l'entité de supervision 6 sur le dispositif B détermine si elle dispose du code associé au composant migré à l'étape 702 (en particulier, classes associées au composant), ce qui peut être le cas par exemple si le dispositif B héberge un composant de la même classe que le composant CM ou gère un dépôt de code de composant. Si elle dispose des classes associées au composant, l'entité de supervision du dispositif B charge les classes (703). Sinon, l'entité de supervision 6 sur le dispositif B envoie une demande de fichier de code associé au composant CM à un ensemble d'entités de supervision 6 hébergées sur d'autres dispositifs (704) et attend de recevoir le fichier de code associé (fichier JAR par exemple). Si l'entité de supervision sur B reçoit le fichier de code associé au composant CM (706), elle charge les classes associées au composant (classes du composant et des données échangées par le composant) à partir du fichier de code à l'étape 707. Le fichier de code reçu peut être stocké localement sur le dispositif A (par exemple, sur disque, en mémoire ou sur carte SD selon le type de dispositif).

L'entité de supervision 6 du dispositif B crée alors une instance du composant par désencapsulation et dé-sérialisation de ses propriétés (707) et le démarre à chaud (709).

Le chargement des classes du composant migré peut se faire par appel à un chargeur de classes qui lui est associé, selon que le dispositif B dispose ou non des classes du composant (étapes 703 et 707).

Le message de demande de code à l'étape 704 peut être envoyé de différentes manières vers un ensemble d'entités de supervision sur d'autres dispositifs, selon les capacités associées aux réseaux de communication dont dépend le dispositif cible B.

Si le dispositif cible appartient à un réseau de communication permettant la diffusion simple (« broadcast » en langue anglo-saxonne) ou la multi-diffusion (« multicast » en langue anglo-saxonne), l'entité de supervision envoie sa demande par diffusion (« broadcast » ou « multicast »). Le message de demande de code est alors reçu par les entités de supervision des dispositifs couramment connectés sur ce réseau (dispositifs voisins), qui continuent à le relayer de la même façon si elles ne disposent pas du fichier de code recherché. Chaque entité de supervision qui relaie ainsi le message à d'autres entités de supervision peut se désigner comme relai possible pour le retour de la réponse vers l'entité de supervision du dispositif cible B.

Si le dispositif cible B ne permet l'envoi de message par diffusion et dépend d'un autre réseau de communication (par exemple 3G ou 4G), l'entité de supervision 6 sur le dispositif cible peut envoyer le message de demande du code du composant à un dispositif servant de proxy. Le proxy associé à une entité de supervision donnée peut être un dispositif comprenant une entité de supervision 6, disposant d'une adresse publique sur Internet, et doté d'une capacité d'envoi en broadcast/multicast. Le dispositif associé au proxy peut être d'un autre type que le dispositif qui utilise le proxy. Par exemple, un dispositif de type PC peut être le proxy d'un dispositif de type Android. Il peut être préalablement associé à l'entité de supervision du dispositif cible par l'utilisateur au moyen d'une interface homme machine. L'envoi de messages en broadcast/multicast est alors remplacé par un envoi en direct sur le proxy associé au dispositif cible B qui relaiera alors les messages vers les dispositifs qui lui sont accessibles, conformément aux étapes suivantes :

- le message de demande de code associé au composant est envoyé au proxy défini pour le dispositif B ;

- le proxy reçoit le message de demande de code: S'il possède la classe recherchée, l'entité de supervision du proxy envoie le fichier de code au dispositif cible ; Sinon, le proxy renvoie par diffusion (broadcast/multicast) le message de demande de code sur tous les réseaux qui lui sont accessibles en se désignant comme relai pour l'hôte recherché ;

- les dispositifs qui reçoivent le message de demande de code, relayé par le proxy, et qui possèdent le fichier de code recherché, répondent au proxy qui pourra alors jouer le rôle de relai. Le fichier de code trouvé pourra alors être relayé au dispositif cible par le proxy. Sinon, ces dispositifs relaient le message vers un ensemble de dispositifs, de la même manière que le dispositif cible, selon leurs capacités d'envoi par diffusion.

En variante, si le dispositif cible n'est associé à aucun proxy et ne dispose pas de capacité d'envoi par diffusion, l'entité de supervision du dispositif cible peut envoyer le message de demande de

code à tous les dispositifs qu'elle connaît comme étant ses voisins. Pour cela, l'entité de supervision peut maintenir les informations relatives aux entités de supervision avec lesquelles elle a communiqué dans une structure de données telles que un DNS (acronyme pour « Domain Name Service » signifiant littéralement Service de Nom de Domaine), stocké localement. Le DNS peut comprendre les noms et adresses de tous les dispositifs avec lesquels elle a communiqué. Lorsqu'une entité de supervision démarre, elle peut envoyer un message de démarrage (par exemple "Hello" signifiant « Bonjour ») aux entités de supervision qui sont connectés sur le même réseau, de sorte que toute nouvelle entité de supervision qui entre sur un réseau est connue et génère une entrée sur les DNS de toutes les entités de supervision couramment connectées sur ce réseau.

Le message de demande de code comprend des informations relatives au composant et peut comprendre notamment le nom de la classe de composant 302 recherché ainsi que le type du dispositif électronique A (par exemple dispositif sous Android, ordinateur personnel PC).

Si l'entité locale 6 d'un dispositif reçoit un message de demande de code et dispose du code associé au composant, elle envoie à l'entité locale 6 du dispositif cible le fichier de code associé au composant (par exemple, fichier .JAR en JAVA) vers le dispositif cible B en le relayant via les entités de supervision par lesquelles le message de demande de code lui est arrivé. Une entité de supervision 6 peut disposer du fichier de code lorsqu'elle gère un dépôt de composants ou en variante lorsque le dispositif est du même type que le dispositif source et détient le fichier recherché parce que le dispositif considéré accueille un composant 302 de la même classe que la classe recherchée. Un tel dépôt de code peut stocker, pour chaque type de dispositif, les fichiers de code associés aux composants. Le dépôt de code peut être partiel, en ce sens qu'il ne contient pas tous les codes de composants utilisés par les dispositifs, mais seulement le code des composants les plus utilisés. Pour optimiser les performances de certains dispositifs ayant des capacités de traitement plus faibles, dits dispositifs contraints, (par exemple, dispositif contraint de type téléphone), seuls certains dispositifs non-contraints 5 peuvent être associés à un dépôt de code complet ou partiel. Cela permet de limiter la surcharge de travail des dispositifs non contraints, liée à l'envoi de code en réponse aux demandes des entités de supervision.

Sinon, si l'entité locale 6 d'un dispositif récepteur du message de demande de code ne possède pas le fichier de code associé au composant, elle envoie le message de demande de code à d'autres entités de supervision 6, comme l'entité de supervision du dispositif cible B. Dans certaines formes de réalisation de l'invention, l'entité de supervision 6 peut également se désigner comme relai possible pour le retour de réponse vers le dispositif cible.

L'homme du métier comprendra que le procédé de chargement dynamique de code selon les étapes 702 à 707 peut s'appliquer de manière similaire à la création d'un composant sur un dispositif.

La figure 8 est un organigramme général illustrant le chargement dynamique de classe de composant sur le dispositif cible B, selon le cas (étapes 703 et 707 de la figure 7).

Si la classe du composant est une classe résidente sur le dispositif (800), comme par exemple une classe standard Java ou une classe incluse dans l'entité locale de supervision 6, à l'étape 801 l'appel est transmis au chargeur de classes standard de java. Le chargeur standard de la machine virtuelle JAVA permet le chargement dynamique de code par le biais de spécialisations de la classe du chargeur.

Si la classe du composant n'était pas connue (802) et a été chargé à l'étape 707 de la figure 7 par le procédé de chargement dynamique de code, un chargeur de code (encore appelé chargeur de classe) associé au fichier est créé par l'entité locale de supervision 6 et enregistré (étapes 707 et 708 de la figure 7) à l'étape 803. Il est ensuite utilisé pour charger le code (804). Le chargeur de classe associé à chaque fichier est stocké sur le dispositif B. Le rôle de ce fichier de classe est de charger dans la mémoire de la machine le code correspondant à une classe demandée. A chaque composant 302 hébergé sur le dispositif B est ainsi associé le chargeur de classes qui a servi à le créer de telle sorte que les créations futures d'objets issus de ce composant (par exemple lorsque le composant créé un objet qu'il veut envoyer par un connecteur de sortie) peuvent se faire ensuite au travers de ce même chargeur de classes. Dans les dispositifs dépendants de la machine virtuelle JAVA, la création du chargeur de classe peut comprendre la définition d'une classe spécifique (appelée Classloader) pour charger du code dans la machine virtuelle java et l'instanciation d'un objet ensuite associé au composant CM. Dans le cas particulier où le dispositif électronique A utilise un système d'exploitation de type Android, le chargeur de classe peut en outre hériter d'un autre type de classe de chargement de code appelé « DexClassLoader » car le code binaire et la façon de charger le code sur les dispositifs de type Android sont différents. Selon une caractéristique de l'invention, le lien entre le composant 302 et son chargeur de classes peut être mémorisé dans le conteneur 305 du composant 302 (sous la forme d'un objet qui peut être ajouté aux propriétés du conteneur). Cette association entre un composant 302 et un chargeur de classes permet l'accès par un composant 302 aux ressources incluses dans le fichier de code associé au composant (fichier Jar) au moyen de méthodes d'accès aux ressources appelées «getResourceAsStream » (pour recevoir les données sous forme de flux) et « getResourceAsByteArray » (pour recevoir les données sous forme de données binaires). Ces méthodes appartiennent à la classe du modèle de composant dont il hérite, appelée BCModel.

Le chargeur de classes pour un dispositif de type ordinateur personnel (PC) diffère de celui utilisé pour un dispositif utilisant le système d'exploitation Android en raison du mode différent de chargement de classes entre ces deux types de dispositifs.

Si la classe du composant 302 est comprise dans un fichier disponible localement (805), l'entité de supervision locale 6 du dispositif cible B transmet l'appel au chargeur de classes associé à ce fichier à l'étape 806. Un tel chargeur de classes a été créé conformément aux étapes 802 et 803, lors de la réception initiale de ce fichier par le dispositif. Le chargeur de classes associé au fichier

de code charge alors la classe du composant ainsi que les classes des objets en entrée et sortie du composant.

L'échange de commandes externes entre les entités de supervision 6 et de commandes internes entre chaque entité A et B et le composant CM assurent le transfert du composant de A vers B et la redirection des connecteurs de A vers B. Ces échanges permettent notamment de restituer l'état du composant dans le dispositif cible, et le démarrage à chaud du composant (le composant est mis directement à l'état actif sur le dispositif cible).

Lorsque le composant CM est migré de A vers B, ses connecteurs 303 d'entrée et de sortie sont redirigés par l'entité de supervision du dispositif cible de façon à ce qu'ils aboutissent sur B et non plus sur A.

La figure 9 montre les interactions entre des connecteurs 303 et un composant 302 auquel ils sont reliés, selon un exemple de réalisation de l'invention.

Un composant 302 peut accéder à ses flux d'entrée et de sortie par le biais de mécanismes fournis par son conteneur 305. Dans une forme de réalisation de l'invention, les données lues en entrées ou écrites en sorties d'un composant peuvent être des objets sérialisables qui peuvent être prédéfinis par le concepteur. Comme les flux peuvent transporter des données continues ou discontinues, le système de supervision 10 supporte deux types de moyens d'accès aux données, le premier moyen d'accès étant adapté aux flux continus (suite continue dans le temps, à intervalle régulier, d'échantillons de données) et le second moyen d'accès étant adapté aux flux non continus (informations délivrées de manière irrégulière).

Le premier moyen d'accès permet un accès direct aux données. Le composant 302 peut lire dans le flux de son choix par une opération bloquante jusqu'à ce qu'une donnée soit disponible : l'exécution du composant est suspendue jusqu'à ce que la donnée soit disponible, tandis que les autres composants peuvent continuer à s'exécuter. En variante, le composant 302 peut récupérer la première donnée disponible dans l'un de ses flux d'entrée par une opération qui le bloque jusqu'à ce qu'une donnée soit disponible sur l'un des flux d'entrée. Le deuxième moyen d'accès permet un accès aux données en utilisant des écouteurs. Selon ce deuxième moyen d'accès, le composant 302 met en place sur un flux un écouteur 61 d'entrée dont une méthode sera automatiquement appelée dès qu'une donnée sera présente sur ce flux. Les écouteurs peuvent être placés sur certaines entrées seulement, sur toutes les entrées ou encore sur aucune. Un écouteur d'entrée peut être supprimé à tout moment par le composant 302 qui retournera alors au premier moyen d'accès (mode d'accès direct).

L'écriture dans un flux de sortie par un composant 302 se fait par une opération d'accès direct qui ne peut être bloquante que si aucun flux n'est connecté sur la sortie correspondante.

Lorsqu'un connecteur d'entrée 303 dispose d'une nouvelle donnée, il en informe l'Unité de contrôle (UC) 40 du conteneur du composant 302 auquel il est relié (1), qui peut transmettre cette information à un gestionnaire d'écouteurs 60 du composant 302. Le gestionnaire d'écouteurs 60 est adapté pour gérer une file d'attente des écouteurs du composant à activer (2) et exécuter les méthodes appropriées de ces écouteurs (3). Le gestionnaire d'écouteurs peut sélectionner dans la file d'attente un à un les écouteurs disponible et exécuter la méthode associée. Dès que cette méthode est terminée, il peut prend ensuite le suivant dans la file d'attente et réitérer le procédé jusqu'à ce que la file d'attente soit vide. Lorsque le composant 302 doit être arrêté, par exemple parce qu'il est migré sur un autre dispositif (étape 602 de la figure 6), l'entité de supervision 6 du dispositif sur lequel s'exécute le composant peut arrêter le gestionnaire d'écouteurs 60 en utilisant les mêmes mécanismes que le composant lui-même (exceptions) pour que les données des connecteurs 303 ne soient plus traitées. Par suite, le gestionnaire d'écouteurs ne lance plus d'écouteurs pour traiter les données d'entrée. Ces données restent donc dans le connecteur 303 pour pouvoir être récupérées par le prochain composant qui sera relié à ce connecteur. En mode d'accès direct (absence d'écouteurs), l'entité de supervision du dispositif source arrête le composant à migrer en communiquant avec l'unité de contrôle du composant. L'unité de contrôle 40 du composant arrête alors les unités d'entrée 40 du composant. Les données d'entrée restent alors également dans les connecteurs 303 d'entrée pour pouvoir être réutilisées après la reconnexion des connecteurs.

Le démarrage d'un composant 302 par l'entité de supervision 6 du dispositif sur lequel s'exécute le composant n'est pas assujetti à l'existence des connecteurs 303 qui lui sont reliés. L'exécution d'un composant 302 peut se poursuivre tant qu'il ne tente pas d'accéder à un flux d'entrée ou de sortie.

Le démarrage du composant se fait en passant de l'état 50 à 52 de la figure 5, après avoir positionné les propriétés du composant aux valeurs reçues par sérialisation. Le redémarrage peut se faire sans attendre que les connecteurs aient été redirigés vers la nouvelle localisation du composant.

La figure 10 montre le diagramme d'états de l'unité d'entrée 41 d'un conteneur 305 de composant 302.

Le fonctionnement des flux d'entrée est géré par les unités d'entrée 41 du conteneur 305 du composant 302. Une unité d'entrée (UE) 41 peut être dans un état « non-créé » (état 700). Une unité d'entrée créée peut être « arrêtée » (état 701), « en marche et connectée » (état 702), ou encore « en marche et non connectée » (état 703). Lorsqu'une unité d'entrée 41 créée est arrêtée (état 701), une tentative de lecture par le composant 302 sur le flux d'entrée de cette unité d'entrée provoque une exception qui arrête le composant. Cette procédure d'exception fournie par le conteneur 305 de composants consiste à arrêter le composant 302 puis à lui faire exécuter sa méthode de destruction ("destroy") pour qu'il se termine comme prévu par le concepteur du composant. Ainsi, quand un composant 302 doit être migré (ou plus généralement arrêté), l'entité de supervision 10 du dispositif cible place toutes ses unités d'entrée et de sortie (41, 42) en mode

« arrêté » (état 700). Lorsqu'une unité d'entrée 41 est en marche (i.e. l'unité d'entrée est en cours d'exécution), elle peut être reliée à un connecteur 303 (état 702) ou non-reliée à un connecteur 303 (état 703). Lorsqu'elle n'est pas reliée à un connecteur (état 703) elle peut soit être arrêtée (état 700), soit passer en attente de lecture en cas de tentative de lecture par le composant 302 sur le flux d'entrée de l'unité d'entrée (état 704), soit passer en attente de connexion (état 705).

A chaque tentative de lecture par le composant 302 depuis l'état 701, l'unité d'entrée 41 passe à l'état d'attente de lecture 704, puis vérifie si elle est reliée à un connecteur 303 (état 706 de recherche de connexion). Si elle est reliée à un connecteur 303, l'unité d'entrée 41 tente de récupérer une donnée (état de lecture 707). Sinon, si elle n'est reliée à aucun connecteur (état 708 d'attente de disponibilité de connecteur), l'unité d'entrée 41 bloque le composant 302 jusqu'à ce qu'elle soit à nouveau connectée à un connecteur (retour à l'état 706) ou jusqu'à ce que le système de supervision 10 arrête l'unité d'entrée 41 (état 701).

Lors d'une tentative de lecture dans un connecteur 303 en entrée (état 707), après que l'unité d'entrée 41 a identifié une connexion avec ce connecteur 303 (état 706), le composant 302 est bloqué jusqu'à ce qu'une donnée soit présente sur l'entrée la reliant au connecteur 303. Pendant le blocage du composant 302, l'unité d'entrée 41 vérifie que le connecteur 303 reste présent. Si le connecteur 303 disparaît ou est déconnecté de cette entrée 41 alors que le composant 302 est bloqué, l'unité d'entrée 41 suspend la lecture en cours et attend qu'une nouvelle connexion soit réalisée (état 708 d'attente de disponibilité de connecteur). Dès qu'une telle connexion est établie, l'unité d'entrée reprend la lecture suspendue (retour à l'état 707).

Lorsqu'une donnée est présente sur l'entrée reliant l'unité d'entrée au connecteur 303, l'unité d'entrée 41 passe à l'état 709 pour tenter de récupérer des données. Si aucune donnée n'est disponible, l'unité d'entrée passe en état d'attente (état 710) jusqu'à ce qu'une donnée soit disponible (état 711). Lorsqu'une donnée est disponible, l'unité d'entrée peut soit être arrêtée (état 701) soit revenir à l'état 702 jusqu'à la prochaine tentative de lecture de donnée.

En complément, le système de supervision 10 peut à tout moment arrêter le composant 302. Un ensemble de sémaphores peuvent être utilisés de manière à bloquer un composant 302 et permettre l'exécution des autres composants pendant que l'un d'entre eux est en attente soit d'un connecteur soit d'une donnée.

Le diagramme d'état de la figure 10 s'applique de manière similaire aux unités de sortie 42. Les flux de sortie en provenance d'un composant 302 peuvent être dupliqués autant de fois que nécessaire pour être transmis à plusieurs composants qui lui sont reliés. Cette duplication est totalement transparente pour les composants 302 de sorte que l'ajout ou le retrait d'un connecteur 303 n'a pas d'incidence sur son fonctionnement. Lorsqu'il n'y a plus de flux en sortie d'une unité de sortie 42, le composant 302 est bloqué dès qu'il tente de produire une donnée sur cette sortie. Il est automatiquement relancé dès la connexion d'un nouveau connecteur puis la donnée en attente y est écrite. Comme les unités d'entrée 41, les unités de sortie 42 peuvent être arrêtées ou en

marche, connectées ou non connectées. Lorsqu'une unité de sortie est arrêtée, une tentative d'écriture par le composant 302 provoque une exception qui l'arrête à son tour.

Ce mode de fonctionnement permet de supprimer, déconnecter ou reconnecter des flux d'entrée/sortie, de manière dynamique, sans perturber le fonctionnement des composants 302, qui sont suspendus lorsqu'ils tentent d'accéder aux flux d'entrée/sortie, et relancés lorsque les flux d'entrée/sortie sont à nouveau disponibles. Cette capacité de connexion/déconnexion dynamique des flux d'entrée/sortie est particulièrement adaptée aux environnements mobiles où de telles situations peuvent se produire fréquemment.

L'unité de contrôle 40 du conteneur 305 de composant constitue le lien entre le conteneur de composant et l'entité de supervision 6. En particulier, chaque entité de supervision 6 peut communiquer avec l'unité de contrôle 40 du conteneur 305 d'un composant pour faire passer une unité d'entrée 40 ou une unité de sortie 41 du composant à l'état arrêté lorsqu'elle veut arrêter le composant. Chaque entité de supervision 6 peut en outre communiquer avec l'unité de contrôle 40 du conteneur 305 d'un composant pour recueillir des informations sur l'activité du composant.

La figure 11 représente la structure générale d'un modèle de connecteur 303 qui permet de relier deux composants 302, selon une forme de réalisation de l'invention.

Selon cette forme de réalisation, un connecteur 303 peut être encapsulé par des conteneurs 805. La fonction principale d'un connecteur 303 est de relier deux composants 302 entre eux et de faire circuler l'information entre eux. De la même manière qu'un composant 302, un connecteur 303 constitue une entité de première classe en ce qu'il peut être créé et détruit dynamiquement. Les connecteurs 303 ne sont pas limités à la mise en œuvre d'un ou de plusieurs modes de communication spécifiques (par exemple de type Client/Serveur, Pipe & Filter, etc.). Par ailleurs, un connecteur 303 peut agir sur l'information échangée entre deux composants de manière à adapter les données dynamiquement. Pour cela, chaque connecteur 303 peut comprendre un composant 802. Le composant 802 contenu dans un connecteur 303 peut non seulement faire circuler l'information mais également la modifier au passage, par exemple en cryptant ou en comprimant les données avant de les transmettre.

Comme montré sur la figure 11, un connecteur 303 peut être encapsulé dans un conteneur 805 doté d'une unité d'entrée (UE) 81, d'une unité de sortie (US) 82 et d'une unité de contrôle (UC) 80 de manière analogue au conteneur 305 d'un composant métier 302. Cependant, contrairement au conteneur 305 de composant métier 302, le conteneur 805 de connecteur 303 n'accepte qu'une seule unité entrée 81 et qu'une seule unité de sortie 82. Par ailleurs, l'unité de contrôle 80 (UC) permet au système de supervision 10 de superviser le fonctionnement du connecteur 303. L'unité d'entrée (UE) 81 et l'une unité de sortie (US) 82 peuvent être reliées à des tampons respectivement 810 et 820 pour éviter les pertes de données lors des reconfigurations. Les données sont stockées dans les tampons 810 et 820 jusqu'à ce qu'elles puissent être transférées. En outre, les tampons 810 et 820 permettent de détecter des situations pouvant nécessiter des

reconfigurations des composants 302 sur l'ensemble des dispositifs mobiles 5, par exemple par migration de certains composants 302 entre les dispositifs mobiles. Ainsi, si un tampon de sortie 820 est saturé, l'unité de contrôle 80 détecte que le composant suivant ne traite pas les données assez rapidement ou que la liaison réseau est lente. Si un tampon d'entrée 810 est saturé, l'unité de contrôle 80 détecte un dysfonctionnement du composant 802 contenu dans le connecteur 303. Par ailleurs, si les tampons 810 et 820 sont vides, cela permet à l'unité de contrôle 80 de détecter la fluidité de la circulation des données et de déclencher dynamiquement une reconfiguration des composants 302 sur l'ensemble des dispositifs 5 pour augmenter la qualité du service.

Ainsi, le composant 802 encapsulé dans le connecteur 303 assure le transfert de données entre l'unité d'entrée 81 et l'unité de sortie 82. Il peut appliquer tout type de communication orientée processus (compression, règles de priorité entre les données, agrégation des données, etc.). Le composant 802 est essentiellement prévu pour lire un échantillon de données dans l'unité d'entrée 81 et l'écrire dans l'unité de sortie 82.

Le connecteur 303 est configuré pour informer l'entité de supervision 6 du dispositif sur lequel il s'exécute 5 de l'état de la circulation des données dans l'application. Il est en outre adapté pour lever des alarmes quand des données s'accumulent dans ses tampons 810 et 820 mais également quand la circulation des données devient fluide après une accumulation dans les tampons 810 et 820. Le système de supervision 10 peut ainsi surveiller la circulation de données dans un hôte 5 ou entre deux hôtes 5 sur le réseau. Les niveaux des alarmes sont paramétrables dans le système de supervision 10.

L'entité de supervision 6 peut communiquer avec l'unité de contrôle 80 des connecteurs hébergés sur le même dispositif pour recevoir des alertes. L'unité de contrôle 80 émet de telles alarmes en fonction de l'état des tampons 810 et 820. A partir des alertes reçues, l'entité de supervision 6 peut déclencher des reconfigurations qui modifient la cartographie des composants sur l'ensemble des dispositifs de manière transparente.

Les connecteurs 303 correspondent ainsi à des flux de données qui peuvent être également encapsulés dans des conteneurs 805. Ces conteneurs 805 de connecteurs permettent au système de supervision d'effectuer des déploiements et des reconfigurations dynamiques pendant l'exécution de l'application. Les connecteurs forment eux-mêmes des composants capables de contrôler le transfert des données. Ils permettent notamment à l'entité de supervision locale de contrôler l'état de l'information qui circule entre les composants.

Selon un aspect de la présente invention, les connecteurs 303 peuvent fonctionner en mode synchrone. Ainsi, pour chaque donnée envoyée un acquittement est renvoyé. Aucune nouvelle donnée ne peut être envoyée tant que l'acquittement n'a pas été reçu. Ce mécanisme de synchronisation permet au système de supervision 10 de contrôler et/ou de mesurer la circulation des données. Ainsi, aucune donnée ne peut être accumulée hors de son « middleware » (encore

appelé « intergiciel »), par exemple dans les mémoires tampon (« buffers » en langue anglo-saxonne) des connecteurs réseau (« sockets » en langue anglo-saxonne).

Un connecteur 303 peut être utilisé pour relier deux composants 302 sur le même dispositif électronique 5 (connecteur interne). En variante, un connecteur 303 peut être utilisé pour relier deux composants 302 placés sur des dispositifs électroniques 5 différents (connecteur distribué). En raison de l'hétérogénéité des réseaux (ethernet, wifi, bluetooth, 3G, etc.) qui peuvent être utilisés sur l'ensemble des dispositifs électroniques 5 couverts par le système de supervision 10, il peut arriver que deux dispositifs 5 devant être reliées par un connecteur 303 ne puissent pas communiquer directement. Le système de supervision 10 est alors configuré pour trouver un dispositif mobile intermédiaire 5 pouvant servir de passerelle entre les deux types de réseaux.

Ainsi, le système de supervision 10 supporte trois types de connecteurs selon les définitions suivantes : des connecteurs internes, des connecteurs distribués et des connecteurs relai.

La figure 12 représente un connecteur interne 303. Le connecteur 303 est interne à un hôte 5 qui relie deux composants 302 sur le même dispositif électronique 5. Lorsque le système de supervision 10 détermine que deux composants 302 doivent être reliés, l'entité de supervision 6 locale sur le dispositif 5 crée un conteneur 805 de connecteur 303 sur le dispositif 5 et le relie aux conteneurs respectifs 305 des deux composants 302, de sorte, que l'unité d'entrée 81 du connecteur résultant 303 est reliée à l'unité de sortie 42 de l'un des composants et l'unité de sortie 82 du connecteur 303 est reliée à l'unité d'entrée 41 de l'autre composant 302.

La figure 13 représente un connecteur distribué 303 pour relier deux composants 302 localisés sur deux hôtes distincts A et B, comme par exemple deux téléphones intelligents (« smartphones ») distincts, deux ordinateurs portables (PC) distincts, ou encore un téléphone intelligent et un ordinateur portable, etc., lorsque les deux hôtes ont des moyens de communication compatibles (les deux hôtes peuvent communiquer directement par réseau). Le connecteur distribué 303 établit alors une communication par réseau 102 pour connecter les composants 302.

La figure 14 illustre la structure d'un connecteur relai 303 qui peut être utilisé pour connecter deux composants 302 placés respectivement sur deux hôtes distincts A et C qui ne peuvent pas communiquer entre eux. Une telle situation se produit lorsque le réseau de communication 120 de l'hôte A (par exemple réseau 3G) est incompatible avec le réseau 121 de l'hôte C (par exemple réseau wifi). Dans ce cas, un connecteur 303B sur un hôte relai B peut être utilisé comme relai sur le réseau (l'hôte relai doit pouvoir établir une liaison directe avec A et une autre liaison directe avec C). Un tel connecteur 303B permet de créer des ponts entre deux réseaux de types différents (distincts de routes complexes). Ainsi, pour relier deux composants 302 sur deux hôtes utilisant des réseaux différents, le système de supervision 10 crée un connecteur 303B sur un hôte relai B ayant simultanément accès aux deux réseaux 120 et 121.

L'unité d'entrée 81 du connecteur 303B sur l'hôte relai B est liée à l'unité de sortie 42 du composant 302 sur le premier hôte A et l'unité de sortie 82 du connecteur 303B sur l'hôte relai B est reliée à l'unité d'entrée 41 du composant 302 sur le second hôte C. Le connecteur relai 303 se présente ainsi en trois parties 303A, 303B et 303C, et utilise les mêmes mécanismes de connexion qu'un connecteur distribué.

Le procédé de redirection d'un connecteur est mis en œuvre par l'entité de supervision sur un dispositif source pour rediriger les connecteurs 303 qui sont reliés à un composant migré (étape 608 de la figure 6). La redirection d'un connecteur 303 peut aboutir à différentes situations selon les connexions de son entrée et de sa sortie avant la migration, comme indiqué par le tableau de la figure 15.

Ainsi, si le composant sur A était lié à un connecteur qui venait de ou allait vers A (connecteur interne entre 2 composants sur A), suite à la migration du composant sur B, le connecteur 303 sur le dispositif A est remplacé par un connecteur distribué ou un connecteur relai entre A et B (connecteur distribué ou par relai). L'entité de supervision 6 sur le dispositif A initie alors la création d'un connecteur distribué si le dispositif A et le dispositif B ont des réseaux compatibles ou d'un connecteur relai entre A et B dans le cas contraire. Le connecteur initial sur A est supprimé.

Si le composant sur A était lié à un connecteur qui venait de ou allait vers B (connecteur distribué ou relai entre le composant sur A et un composant sur B), suite à la migration du composant sur B, le connecteur 303 sur le dispositif A est remplacé par un connecteur interne sur B (entre 2 composants sur B). L'entité de supervision 6 sur le dispositif A initie alors la création d'un connecteur interne sur A.

Si le composant sur A était lié à un connecteur qui venait de ou allait vers C (connecteur distribué ou relai entre le composant sur A et un composant sur C), suite à la migration du composant sur B, le connecteur 303 sur le dispositif A est remplacé par un connecteur distribué ou relai entre B et C (entre le composant sur B et un composant sur C). L'entité de supervision 6 sur le dispositif A initie alors la création d'un connecteur distribué entre B et C si le dispositif B et le dispositif C ont des réseaux compatibles ou un connecteur relai entre B et C dans le cas contraire. Le connecteur distribué ou relai initial entre A et C est supprimé.

Ainsi la redirection de connecteurs, en réponse à la migration d'un composant entre le dispositif source et le dispositif cible, peut nécessiter, selon les cas, la création d'un connecteur interne, d'un connecteur distribué ou d'un connecteur relai, chaque type de connecteur ayant au moins une partie sur le dispositif cible, ou encore la suppression d'un connecteur existant.

La figure 16 est un organigramme illustrant les étapes mises en œuvre pour créer le connecteur distribué entre un hôte A et un hôte B par l'entité de supervision 6 sur l'hôte A, lorsque les hôtes A et B ont des moyens de communication compatibles. Quand l'entité de supervision 6 sur l'hôte A reçoit une commande de création d'un connecteur de l'hôte A vers l'hôte B (étape 90), elle

détermine une route vers B (étape 91). Si la route vers B ainsi déterminée est directe (étape 92), c'est-à-dire qu'elle ne passe pas par des hôtes intermédiaires, l'entité de supervision 6 sur l'hôte A envoie une commande à l'entité de supervision 6 sur l'hôte B pour que cette dernière crée un conteneur de connecteur 805B (étape 93). De son côté, l'entité de supervision 6 sur l'hôte A crée un conteneur de connecteur 805A (étape 94). L'homme du métier comprendra que les étapes 93 et 94 peuvent être réalisées sensiblement en même temps ou successivement. Le connecteur 303 qui en résulte est un élément en deux parties, 303A encapsulé par le conteneur 805A et 303B encapsulé par le conteneur 805B, avec un fil d'exécution client (« thread » en langue anglo-saxonne) d'un côté (sur l'hôte B) et un fil d'exécution serveur (« thread » en langue anglo-saxonne) de l'autre côté (sur l'hôte hôte A). Un mécanisme de synchronisation est démarré entre ces deux fils d'exécution afin de synchroniser les deux parties du connecteur 303 (95).

Si l'entité de supervision 6 sur l'hôte B reçoit une commande de création d'un connecteur de l'hôte A vers l'hôte B, un procédé similaire à celui de la figure 10A est mis en œuvre, en permutant les rôles de l'hôte A et de l'hôte B.

La figure 17 est un organigramme illustrant le traitement d'une commande de suppression d'un connecteur distribué sur un hôte A et un hôte B, par l'entité de supervision 6 sur un hôte A.

En réponse à la réception d'une commande de suppression d'un connecteur distribué 303 (étape 95), l'entité de supervision sur l'hôte A envoie une commande à l'entité de supervision 6 sur l'hôte B pour qu'elle supprime le conteneur 303 et le fil (« thread ») de communication (étape 96). De son côté, l'entité de supervision sur l'hôte A supprime son propre conteneur 303 et son fil (« thread ») de communication (étape 97). L'homme du métier comprendra que les étapes 96 et 97 peuvent être réalisées sensiblement en même temps ou successivement.

La figure 18 est un organigramme des étapes mise en œuvre par les entités de supervision pour relier deux composants 302 sur deux hôtes A et C ayant des moyens de communication incompatibles.

En réponse à une commande de création d'un connecteur 303 entre un hôte A et un hôte C (étape 130), l'entité de supervision 6 sur l'hôte A calcule une route de A vers C (étape 132), si la commande a été reçue par l'hôte A. Pour détecter l'incompatibilité des réseaux, l'hôte A exécutant la commande de création peut tenter d'atteindre l'autre hôte B, et s'il n'y parvient pas déterminer, qu'il n'y a pas de liaison directe possible entre les hôtes A et B. Si la route ainsi déterminée n'est pas directe (133) et passe par un hôte B pouvant servir de relai (134), l'entité de supervision locale 6 sur l'hôte A envoie une commande à l'entité de supervision locale 6 sur l'hôte relai B pour qu'elle crée un connecteur 303B de B vers C (étape 135). L'entité de supervision locale 6 sur l'hôte relai B crée un conteneur de connecteur 805B encapsulant un connecteur 303B (137). De son côté, l'entité de supervision locale 6 sur l'hôte A crée un conteneur de connecteur 805A sur l'hôte A encapsulant un connecteur 303A (134). L'entité de supervision locale 6 sur l'hôte B envoie à son

tour une commande à l'entité de supervision locale 6 sur l'hôte C pour qu'elle crée un conteneur de connecteur 805C encapsulant un connecteur 303C (étape 139). L'entité de supervision locale 6 sur l'hôte relai C crée alors un conteneur de connecteur 805C encapsulant un connecteur 303C (137).Un connecteur relai 303 en trois parties 303A, 303B et 303C est ainsi créé. Un mécanisme de synchronisation est ensuite mis en œuvre entre les trois parties 303A, 303B et 303C du connecteur 303 (étape 140) pour synchroniser ces trois parties.

Les communications assurées par les connecteurs distribués ou relai utilisent un modèle client/serveur. Lorsqu'un connecteur distribué est créé il lance une procédure de synchronisation permettant de s'assurer que les différentes parties qui le constituent soient prêtes. Au cours de cette procédure de synchronisation, les mécanismes sont mis en place pour permettre la communication ultérieure entre les parties de connecteurs distribué ou relai.

Les interactions entre les entités de supervision et les connecteurs permettent de contrôler les communications entre les composants 302 par sérialisation des objets échangés. La sérialisation permet de traduire les propriétés constituant l'état des objets dans un format en permettant le stockage ou le transport des objets sur une liaison réseau, ainsi que la reconstitution, notamment sur un autre dispositif, des propriétés de l'objet à partir des informations contenues dans ce format (dé-sérialisation). Le système de supervision 10 peut définir notamment une classe, appelée ci-après « Sample » (signifiant littéralement « échantillon »), dont héritent les classes des objets échangés au travers des connecteurs 303. Des informations peuvent être ajoutées aux données telles que leur date de création et le flux sur lequel les données ont été transportées.

Lorsque le code des classes des composants et des objets échangés n'est pas résident sur chacun des hôtes 5 et est chargé à la demande sur le dispositif cible qui accueille un composant migré par le système de supervision 10, la disponibilité des classes de composants et des objets échangés sur un dispositif donné dépend de la création du composant sur ce dispositif 5. Toutefois, pour pouvoir transporter des objets par sérialisation entre composants 302, les dispositifs 5 hébergeant les composants doivent disposer des classes de ces objets. Lorsqu'un connecteur 303 sur un dispositif 5 est relié à un composant 302, les classes des objets en entrée et sortie du composant 302 ont été chargées au préalable avec le composant 302 dans le cadre de la création ou de la migration du composant sur l'hôte. Le connecteur 303 a ainsi accès aux classes des objets.

Toutefois, comme le système de supervision 10 est réparti, il peut arriver qu'un connecteur 303 soit créé sur un dispositif donné avant le composant 302 auquel il est relié (cas i.), de sorte que le dispositif 5 ne dispose pas du code des données associées au composant 302 lors de la création du connecteur 303. En effet, comme la création d'un connecteur 303 entre un hôte A et un hôte B peut être initiée par l'hôte A ou par l'hôte B, la création de la partie 303A du connecteur se trouvant sur A peut survenir par exemple suite à une commande provenant de l'hôte B tandis que la création du composant 302 associé à ce connecteur sur l'hôte A peut être provoquée par une commande provenant d'un autre hôte C distinct de l'hôte B, par exemple parce qu'il met en œuvre

une restructuration ou reconfiguration dynamique de l'application suite à un changement de contexte. Par suite, les deux commandes reçues par l'hôte A (commande de création d'un connecteur 303A provenant de l'hôte B et commande de création d'un composant 302 provenant de l'hôte C) étant issues de deux machines différentes, l'ordre dans lequel elles parviennent à l'hôte A est imprévisible : ainsi, l'hôte B pourrait envoyer une donnée sur le connecteur 303A le reliant à l'hôte A avant que l'hôte A ne dispose effectivement du code de la classe de cette donnée.

Par ailleurs, un connecteur 303 sur un hôte 5 donné peut n'être relié à aucun composant 302 (cas ii.), comme dans le cas d'un connecteur relai incluant un connecteur 303B placé sur un hôte relai B dans le but de relier deux autres hôtes A et C ne partageant pas le même réseau (figure 12). Le code des objets transportés par le connecteur 303 n'a alors pas été préalablement chargé dynamiquement sur l'hôte où a été créé le connecteur, comme le chargement dynamique du code associé au composant est déclenché pour la création où la migration d'un composant sur l'hôte. En l'absence du code des données, l'hôte ne peut relayer les données.

Pour remédier à la situation, le système de supervision 10 encapsule les classes des objets transmis par dans une classe d'encapsulation spécifique (désignée ci-après par « EncapsulatedSample », expression signifiant littéralement « Echantillon encapsulé ») de sorte que les connecteurs 303 qui n'ont pas accès aux classes des données ne manipulent que des objets de cette classe d'encapsulation « EncapsulatedSample ». Les données transmises sont ainsi encapsulées dans un conteneur prévu à cet effet de façon à pouvoir les transporter et les recevoir même en l'absence du code des classes de ces données. Ces données pourront être désencapsulées et utilisées lorsque leur code sera enfin disponible sur un dispositif où elles sont transmises.

L'encapsulation des données échangées par le système de supervision 10 permet de transporter sur les connecteurs 303 des informations dont le code n'est pas disponible, ce que la sérialisation simple ne permet pas de faire. Par cette encapsulation, le composant émetteur et le composant destinataire final des données échangées sont ainsi capables de les traiter puisqu'ils disposent du code des classes de ces données qui a été dynamiquement chargé en même temps que le code du composant lui même.

Ainsi, lorsque le connecteur 303 est le connecteur 303B servant de relai entre les deux autres parties d'un connecteur relai (cas ii.), le rôle de ce connecteur intermédiaire 303B peut se limiter à transmettre les données encapsulées. Dans le cas d'un connecteur 303 lié à un composant 302 mais créé avant le composant 302 (cas I.), le connecteur 303 transmet la donnée encapsulée dans la classe « EncapsulatedSample » à l'unité d'entrée 81 du conteneur de connecteur 805 qui peut extraire de la classe d'encapsulation l'objet contenu puisque le code de la classe de cet objet a été chargé avec le composant 302 qui la traite.

La figure 19 représente la structure générale de communication permettant aux entités de communiquer entre elles.

Chaque entité de supervision 6 peut comprendre une ou plusieurs files d'attente 101 pour stocker les messages entrants ou sortants. Ces files d'attente permettent aux différents services de chaque entité de supervision locale 6 et aux connecteurs 303 des applications d'utiliser le réseau en concurrence. Lors de l'envoi d'un message par une entité de supervision locale 6, le message peut ainsi être mis en attente dans une file d'attente jusqu'à ce que le réseau soit disponible et/ou jusqu'à ce que le message puisse être réellement envoyé. Du point de vue du service ou du connecteur qui est à l'origine de l'envoi du message, l'envoi est considéré comme fait, mais en réalité l'envoi peut être différé.

Chaque entité de supervision 6 comprend en outre au moins un émetteur 102 pour transmettre les messages placés dans les files d'attente 101 à un client d'envoi 103 correspondant au réseau approprié en fonction de son destinataire. Si ce client 103 ne peut pas atteindre le destinataire identifié, l'émetteur 102 peut faire appel à l'unité de routage 15 pour qu'elle détermine si un autre dispositif 5 peut servir de relai. Le message est alors envoyé à ce dispositif relai qui le transmettra au destinataire.

Chaque entité de supervision comprend en outre un serveur de réception 105 pour la réception des messages provenant d'autres entités de supervision 6. Le message reçu est placé dans une boîte à lettres 106 avant d'être délivré au service de l'entité de supervision auquel il est destiné. Un mutex 107 peut être utilisé pour éviter que deux requêtes reçues soient en concurrence. Si le dispositif 5 récepteur ne correspond pas au destinataire du message, le message est immédiatement renvoyé de façon à assurer la fonction de relai permettant les passerelles entre types de réseaux différents.

Chaque entité de supervision 6 fonctionnant sur un dispositif électronique 5 qui dispose d'une adresse publique peut lancer un serveur supplémentaire de proxy. Sur un dispositif 5 configuré pour accéder à un réseau par satellite, l'adresse de l'un de ces proxys peut être spécifiée au préalable à l'entité de supervision du dispositif via une interface. Le dispositif 5 pourra alors établir une connexion avec ce proxy qu'il conservera ouverte afin de pouvoir recevoir des messages et des données. Le dispositif 5 peut lancer également un client lié à ce proxy qui sera alors choisi par son émetteur de messages 102 pour tous les messages envoyés par son entité de supervision locale 6.

En complément, pour éviter les déconnexions qui peuvent être provoquées par les fournisseurs d'accès, lors de l'établissement d'une connexion avec le proxy, le client d'envoi 103 peut envoyer, à intervalles réguliers un message de test appelé « PING » (également désigné par l'acronyme « Packet InterNet Groper ») destiné à maintenir cette connexion ouverte. Cet échange de messages de test permet également au dispositif 5 et au proxy, de détecter les pertes de connexion (liée à la mobilité). En outre, des écouteurs peuvent être utilisés (tel que par exemple l'écouteur de diffusion appelé « BroadcastReceiver » pour les dispositifs de type Android) pour permettre aux dispositifs mobiles 5 de basculer automatiquement d'un mode de fonctionnement

par proxy au mode de fonctionnement normal sans proxy, lors d'un changement de type de connexion.

L'unité de routage 15 permet au système de supervision 10 de supporter tout type de réseau de communication entre les dispositifs mobiles 5. En particulier, elle permet de relayer les messages lorsque deux dispositifs 5 dépendant de réseaux de communication différents ne peuvent pas établir de communication directe entre eux (par exemple, lors de la création d'un connecteur distribué sur deux hôtes ayant des moyens de communication incompatibles). Cela permet notamment d'établir à tout moment une connexion entre deux entités de supervision locales 6.

L'unité de routage 15 peut être interrogée par l'émetteur 102 de l'entité de supervision 6 pour déterminer un relai pour une communication avec l'entité de supervision hébergée sur un autre dispositif, lorsque cela est nécessaire. Elle peut être également interrogée par l'entité de supervision pour déterminer le dispositif qui doit accueillir un connecteur relai lorsque deux composants doivent être reliés alors qu'ils ne disposent pas de possibilité de lien direct. Ainsi quand l'entité de supervision sur l'hôte A reçoit une commande pour créer un connecteur avec l'hôte B, l'hôte A interroge l'unité de routage 15 pour trouver une route vers B. Cette route peut être directe ou indirecte, dans ce dernier cas un hôte pouvant servir de relai est identifié et un connecteur avec relai est créé.

Dans une forme de réalisation de l'invention, la recherche de routes utilise un mécanisme de type « PING » et des messages de diffusion (« broadcast », « multicast »), ou en point à point vers les hôtes voisins. Ainsi si un dispositif A cherche une route pour atteindre un dispositif B, le mécanisme est le suivant : le dispositif A tente tout d'abord d'atteindre directement le dispositif B par envoi d'un message de type PING. Si le dispositif B répond au message PING, la liaison est directe et la recherche de route est terminée. Dans le cas contraire, le dispositif A envoie à tous ses voisins un message de recherche du dispositif B. Chacun de ses voisins tente alors de joindre le dispositif B par un message de PING. Les dispositifs qui parviennent à joindre le dispositif B indiquent au dispositif A qu'ils peuvent servir de relai pour le dispositif B. Ceux qui n'y parviennent pas diffusent à leur tour cette demande à leurs propres voisins. Le dispositif A peut recevoir plusieurs réponses. Dans ce cas, il peut choisir comme route celle qui correspond à la première réponse reçue et qui représente la route la plus rapide.

La figure 20 illustre le procédé de synchronisation mis en œuvre pour synchroniser les différentes parties 303A et 303B d'un connecteur distribué 303 sur un hôte A et un hôte B.

Lorsqu'un connecteur 303A est créé sur L'hôte A, l'entité de supervision locale 6 sur l'hôte A envoie un message 151 de synchronisation « SYNC » qui est transmis sur le réseau vers l'entité de supervision 6 sur l'hôte B sur lequel se trouve l'autre extrémité du connecteur 303B. Un sémaphore d'exclusion mutuelle désigné par la référence 152 peut être utilisé pour gérer l'accès concurrent à la file d'attente des messages de synchronisation 101 utilisée par les clients d'envoi 103 de l'entité de supervision 6 au niveau de l'hôte A.

En réponse à la création de l'autre extrémité du connecteur 303B sur l'hôte B, l'entité de supervision 6 sur l'hôte B répond au message de synchronisation par un message d'acquittement « SYNC ACK » désigné par la référence 154 qui est transmis à l'hôte A.

Par ailleurs, en réponse à la réception du message de synchronisation, l'entité de supervision locale 6 sur l'hôte B peut créer également une interface logicielle de réception (ou « socket » de réception) 162 pour recevoir les données, une boîte aux lettres 106, et une interface logicielle d'acquittement (ou « socket » d'acquittement) 163 pour les acquittements 164. Ainsi, lorsque le connecteur 303B est créé sur l'hôte B, il peut trouver dans la boîte aux lettres 106 le message de synchronisation envoyé par l'autre extrémité de ce connecteur 303B et y répondre par un message d'acquittement « SYNC ACK » 161. Dès lors, les données reçues sont placées dans la boîte à lettres 106 qui ne contient qu'un seul élément. Si le connecteur 303 récupère la donnée dans la boîte aux lettres 106, la boîte aux lettres 106 envoie un message d'acquittement.

La réception du message d'acquittement « SYNC ACK » par l'entité de supervision sur l'hôte A peut provoquer en outre la création d'une interface logicielle (ou « socket » en langue anglo-saxonne) d'envoi de données 155 et d'une interface logicielle (ou « socket ») de réception d'acquittement 156 sur l'hôte A. Un sémaphore 157 est mis en place pour éviter qu'une nouvelle donnée ne puisse être émise avant que l'acquittement pour la précédente donnée n'ait été reçu. Lorsqu'une donnée est envoyée par un connecteur 303, le sémaphore de synchronisation est fermé jusqu'à ce qu'un acquittement soit reçu. Ce mécanisme assure le fonctionnement synchrone des connecteurs 303 en mettant en œuvre un contrôle permettant d'assurer que les connecteurs n'envoient pas plus de données que n'en récupère l'autre extrémité.

Le mécanisme de synchronisation des parties de connecteurs s'applique de la même manière, d'une part entre une extrémité d'un connecteur relai placé sur un hôte A et la partie centrale du connecteur relai placé sur un hôte B, et d'autre part entre la partie centrale du connecteur relai placé sur l'hôte et l'autre extrémité du connecteur relai placé sur un hôte C.

Le mécanisme de synchronisation selon les formes de réalisation de l'invention permet notamment une communication synchrone dans chaque connecteur 303. Ainsi, chaque donnée envoyée par l'entité de supervision hébergeant une partie d'un connecteur distribué ou relai provoque un acquittement de l'entité de supervision réceptrice hébergeant une autre partie du connecteur distribué ou relai. Une nouvelle donnée ne peut être envoyée via le connecteur que si l'acquittement pour la précédente donnée a été reçu de l'entité de supervision réceptrice hébergeant l'autre partie du connecteur distribué ou relai. Un connecteur correspond ainsi à deux liaisons physiques : une liaison pour les données qui circulent dans un sens et une autre liaison pour les acquittements qui circulent dans l'autre sens.

Le système de supervision 10 permet ainsi de migrer des composants en assurant la redirection des connecteurs, de manière transparente, pendant que l'application est en cours de fonctionnement.

Ce procédé de migration s'appuie sur une coopération dynamique et transparente entre les entités de supervision 6 et des échanges internes entre les entités de supervision 6 impliquées dans la migration et le conteneur du composant 302 qui fait l'objet de la migration. En particulier, une telle coopération est mise en place entre les entités de supervision des différents dispositifs impliqués dans la migration, qui peuvent comprendre notamment :

- L'entité de supervision du dispositif source où se trouvait le composant ;
- L'entité de supervision du dispositif cible vers lequel est migré le composant ;
- Les entités de supervision des dispositifs accueillant au moins un connecteur relié à ce composant ;
  - Le cas échéant, les entités de supervision des dispositifs servant de relai pour un connecteur relié au composant faisant l'objet de la migration ;
  - Les entités de supervision des dispositifs pouvant transmettre au dispositif destinataire du composant migré le code associé au composant.

Ces communications permettent d'assurer la migration du composant et la reconnexion de des entrées et sorties du composant faisant l'objet de la migration.

La figure 21 montre un exemple d'architecture de noyau pour chaque entité locale de supervision 6 pour le contrôle de la communication entre composant. Chaque entité locale de supervision 6 peut comprendre :

- Un enregistreur de services 2 qui permet à l'entité locale de supervision 6 d'accéder à un ensemble de services offerts par le système de supervision 10 ;

- une couche de communication indépendante du réseau 24 et une couche de communication dépendante du réseau 25 : ces couches permettent aux entités locales de supervision 6 hébergées sur des dispositifs mobiles respectifs de communiquer entre elles et servent de support aux connecteurs entre composants ; la couche de communication indépendante du réseau 24 fournit des mécanismes (files d'attentes, sémaphores, etc) qui peuvent être utilisés par l'entité de supervision et les connecteurs pour envoyer et recevoir des objets et la couche de communication dépendante du réseau 25 fournit notamment des mécanismes pour implémenter les communications de réseau pour l'entité de supervision et pour les connecteurs.

L'entité de supervision locale 6 peut comprendre en outre les éléments 22 suivant utilisés pour la supervision des applications :

- Un module de supervision 223 (encore appelé « superviseur ») pour exécuter les commande de déploiement ou de reconfiguration en créant, supprimant, migrant des composants et/ou en créant,

supprimant, dupliquant, redirigeant des connecteurs exécuter les commandes de création/suppression/migration/connexion/déconnexion de composants ;

- un gestionnaire de contexte 222 pour contrôler le contexte des applications, notamment à partir de capteurs du dispositif 23 ; il reçoit notamment des informations sur l'état de l'application en cours d'exécution, en particulier des informations relatives aux composants, aux connecteurs liant les composants et aux dispositifs hôtes 5 ;

- les conteneurs 805 de connecteurs 303 ;

- un gestionnaire de classes de composant 226 pour gérer dynamiquement les classes chargées ; il crée en outre des chargeurs de classes pour chaque fichier de code téléchargé pour un nouveau composant accueilli sur l'hôte.

- les conteneurs 305 de composants 302 ; et

- un gestionnaire de modules 227 qui peuvent être des modules d'extension (ou plugin) pour contrôler un ensemble modules 21. Le gestionnaire de modules 227 est adapté pour lancer ou arrêter des modules 21 de l'entité de supervision. Il peut utiliser un fichier de description qui indique les plugins qui sont automatiquement exécutés lorsque l'entité de supervision s'arrête. Des modules 21 peuvent être ajoutés ou supprimés en cours d'exécution.

Les modules 21 peuvent comprendre une fonction de chargement de code 210 pour charger dynamiquement le code des classes correspondant aux composants 302 et aux objets échangés entre composants.

Les modules 21 peuvent en outre comprendre :

- un module pour applications (encore appelée « plugins pour application ») 21 qui permet aux composants d'accéder à des ressources gérées par l'entité de supervision 6. Ces ressources peuvent comprendre des ressources classiques (par exemple, textes, images, etc.) ou encore des ressources spécifiques au matériel (telles que des capteurs 211, un système GPS (acronyme pour « Global Positioning System », ou encore un système SMS (acronyme pour Short Messaging System », etc.)) ; cette unité permet notamment aux composants d'envoyer des commandes aux entités de supervision locales ou distantes et de recevoir des réponses.

- L'unité de routage 15 pour le calcul de routes (encore appelée service de routage) ; et

- Le DNS local 212 (DNS est l'acronyme pour « Domain Name System » signifiant System de Nom de Domaine).

L'enregistreur de services 2 peut fonctionner de manière similaire à l'application JAVA appelée «RMI registry » mais en mode local, c'est-à-dire en faisant uniquement référence aux services

hébergés sur l'entité de supervision locale 6 au dispositif électronique 5. Les services de l'entité de supervision locale 6 y sont enregistrés. Par exemple, lors de la création d'un connecteur distribué, des commandes sont envoyées vers les autres entités locales de supervision 6 en interrogeant l'enregistreur de services 2 pour obtenir le service chargé des couches de communications par réseau 24 et 25. De la même façon, chaque conteneur 805 de connecteur 303 peut être enregistré comme un service qui sera pourra être utilisé par l'entité locale de supervision 6 pour le contrôler et permettre sa découverte dynamique par les conteneurs 305 de composant 302 pour établir leur connexion aux flux d'entrée ou de sortie. Par ailleurs, chaque conteneur 305 de composant 302 peut enregistrer son unité de contrôle 40 comme un service permettant à l'entité locale de supervision de contrôler les différentes phases du cycle de vie d'un composant.

Les composants 302 peuvent accéder aux services du système de supervision 302 par le biais de l'enregistreur de services 2, tels que les services 211. Les autres services sont accessibles par l'entité de supervision locale 6.

Le module de supervision 223 reçoit et exécute des commandes provenant des autres entités locales de supervision 6 hébergées sur les autres dispositifs mobiles 5, des composants 302 ou d'un module de décision.

Ces commandes peuvent inclure des commandes relatives aux composants 302 pouvant comprendre des commandes de création, suppression, migration, connexion, déconnexion et duplication des flux de sortie des composants. Ces commandes peuvent comprendre :

-Une commande de création de composant prenant comme paramètres une liste d'entrée et de sorties pouvant être vides ou marquée pour être utilisée ultérieurement ;

- Une commande de suppression d'un composant donné ;

- Une commande d'envoi d'un composant vers une destination ;

- Une commande de déconnexion d'une entrée d'un composant ;

- Une commande de déconnexion d'une sortie d'un composant ;

- Une commande de reconnexion d'une entrée d'un composant ; et

-Une commande de duplication d'une sortie d'un composant.

Les commandes peuvent en outre inclure des commandes relatives aux connecteurs 303, comme des commandes de création de connecteur, de suppression de connecteurs et de redirection de connecteurs. Les commandes peuvent également comprendre des commandes relatives au contexte permettant de récupérer les états de l'hôte 5, des conteneurs 224 de connecteurs 303,

des conteneurs 305 de composants, et de la qualité de service indiquée par un composant. De telles commandes relatives au contexte incluent :

- une commande qui retourne un objet contenant le contexte d'un hôte, c'est-à-dire l'occupation mémoire, l'état de la batterie, la charge CPU, le débit réseau moyen en entrée et en sortie sur la dernière seconde.

- une commande qui retourne un objet contenant le contexte d'un conteneur. Si le nom désigne un conteneur 303 de composant, cette commande retourne les noms des connecteurs reliés en entrée et en sortie. Si le nom désigne un conteneur de connecteur 303, cette commande retourne les adresses des hôtes hébergeant l'entrée et la sortie de ce connecteur.

- une commande qui retourne un objet contenant la Qualité de Service d'un conteneur. Si le nom désigne un conteneur de composant 302, cette commande retourne la valeur indiquée par le composant. Si le nom désigne un conteneur de connecteur 303, cette commande retourne le taux de remplissage des tampons d'entrées et de sortie du connecteur 303 ainsi que le débit moyen depuis la création.

Les entités locales de supervision 6 peuvent exécuter un ensemble de méthodes qui doivent être surchargées, par exemple :

- Une méthode qui renvoie la qualité de service (QdS) offerte par le composant 302,

- La méthode appelée « run_BC() » qui est exécutée pour faire passer un composant 302 à l'état actif 52 (figure 5). Dans certaines forme de réalisation de l'invention, la méthode « Run_BC » ne se termine jamais (flux de données) et comporte pour cela une boucle du type « tant que » (while(isRunning() en langage de programmation). Si un composant 302 souhaite arrêter son exécution, de façon à ce que l'entité locale 6 puisse le migrer ou l'arrêter, une méthode appelée « idle()» peut être appelée.

Les entités de supervision locales 6 peuvent en outre exécuter des méthodes pouvant être surchargées, notamment :

- La méthode d'initialisation appelée « init() », pour initialiser un composant 302 Les initialisations des propriétés d'un composant 302 peuvent être faites dans la méthode « init » ou au début de la méthode « run_BC » (avant la boucle). La méthode d'initialisation « init » est exécutée lors du premier lancement du composant métier 302. Toutefois, elle n'est pas redémarrée après une migration. Les initialisations faites dans la méthode « init » concernent par conséquent les propriétés qui seront sérialisées lors d'une migration. Ainsi, la création d'une interface, l'accès à des dispositifs locaux ou la mise en place d'écouteurs d'événements sur certains flux d'entrée peuvent être faites au début de la méthode « run_BC » pour être prises en compte lors d'une migration.

- La méthode de destruction appelée « destroy()» qui est exécutée lors de l'arrêt définitif du composant et également avant une migration.

Les méthodes suivantes sont héritées de la classe des modèles de composants, appelée «BCModel», et peuvent être utilisables par chaque composant 302. Elles comprennent des méthodes relatives à l'état du composant 302 incluant :

- une méthode d'arrêt de composant appelée « idle() » qui arrête le composant mais ne le supprime pas ;

- une méthode qui indique si le composant 302 peut continuer à s'exécuter, appelée « isRunning() » et de type booléen. Dans le cas où le composant doit être arrêté, cette méthode peut lever l'exception de classe appelée « StopBCException ».

- une méthode indiquant l'état de migration du composant 302, de type booléen. Cette méthode peut par exemple renvoyer la valeur « VRAI » si le composant 302 a été migré.

Les méthodes « idle()» et «isRunning()» sont de préférence bloquantes : si elles sont appelées par un autre fil d'exécution (thread en langue anglo saxonne), elles ne s'exécutent pas et affichent une erreur.

Les méthodes héritées de la classe des modèles de composants « BCModel » et utilisables par chaque composant 302 peuvent comprendre également des méthodes relatives à l'environnement du composant telles que:

- Une méthode qui renvoie le nom du composant ;

- Une méthode qui renvoie le nombre d'entrées du composant 302 ;

- Une méthode qui renvoie le nombre d'entrées actuellement connectées du Composant 302;

- Une méthode d'indication de connexion d'entrée pour indiquer si l'entrée désignée par le paramètre est actuellement connectée à un connecteur ;

- Une méthode qui renvoie le nombre de sorties du composant 302 ;

- Une méthode qui renvoie le nombre de sorties actuellement connectées du composant 302 ;

- Une méthode d'indication de connexion de sortie pour indiquer si la sortie désignée par le paramètre est actuellement connectée à au moins un connecteur.

Les méthodes héritées de la classe des modèles de composants « BCModel » et utilisables par chaque composant 302 peuvent comprendre aussi des Méthodes relatives aux entrées telles que :

- Une méthode pour créer un filtre de données d'une classe spécifiée, sur une entrée du composant 302;

- Une méthode pour supprimer le filtre de données sur une entrée du composant 302 qui reçoit en paramètre un numéro d'entrée;

- Une méthode de lecture des données sur une entrée du composant;

- Une méthode de lecture des données d'une classe sur une entrée du composant;

- Une méthode de lecture de la première donnée disponible sur l'une des entrées du composant 302 ;

- Une méthode qui indique la disponibilité de données sur une entrée du composant 302;

- Une méthode pour ajouter un écouteur en entrée d'un composant 302;

- Une méthode de suppression d'écouteurs.

D'autres méthodes utilisables par les composants peuvent être relatives aux sorties des composants 302 et comprendre une méthode pour écrire sur une sortie du composant 302.

D'autres méthodes pouvant être appelées par les composants peuvent être relatives aux événements (interfaces ou écouteurs d'entrée), comme une méthode d'attente d'un événement de composant ou une méthode d'envoi d'un événement au composant.

Les composants 302 peuvent également appeler des méthodes relatives aux ressources contenues dans le fichier de code associé au composant (fichier jar). Les ressources peuvent être placées dans un sous-répertoire du répertoire contenant le composant. Le composant 302 peut y accéder en utilisant des méthodes appelées « getResourceAsByteArray » (récupération de la ressource sous forme binaire) ou « getResourceAsStream » (récupération d'un flux de lecture de la ressource).

Selon une autre caractéristique de l'invention, chaque entité locale de supervision 6 peut maintenir des informations relatives à toutes les entités locales de supervision 6 avec lesquelles elle est entrée en communication. En particulier, ces informations peuvent être enregistrées dans le DNS local 212. Pour chaque autre entité locale de supervision 6 identifiée par le DNS 212, le DNS 212 peut stocker les informations suivantes :

- un identifiant unique associé à l'entité locale de supervision 6 ;

- La liste des adresses connues de cette entité locale de supervision 6 sur chacun des réseaux auxquels elle accède ;

- Le Décalage d'horloge avec cette entité locale de supervision 6 et l'erreur maximale de mesure de ce décalage.

Lors de chaque réception de message (par exemple, message de recherche de classe) par l'entité locale de supervision locale 6 en provenance d'une entité locale de supervision 6 hébergée sur un autre dispositif, le DNS 212 enregistre l'entité de supervision 6 émettrice et son adresse. Lors d'échanges de messages de type PING ou de recherche de routes, à la réception de la réponse, le DNS 212 calcule le décalage d'horloges (ces messages contiennent l'heure d'émission extraite de l'horloge locale de l'entité de supervision 6 émettrice). Le temps d'échange de ces messages permet en outre de déterminer une borne maximale d'erreur de la mesure de ce décalage.

Chaque route indirecte trouvée provoque l'ajout de l'entité de supervision 6 servant de relai et de celle à laquelle aboutit la route.

Par ailleurs, à intervalles réguliers, les entités de supervision 6 peuvent échanger les contenus de leurs DNS 212. Les informations ainsi reçues peuvent être utilisées pour mettre à jour chaque DNS local, notamment pour ajouter des entités de supervision 6 qui n'y étaient pas enregistrées, ou pour compléter les listes d'adresses des entités de supervision 6.

Les décalages d'horloges reçus permettent de calculer de nouvelles valeurs, par exemple :

- Décalage entre hôte A et hôte B = décalage entre A et C + décalage entre C et B, et

- Erreur sur le décalage entre hôte A et hôte B = Erreur sur le décalage entre A et C + Erreur sur le décalage entre C et B.

Ces valeurs peuvent ainsi compléter le DNS local 212 ou remplacer les valeurs locales lorsque l'erreur calculée est inférieure à celle couramment connue localement.

Les décalages d'horloges du DNS peuvent être utilisés pour la gestion des connexions, notamment pour dater les objets échangés en heure locale. En effet, les données envoyées sont automatiquement datées à leur création et cette date est ajustée par les connecteurs 303 à la réception. Lorsque le décalage d'horloge avec l'hôte émetteur n'est pas connu, le connecteur 303 date la donnée par l'heure locale de sa réception.

Pour tenir compte de la mobilité des dispositifs 5, les enregistrements du DNS 212 ont de préférence une durée de vie limitée et sont supprimés à leur terme. Une valeur maximale peut être affectée à cette durée de vie lors de chaque communication réussie, puis réajustée à partir des durées de vie des DNS reçus des autres entités de supervision 6 (la valeur maximale peut alors être conservée). Ainsi une entité de supervision 6 qui disparaît ou perd toute connexion pourra être supprimée de tous les DNS. De même, une entité de supervision 6 qui ne communique avec aucune autre entité de supervision (par exemple, aucun message n'a été échangé avec d'autres entités de supervision) pourra être supprimée de tous les DNS, et réapparaître dans les DNS dès

que l'entité de supervision correspondante communiquera à nouveau avec d'autres entités de supervision.

Les inventeurs ont effectué un certain nombre de mesures relatives au système de supervision 10, en particulier sur la complexité, le temps d'exécution des commandes, les temps de transfert d'informations dans les connecteurs et le temps de déploiement d'une application.

La plupart des exécutions de commandes supportées par le système de supervision induisent des temps d'attente faibles (réponse par réseau d'une autre entité de supervision, recherche de route ...). Une reconfiguration est, généralement constituée de plusieurs commandes (ajout/suppression de composants et/ou de connecteurs par exemple). Ces temps d'attente sont optimisés par le système de supervision 10 en permettant l'exécution parallèle de ces commandes. Aussi, le temps d'exécution d'une reconfiguration est inférieur à la somme des temps d'exécution de chacune des commandes prises indépendamment.

D'après les mesures effectuées, les temps de reconfiguration sont suffisamment faibles pour permettre au système de supervision d'adapter rapidement une application à un changement de contexte. Le passage à l'échelle (nombre plus important de dispositifs impliqués dans la reconfiguration) ne modifie pas la situation significativement puisque chaque entité de supervision sur un dispositif peut exécuter ses commandes en parallèle des autres.

Le système de supervision 10 selon l'invention permet ainsi d'exécuter des composants formant tout ou partie d'une application sur différents dispositifs 5, pouvant être mobiles, de déplacer certains composants entre des dispositifs de manière transparente pour les composants liés au composant déplacé, tout en assurant le redémarrage à chaud des composants déplacés.

Le procédé de migration selon les formes de réalisation de l'invention présente permet notamment de réaliser des économies d'énergie, d'utiliser des ressources délocalisées et d'optimiser l'exécution des applications. Il peut être en outre déclenché dynamiquement par le système de supervision 10 pour répondre à des changements de contexte et/ou d'état des ressources (par exemple, baisse ou augmentation de la bande passante). Le procédé de migration selon l'invention permet également à un utilisateur de poursuivre une activité en cours sur un autre dispositif, en la redémarrant dans l'état où elle était sur le dispositif source.

L'invention n'est pas limitée aux modes de réalisation décrits ci-avant à titre d'exemple non limitatif. Elle englobe toutes les variantes de réalisation qui pourront être envisagées par l'homme du métier. En particulier, l'invention n'est pas limitée à l'architecture des entités de supervision représentée sur la figure 21. Elle n'est pas non plus limitée à l'agencement particulier d'éléments de communication des figures 19 et 20. Par ailleurs, les entités de supervision peuvent être paramétrées de différentes manières, par l'utilisateur. Ainsi, les entités de supervision peuvent utiliser une liste de composants refusés par l'utilisateur, qui peut être paramétrée à travers une interface de configuration de l'entité de supervision, pour désigner des composants qui ne doivent

pas être installés sur le dispositif (par exemple un composant devant utiliser un système de localisation GPS ne sera pas installé par l'entité de supervision si l'utilisateur a spécifié qu'il refusait d'être localisé). Les entités de supervision peuvent en outre être configurées pour gérer l'achat de composants.

L'homme du métier comprendra par ailleurs que les entités de supervision 6 selon les formes de réalisation de l'invention peuvent être mises en œuvre de diverses façons par matériel (« hardware »), logiciel, ou une combinaison de matériel et de logiciels.

En particulier, les éléments de chaque entité de supervision peuvent être combinés ou séparés en sous-éléments pour mettre en œuvre l'invention. En outre, ils peuvent être mis en œuvre sous la forme de programmes d'ordinateur exécutés par un processeur. Un programme d'ordinateur est un ensemble d'instructions qui peuvent être utilisées, directement ou indirectement, par un ordinateur.

Un programme d'ordinateur peut être écrit dans n'importe quel langage de programmation, y compris les langages compilés ou interprétés, et il peut être déployé sous n'importe quelle forme dans l'environnement informatique choisi.

Revendications

1. Système de supervision d'applications s'exécutant sur un ensemble de dispositifs électroniques (5) reliés entre eux par un ou plusieurs réseaux, caractérisé en ce que chaque dispositif (5) comprend une entité de supervision locale (6), les entités de supervision coopérant entre elles pour contrôler les applications s'exécutant sur les dispositifs électroniques (5), chaque application comprenant un ensemble de composants applicatifs, chaque composant applicatif (302) étant encapsulé dans un conteneur (305), et les composants étant reliés entre eux par des connecteurs (303), et en ce que l'entité de supervision d'un dispositif donné, dit dispositif source, est configurée pour exécuter les étapes suivantes, en réponse à la réception d'une commande de migration d'un composant vers un dispositif cible :
   - arrêter le composant sur le dispositif source, l'arrêt du composant interrompant l'arrivée de données dans les connecteurs d'entrée du composant,
   -sérialiser et encapsuler les propriétés du composant dans un conteneur de l'entité de supervision,
   -envoyer un message de demande migration à l'entité de supervision du dispositif cible, ledit message comprenant le composant sérialisé et encapsulé, et
   - rediriger les connecteurs (303) du composant en fonction de l'état des connexions de chaque connecteur sur le dispositif source.

2. Système de supervision selon la revendication 1, caractérisé en ce que, en réponse au message de demande de migration du composant (302), l'entité de supervision (6) du dispositif cible est configurée pour déterminer si le code exécutable associé au composant est disponible sur le dispositif cible.

3. Système de supervision selon la revendication 2, caractérisé en ce que l'entité de supervision (6) du dispositif cible (5) est apte à charger le code exécutable associé au composant pour désencapsuler et dé-sérialiser les propriétés du composant en utilisant un chargeur de code, et démarrer le composant, si le code associé au composant est disponible sur le dispositif cible.

4. Système de supervision selon la revendication 2, caractérisé en ce que l'entité de supervision (6) du dispositif cible (5) est apte à envoyer un message de demande du code associé au composant vers les entités de supervision d'un ensemble de dispositifs, si le code associé au composant n'est pas disponible sur le dispositif cible, et en ce que l'entité de supervision du dispositif cible est apte à charger le code associé au composant pour désencapsuler et dé-sérialiser les propriétés du composant en utilisant un chargeur de code, et démarrer le composant, en réponse à la réception dudit code depuis l'une au moins des entités de supervision (6) dudit ensemble d'entités de supervision.

5. Système de supervision selon la revendication 4, caractérisé en ce que ledit ensemble d'entités de supervision comprend les entités de supervision des dispositifs voisins accessibles par diffusion si le dispositif cible supporte l'envoi par diffusion de message.

6. Système de supervision selon la revendication 4, caractérisé en ce que le message de demande de code est envoyé à un proxy prédéfini si le dispositif cible ne supporte l'envoi de message par diffusion, et en ce que ledit proxy est apte à relayer ledit message de demande de code par diffusion, ledit ensemble d'entités de supervision comprenant les entités de supervision des dispositifs auxquels est diffusé le message relayé par le proxy.

7. Système de supervision selon la revendication 4, caractérisé en ce que l'entité de supervision (6) du dispositif cible maintient un service de nom de domaine (224) pour mémoriser les informations relatives aux entités de supervision avec lesquelles elle communique, et en ce que ledit ensemble d'entité de supervision est déterminé à partir des informations maintenues dans le service de nom de domaine, si le dispositif cible ne supporte l'envoi de message par diffusion.

8. Système de supervision selon l'une des revendications 3 à 7, caractérisé en ce que le code associé au composant comprend un ensemble de classes, et en ce que ledit chargeur de code est un chargeur de classes associé au composant et créé localement sur le dispositif en réponse à la création de la première instance du composant.

9. Système de supervision selon l'une des revendications 3 à 8, caractérisé en ce que le code associé au composant est un fichier de code de type JAR.

10. Système de supervision selon l'une des revendications 8 et 9, caractérisé en ce que le lien entre le chargeur de classe et le composant (302) est enregistré dans le conteneur (305) du composant.

11. Système de supervision selon l'une des revendications précédentes, caractérisé en ce que la redirection des connecteurs par l'entité de supervision sur le dispositif source est mise en œuvre de manière indépendante avec le démarrage du composant migré sur le dispositif cible par l'entité de supervision sur le dispositif cible.

12. Système de supervision selon l'une des revendications précédentes, caractérisé en ce que le composant sur le dispositif source était lié à un connecteur distribué entre le dispositif source et le dispositif cible, et en ce que l'entité de supervision sur le dispositif source est apte à rediriger ledit connecteur distribué en créant un connecteur interne sur le dispositif cible.

13. Système de supervision selon l'une des revendications 1 à 11, caractérisé en ce que le composant sur le dispositif source était lié à un connecteur interne lié au composant sur le dispositif source, et en ce que l'entité de supervision sur le dispositif source est apte à rediriger ledit connecteur interne en créant un connecteur distribué sur le dispositif cible et le dispositif source si le dispositif cible et le dispositif source ont des réseaux de communication compatibles, ou d'un connecteur relai entre le dispositif cible et le dispositif source, si le dispositif cible et le dispositif source n'ont pas de réseaux de communication compatibles.

14. Système de supervision selon l'une des revendications 1 à 11, caractérisé en ce que le composant sur le dispositif source était lié à un connecteur distribué ou un connecteur relai entre le dispositif source et un dispositif donné distinct du dispositif source et du dispositif cible, et en ce que l'entité de supervision sur le dispositif source est apte à rediriger ledit connecteur distribué en créant un connecteur distribué sur le dispositif cible et le dispositif donné si le dispositif cible et le dispositif donné ont des réseaux de communication compatibles, ou d'un connecteur relai entre le dispositif cible et le dispositif donné, si le dispositif cible et le dispositif source n'ont pas de réseaux de communication compatibles.

15. Système de supervision selon l'une des revendications 14 et 15, caractérisé en ce que la création d'un connecteur distribué ou relai comprend la synchronisation des parties du connecteur distribué ou relai par un échange de messages d'acquittement.

16. Système de supervision selon la revendication 15, caractérisé en ce que la synchronisation des parties du connecteur distribué ou relai comprend la création d'interfaces logicielles de communication entre les parties de connecteurs, lesdites interfaces logicielles étant utilisées pour le transfert de données sur ledit connecteur distribué ou relai.

17. Système de supervision selon l'une des revendications précédentes, caractérisé en ce que l'entité de supervision sur chaque dispositif est apte à contrôler chaque connecteur hébergé sur le dispositif en utilisant un conteneur (805) encapsulant le connecteur (303).

18. Système de supervision selon l'une des revendications précédentes, caractérisé en ce que l'entité de supervision du dispositif cible est apte à communiquer avec le conteneur du composant pour l'arrêter jusqu'à ce qu'un connecteur lui soit connecté.

19. Système de supervision (10) selon l'une des revendications précédentes, caractérisé en ce que les données échangées entre les composants sont encapsulées dans une classe, et en ce qu'une donnée reçue par un composant hébergé sur un dispositif est désencapsulée et traitée par l'entité de supervision si le dispositif dispose de la classe du composant.

20. Système de supervision selon l'une des revendications précédentes, caractérisé en ce que le message de demande de migration comprend des informations relatives à l'état des entrées et sorties du composant.

21. Procédé de supervision d'applications s'exécutant sur un ensemble de dispositifs électroniques (5) reliés entre eux par un ou plusieurs réseaux, chaque dispositif (5) comprenant une entité de supervision locale (6), les entités de supervision coopérant entre elles pour contrôler les applications s'exécutant sur les dispositifs électroniques (5), chaque application comprenant un ensemble de composants applicatifs, chaque composant applicatif (302) étant encapsulé dans un conteneur (305) par l'entité de supervision du dispositif hébergeant le composant, et les composants étant reliés entre eux par des connecteurs (303), caractérisé en ce que le procédé comprend, en réponse à la réception par l'entité de supervision d'un dispositif donné, dit dispositif source, d'une commande de migration d'un composant du dispositif source vers un dispositif cible :

- arrêter le composant, l'arrêt du composant interrompant l'arrivée de données dans les connecteurs d'entrée du composant,

-sérialiser et encapsuler les propriétés du composant dans un conteneur de l'entité de supervision,

-envoyer un message de demande migration à l'entité de supervision du dispositif cible, ledit message comprenant le composant sérialisé et encapsulé, et

- rediriger les connecteurs (303) du composant en fonction de l'état des connexions de chaque connecteur sur le dispositif source.

068712

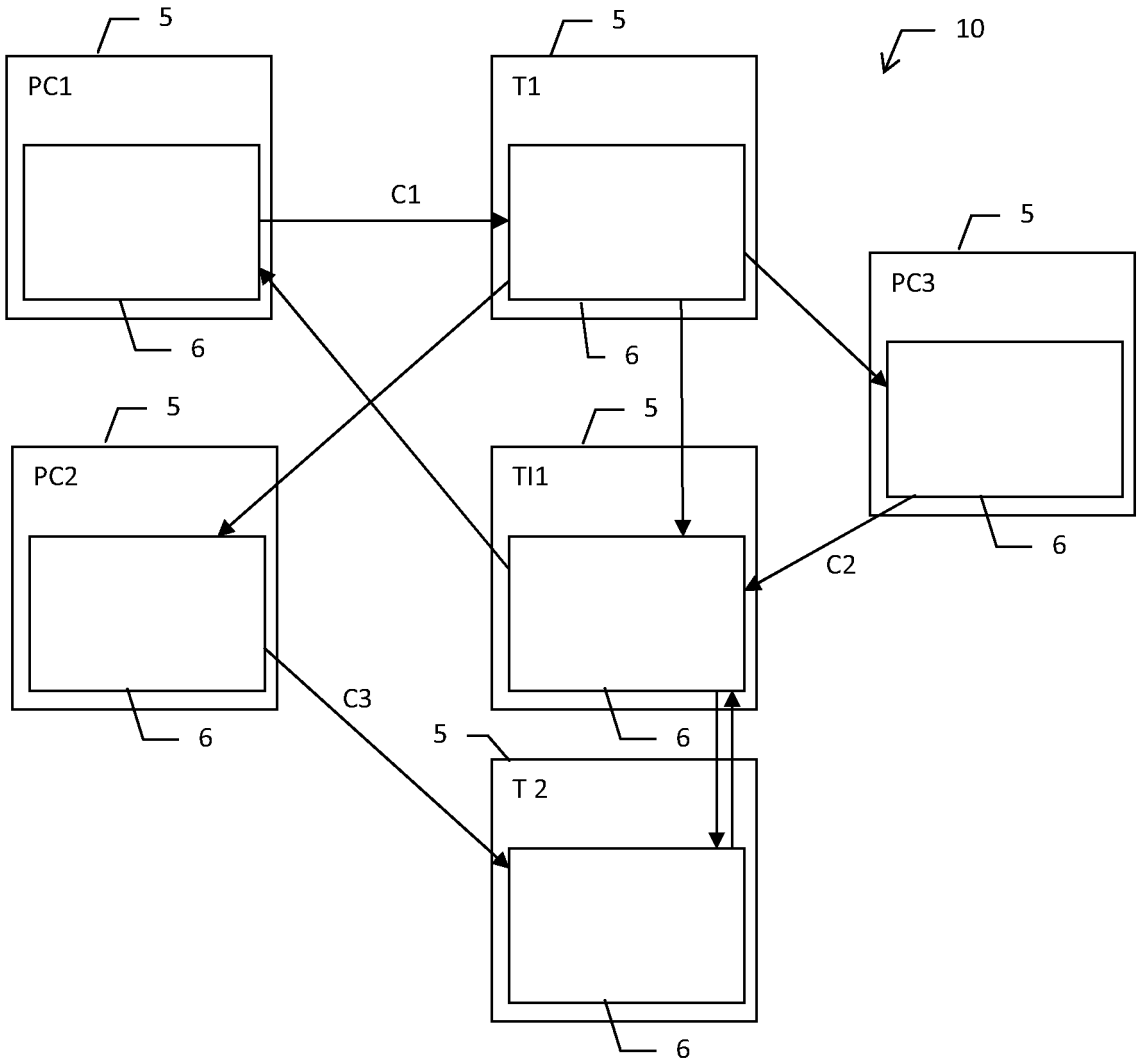


Figure 1

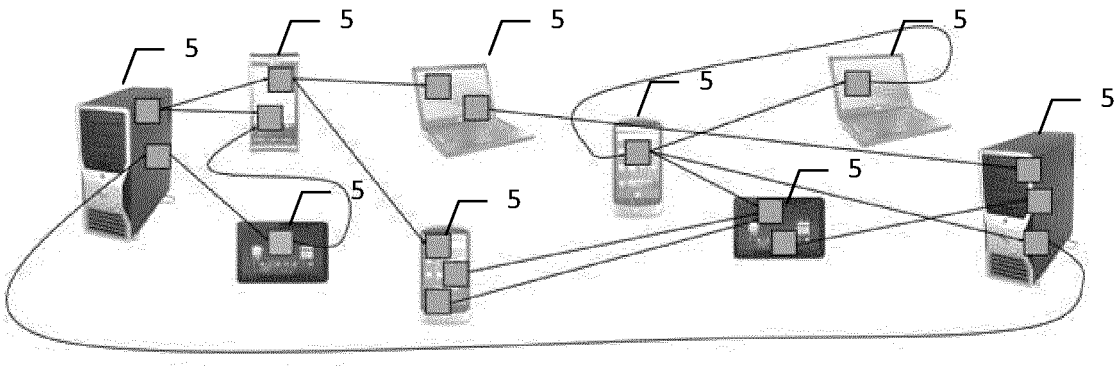


FIGURE 2

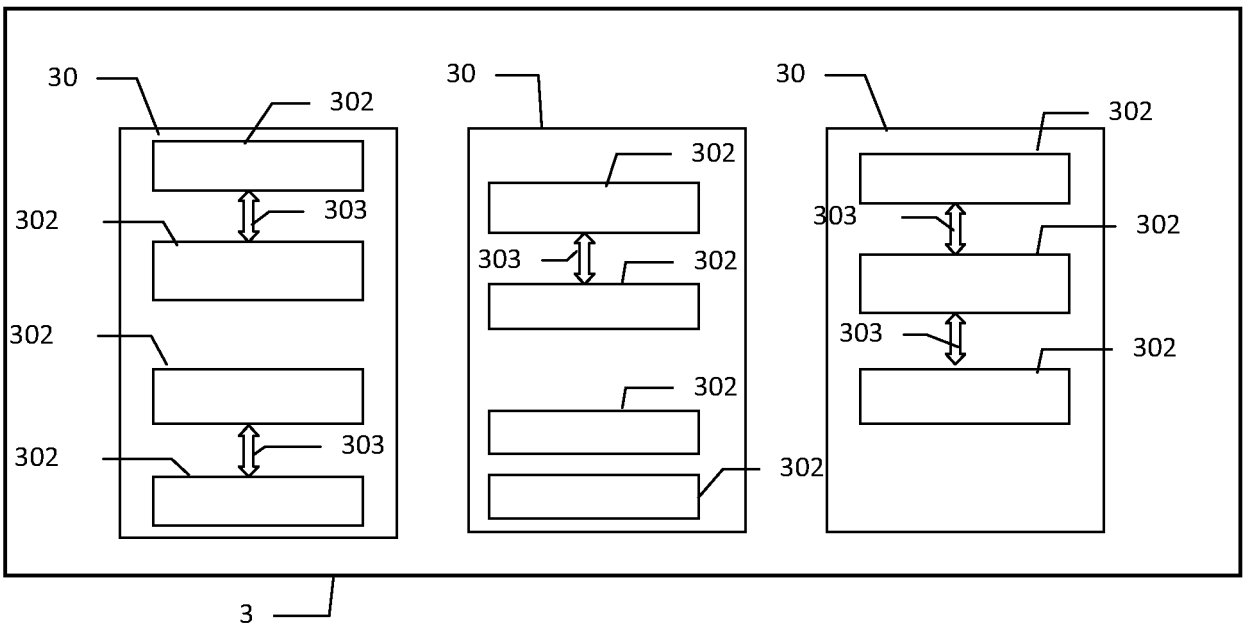


Figure 3

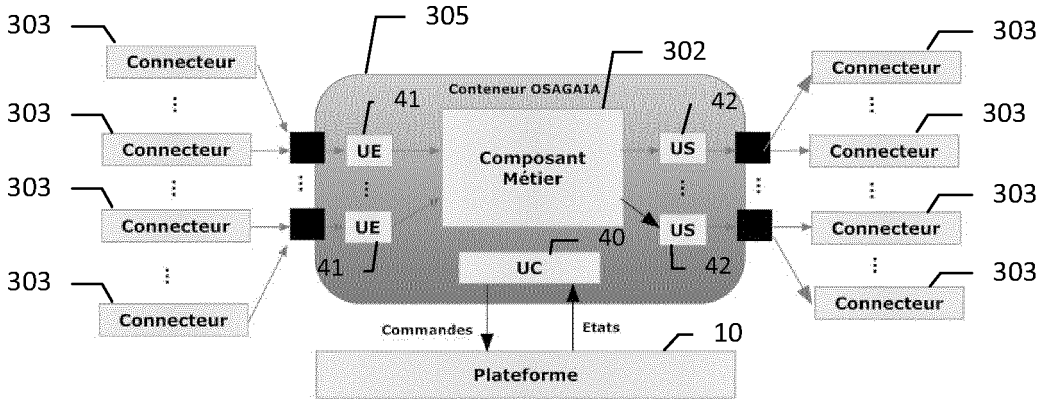


Figure 4

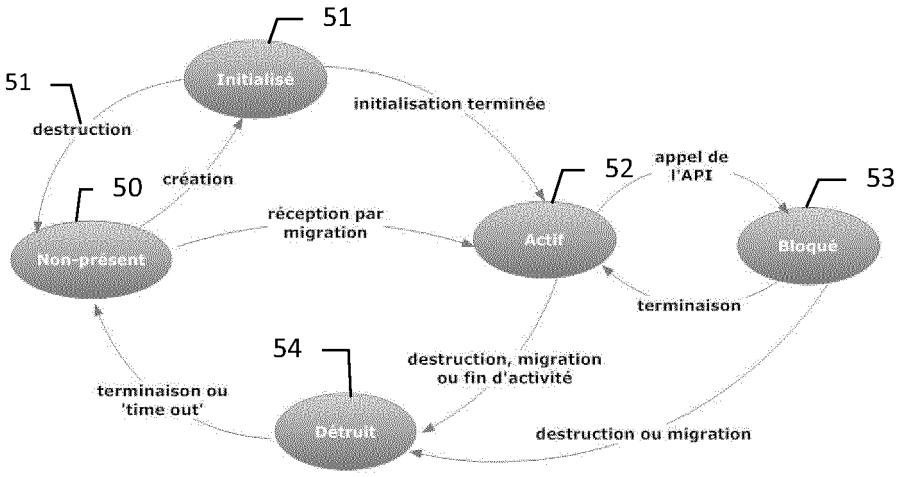


Figure 5

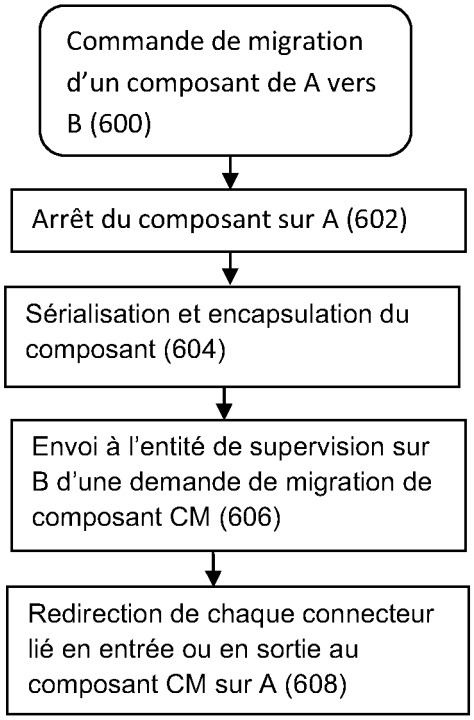

Commande de migration d'un composant de A vers B (600)
Arrêt du composant sur A (602)
Sérialisation et encapsulation du composant (604)
Envoi à l'entité de supervision sur B d'une demande de migration de composant CM (606)
Redirection de chaque connecteur lié en entrée ou en sortie au composant CM sur A (608)

FIGURE 6

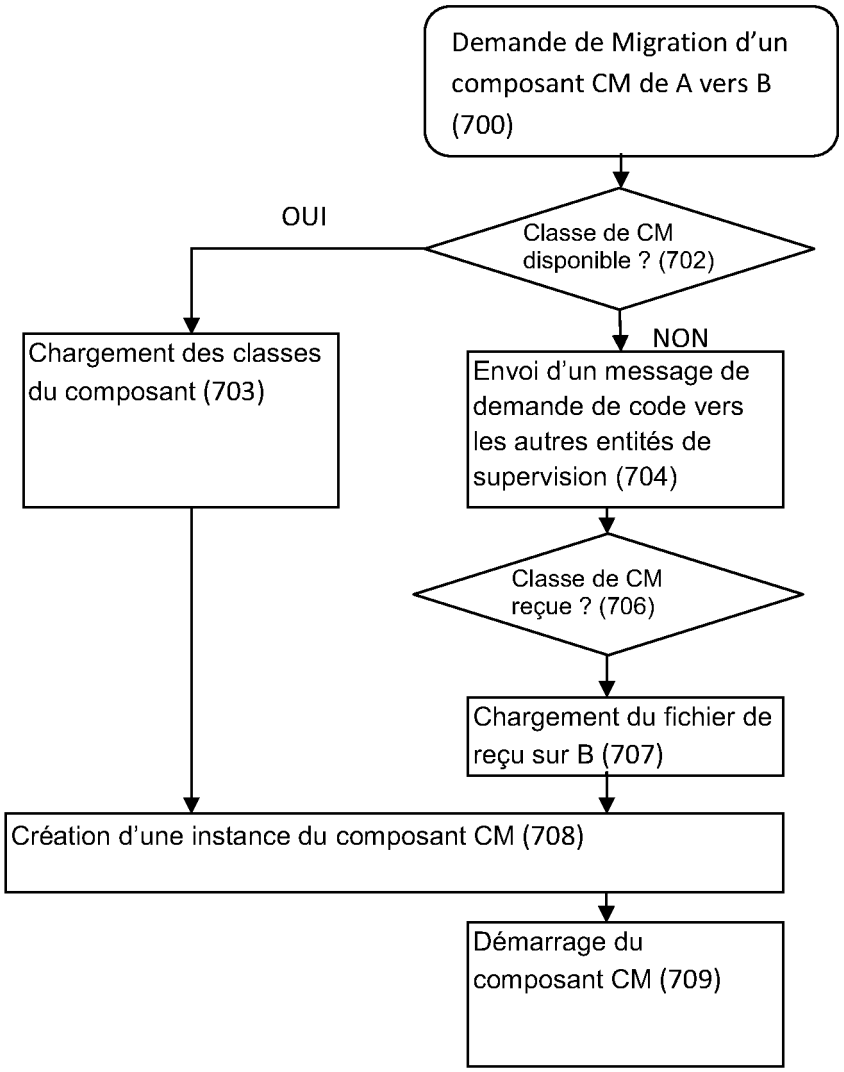

Demande de Migration d'un composant CM de A vers B (700)
Classe de CM disponible ? (702)
OUI
NON
Chargement des classes du composant (703)
Envoi d'un message de demande de code vers les autres entités de supervision (704)
Classe de CM reçue ? (706)
Chargement du fichier de reçu sur B (707)
Création d'une instance du composant CM (708)
Démarrage du composant CM (709)

FIGURE 7

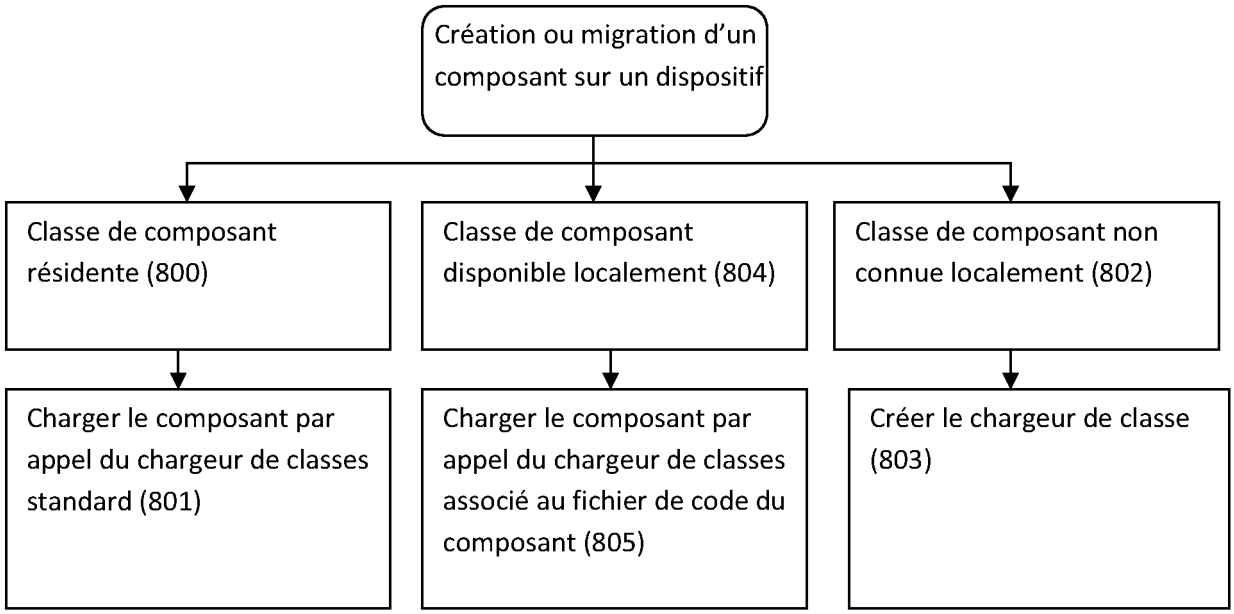


FIGURE 8

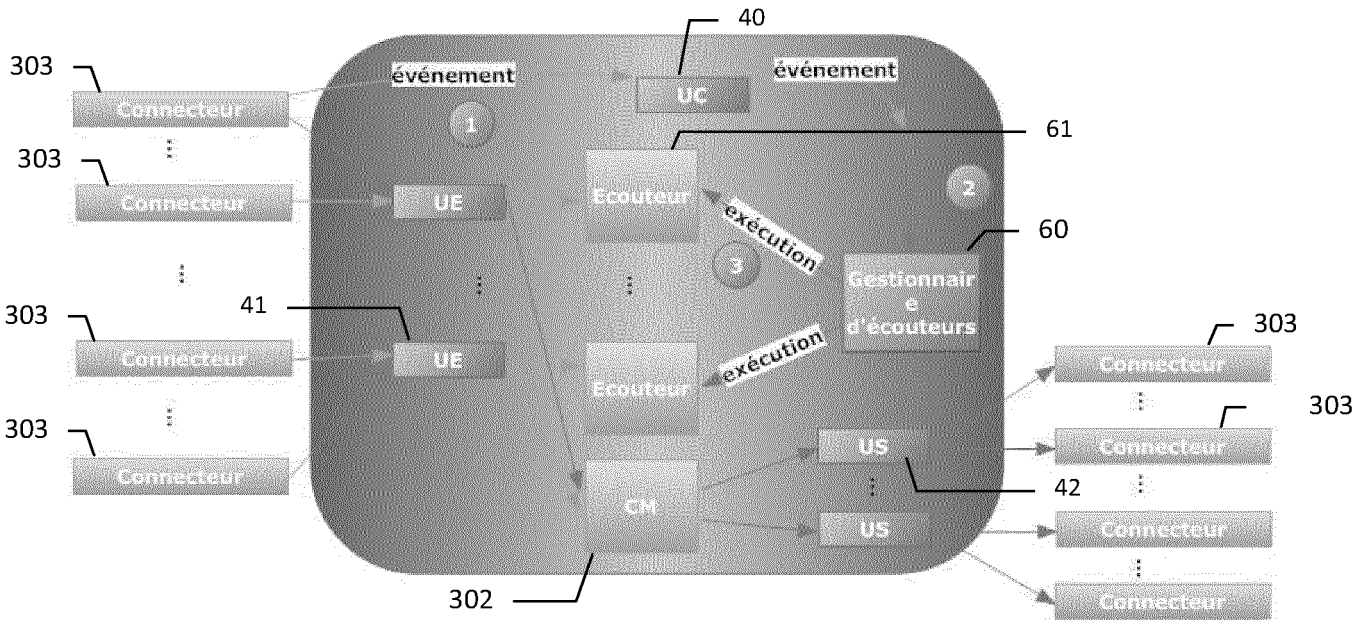


FIGURE 9

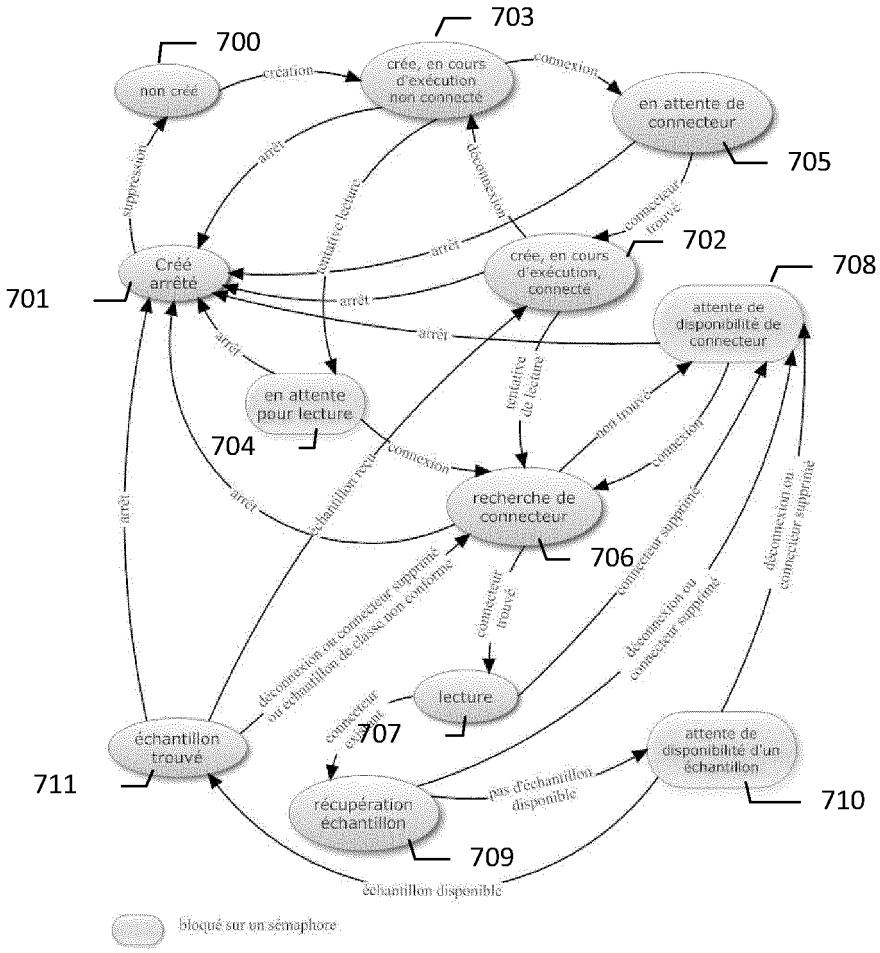


FIGURE 10

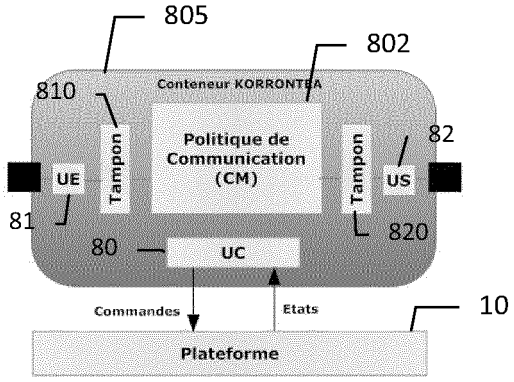


Figure 11

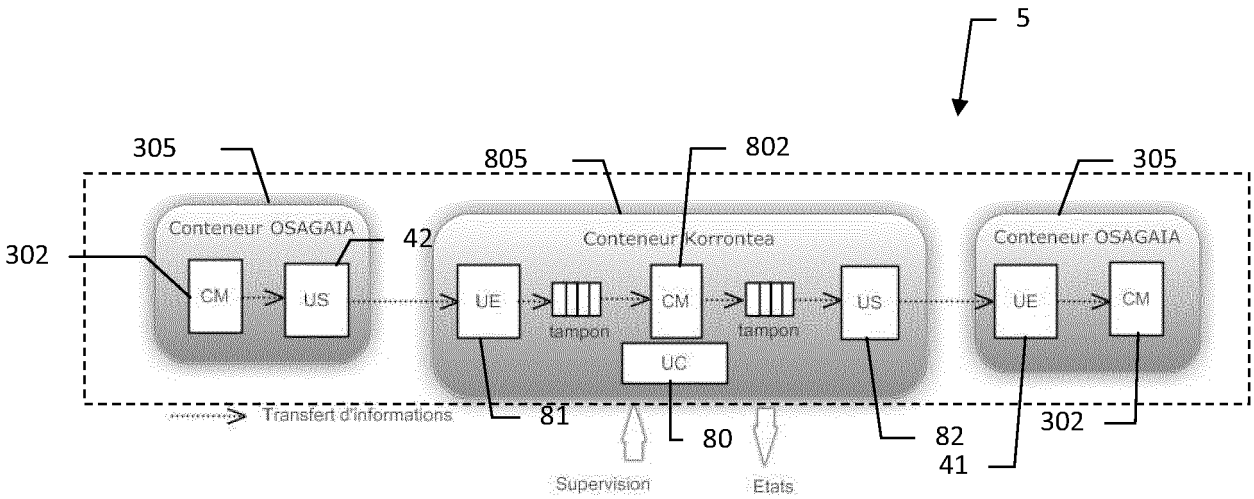

5
305
805
802
305
Conteneur OSAGAIA
Conteneur Korrontea
Conteneur OSAGAIA
302
42
CM
US
UE
tampon
CM
tampon
US
UE
CM
UC
Transfert d'informations
81
80
82
41
302
Supervision
Etats

Figure 12

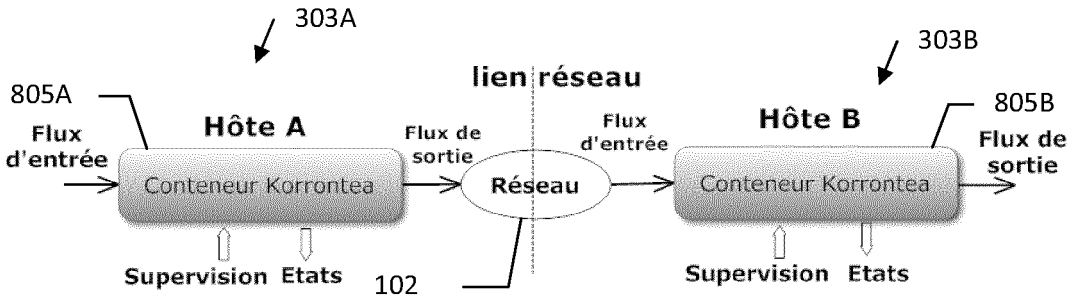


Figure 13

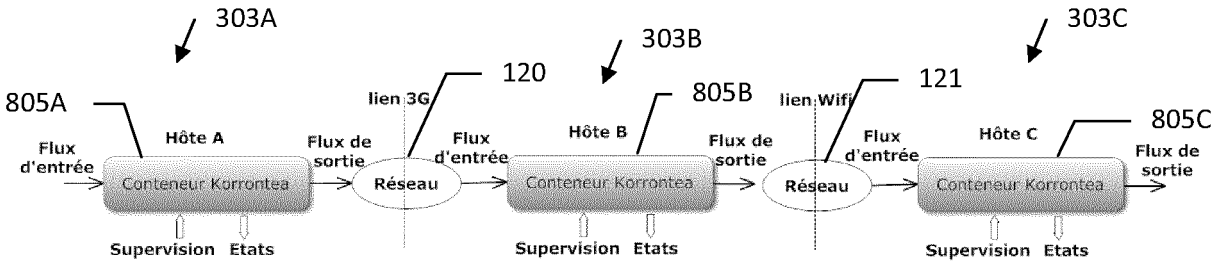


Figure 14

Sur A, le connecteur venait de ou allait vers	**Sur B, le connecteur vient de ou va vers**
A (connecteur interne)	A (connecteur distribué ou par relais)
B (connecteur distribué ou par relais)	B (connecteur interne)
C (connecteur distribué ou par relais)	C (connecteur distribué ou par relais)

Figure 15

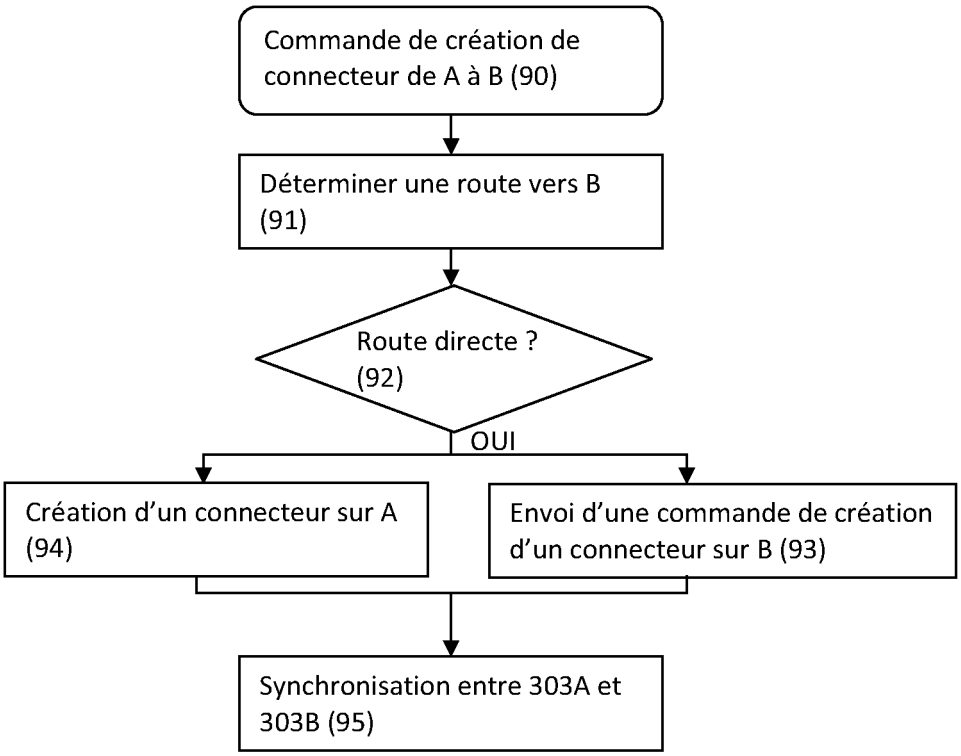

Commande de création de connecteur de A à B (90)
Déterminer une route vers B (91)
Route directe ? (92)
OUI
Création d'un connecteur sur A (94)
Envoi d'une commande de création d'un connecteur sur B (93)
Synchronisation entre 303A et 303B (95)

FIGURE 16

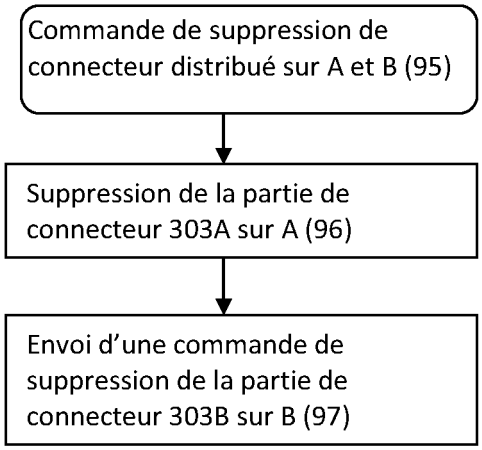

Commande de suppression de connecteur distribué sur A et B (95)
Suppression de la partie de connecteur 303A sur A (96)
Envoi d'une commande de suppression de la partie de connecteur 303B sur B (97)

FIGURE 17

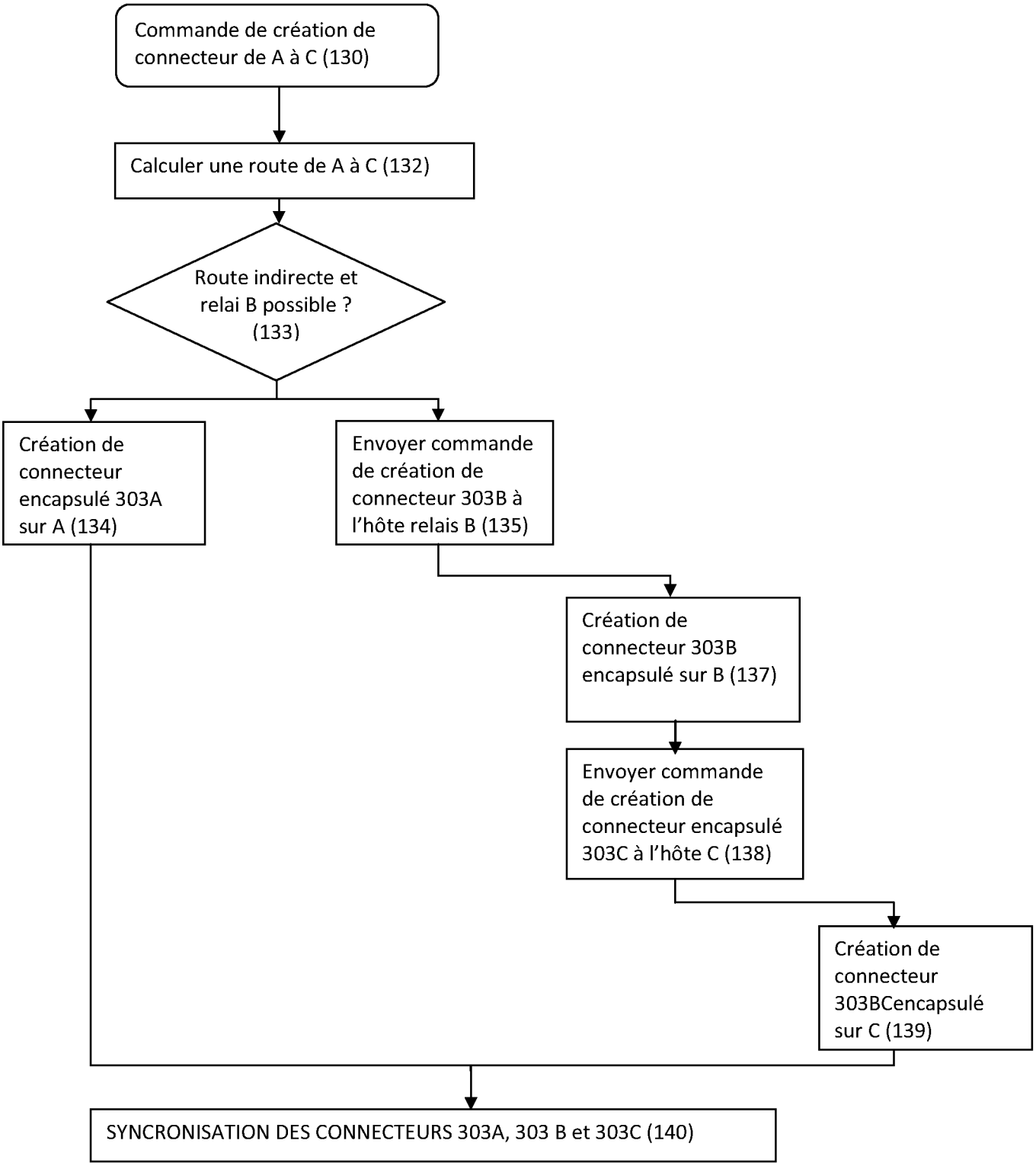

Commande de création de connecteur de A à C (130)
Calculer une route de A à C (132)
Route indirecte et relai B possible ? (133)
Création de connecteur encapsulé 303A sur A (134)
Envoyer commande de création de connecteur 303B à l'hôte relais B (135)
Création de connecteur 303B encapsulé sur B (137)
Envoyer commande de création de connecteur encapsulé 303C à l'hôte C (138)
Création de connecteur 303BCencapsulé sur C (139)
SYNCRONISATION DES CONNECTEURS 303A, 303 B et 303C (140)

Figure 18

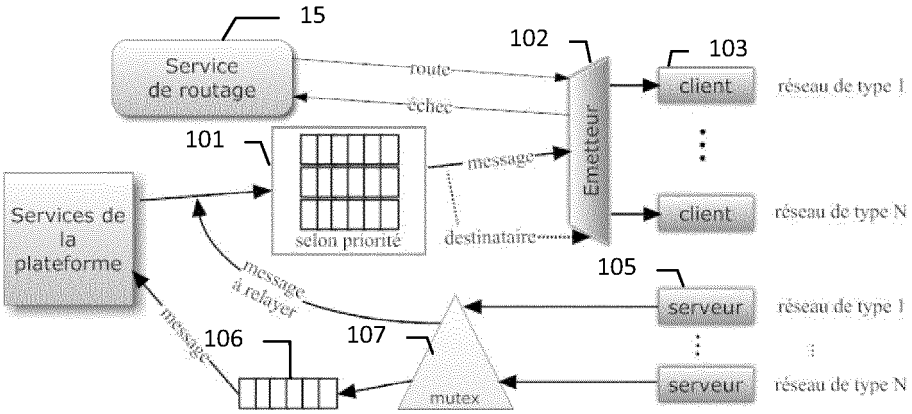


FIGURE 19

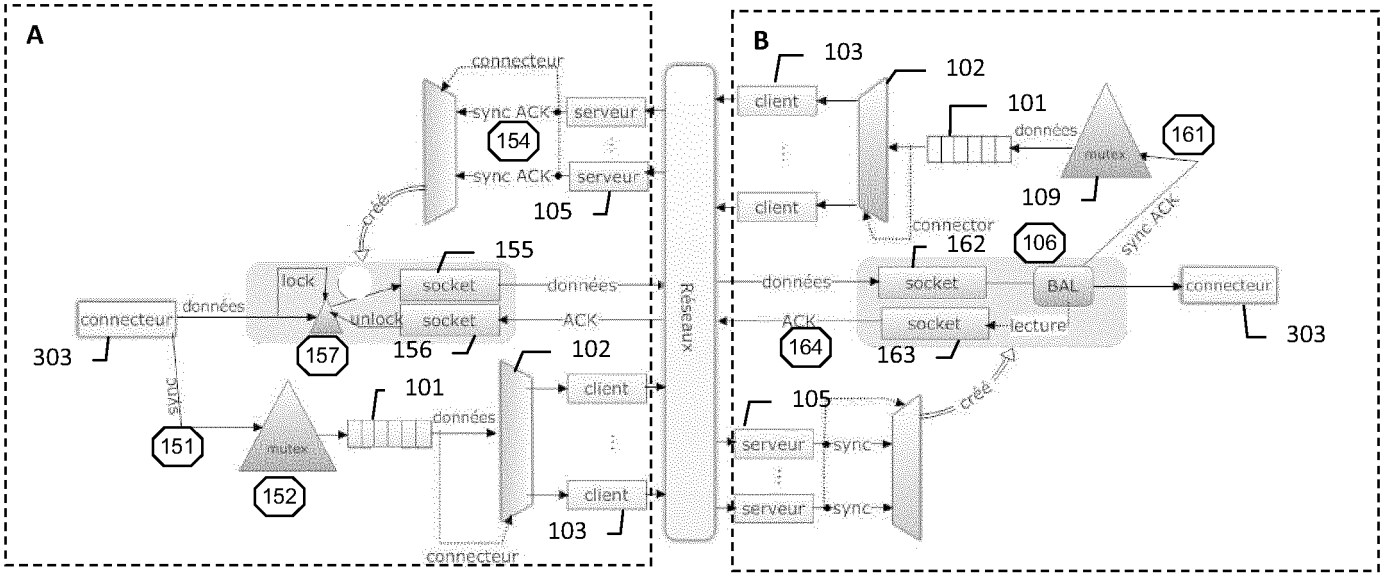


Figure 20

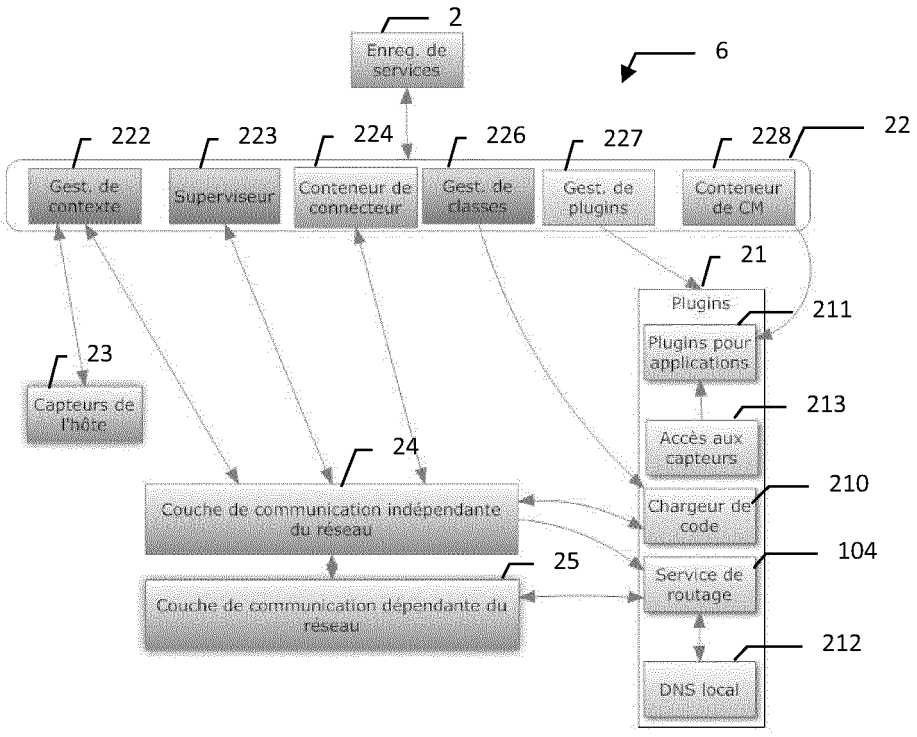

2
Enreg. de services
6
222
223
224
226
227
228
22
Gest. de contexte
Superviseur
Conteneur de connecteur
Gest. de classes
Gest. de plugins
Conteneur de CM
21
Plugins
211
Plugins pour applications
213
Accès aux capteurs
210
Chargeur de code
104
Service de routage
212
DNS local
23
Capteurs de l'hôte
24
Couche de communication indépendante du réseau
25
Couche de communication dépendante du réseau

Figure 21

RÉPUBLIQUE FRANÇAISE

INSTITUT NATIONAL DE LA PROPRIÉTÉ INDUSTRIELLE

RAPPORT DE RECHERCHE PRÉLIMINAIRE

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

N° d'enregistrement national

FA 782657
FR 1354889

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	Marc Dalmau ET AL: "Kalimucho - OW2 annual conference", OW2 annual conference 2011, 6 janvier 2012 (2012-01-06), pages 1-57, XP055098717, Extrait de l'Internet: URL:http://www.ow2.org/xwiki/bin/download/OW2Con-2011/Program/Kalimucho-OW2con11-M-Dalmau.pdf [extrait le 2014-01-27] * le document en entier * -----	1-21	H04L12/24 H04L12/16
X	Marc Dalmau ET AL: "Kalimucho, Part I", , 6 janvier 2012 (2012-01-06), page 1, XP054975303, Extrait de l'Internet: URL:http://www.youtube.com/watch?v=n5p7CV2lEVI [extrait le 2014-01-30] * le document en entier * -----	1-21	
A	Mohammed El Amine Matougui ET AL: "A Middleware Architecture for Autonomic Software Deployment", The Seventh International Conference on Systems and Networks Communications, ICSNC 2012, 18 novembre 2012 (2012-11-18), pages 13-20, XP055099321, ISBN: 978-1-61-208231-8 Extrait de l'Internet: URL:http://www.thinkmind.org/download.php?articleid=icsnc_2012_1_30_20169 [extrait le 2014-01-30] * alinéa [III.E] * * alinéa [III.F] * * alinéas [00IV] - [00VI] * -----	1-21	

DOMAINES TECHNIQUES RECHERCHÉS (IPC)

G06F
H04L

Date d'achèvement de la recherche	Examinateur
31 janvier 2014	Peeters, Dirk

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

EPO FORM 1503 12.99 (P04C14)