**Abstract** :

L’internet des objets (IoT) a largement émergé ces dernières années grâce à la prolifération des réseaux de communications et des technologies d’informations. Il repose sur l'interconnexion d'une multitude d'objets connectés hétérogènes, dotés de ressources limitées et déployés dans des environnements distribués à large échelle. Le Web des objets (WoT) est un nouveau paradigme qui est apparu pour assurer l’interopérabilité en terme de représentation de données et de protocoles d’interaction entre les différents objets IoT en s’appuyant sur la notion d’avatar [c43].

Pour surmonter les défis liés au déploiement à large échelle de l'IoT et la découverte efficace des ressources entre avatars, l’intégration des concepts des réseaux sociaux au WoT donnant naissance ainsi au Social Web of Things (SWoT) se montre comme une solution très prometteuse permettant aux différents appareils d'interagir, de faciliter le partage d'informations et par conséquent permettre l’émergence d’une variété d'applications attrayantes. Les avatars appartenant au même réseau social doivent être capables de raisonner conjointement sur leurs services respectifs et les composer avec ceux des autres avatars pour atteindre un but commun d’une manière autonome, distribuée et interactive. Pour ce faire, ils utilisent un protocole d’interaction en échangeant plusieurs types de messages (REQUEST, PROPOSE, CONFIRM, etc) pour négocier la réalisation d’une tâche donnée.

Dans cet article, nous proposons une nouvelle manière dynamique et distribuée basée sur les avatars introspectifs dotés de capacités de raisonnement sémantiques qui leurs permettent de prendre des décisions intelligentes, de se coordonner entre eux pour accomplir des tâches complexes en utilisant un langage de communication expressif et les relations sociales qui rend la découverte et la sélection des services plus efficace.

**Introduction**

L'Internet des objets (IoT) est une tendance très en vogue actuellement. Il permet d’interconnecter une multitude d’objets hétérogènes, gérer leurs interactions intenses, générer d’énorme flux de données et de fournir de millions de services. Au fil du temps, l’IoT a souffert d’une fragmentation verticale des approches adoptées pour couvrir les besoins des différents domaines d’application. En effet, diverses solutions avec des APIs distinctes ont été conçues séparément pour les différentes catégories d’applications. Cela a engendré des solutions «silos» uniques qui fonctionnent souvent dans le matériel spécifique du fournisseur ce qui limite considérablement le développement des applications IoT en les rendant très dépendantes du matériel et par conséquent très difficiles à déployer et à maintenir. Pour pallier ce problème, une nouvelle couche abstraire est conçue au-dessus de ce paradigme qui cache tous les problèmes d’hétérogénéité IoT soit au niveau de données ou au niveau des communications appelée Web of Things (WoT). Le WoT vise essentiellement à réduire le processus de développement des applications IoT en représentant chaque objets IoT (appareil physique ou entité logicielle) par un artefact sur le Web appelé Avatar. L’avatar permet d’exposer les services offerts par les divers dispositifs IoT sous forme de ressources accessibles via des interfaces RESTful standards. Il est également capable de raisonner sur ces propres données contextuelles et prendre des décisions intelligentes de comportement à adopter comme il peut collaborer avec d’autres avatars pour résoudre des problèmes complexes. Un avatar peut être déployé soit sur le device qu’il représente, soit sur des nœuds Fog ou dans le Cloud selon le contexte. Les infrastructures basées sur la notion d’avatars constituent l'une des solutions les plus prometteuses pour le WoT car elles permettent d’assurer l’interopérabilité d’un côté et l’autonomie des décisions d’un autre côté.

Ces dernières années, beaucoup d'attention est donnée au sujet de l’intégration des concepts de sociabilités dans le domaine de Web des Objets donnant ainsi le paradigme Social Web of Things (SWoT) permettant ainsi de transformer les avatars intelligents en avatars sociaux. Ces derniers auront l’habilité de créer des liens sociaux d’une manière autonome pour faire preuve d'un comportement collectif dans la résolution des problème complexes.

Cette nouvelle vision d’avatars sociaux présente plusieurs avantages et promet d’important progrès notamment en terme découverte opportuniste des ressources et services, en terme d’évolutivité des systèmes IoT via l’intégration de plus en plus d’objets connectés et de sécurité en intégrant la métrique de confiance dans l’établissement des relations sociales.

L’approche qu’on propose décrit une nouvelle vision dans laquelle les avatars appartenant au même réseau social doivent être capables de communiquer et de coopérer efficacement pour mener des problèmes complexes communs. L'une des approches les plus utilisées s’agit des protocoles d’interaction où les avatars interagissent entre eux pour découvrir et composer leurs services par envoi de messages.

Dans ce contexte, nous visons à travers ce travail à : a) énumérer les principales activités de recherche réalisées dans le cadre des SWoT ; b) décrire l’architecture SWoT basée sur les avatars et les fonctionnalités requises à intégrer dedans pour permettre la mise en place d’un réseau social entre avatars; c) identifier les politiques les plus appropriées pour l’établissement des relations de voisinage sociales entre les avatars et leur gestion. d) proposer un protocole d’interaction entre avatars pour la composition de leurs services en tenant en compte de leur aspect non fonctionnel (QoS). e) Et enfin, valider la solution proposée à travers des expérimentations.

1. **Related works**

Les paradigmes IoT [c1] représentent des systèmes dynamiques à large échelle qui envisagent la présence d’une multitude d’objets hétérogènes capables d’interagir et de collaborer entre eux pour fournir des services à valeur ajoutée et créer des applications IoT attrayantes[c2].

Ce domaine relève plusieurs défis notamment en terme de scalabilité et d’hétérogénéité [c3]. Pour faire face à ces problèmes, de plusieurs types d’environnements intelligents [c3], de nombreux standards IoT comme smartM2M et oneM2M [c4] ainsi que de plateformes de services [c5] comme OM2M ont été proposés [c6]. La scalabilité de ce genre de systèmes requiert l’intégration des technologies de Cloud et de Fog Computing [c7] ainsi que les capacités d’interactions sociales [c8].

La mise en place des applications et les services IoT à valeur ajoutée repose sur la découverte et la composition des différents objets connectés dans des environnements hautement dynamique et imprévisible. Grâce à l’utilisation des plateformes de services IoT qui assurent l’interopérabilité de communication avec les différents types d’objets hétérogènes et exposent leurs fonctionnalités en tant que services RESTful, ce problème se transforme en un problème de composition de services.

Le problème de composition de services est largement étudié par plusieurs communautés scientifiques [c9, c11, c12, c13, c14, c15]. Les travaux existants peuvent être classés en deux catégories: ceux qui utilisent des techniques de composition statiques définies à l’aide d’un Business Process et qui sont à leur tour divisés en deux groupes : orchestration et chorégraphie. L’autre catégorie concerne les techniques de composition dites dynamiques capables de découvrir les services disponibles à un instant donné ainsi que leurs fonctionnalités pour satisfaire un besoin donné et adapter cette composition en fonction de contexte.

Ces dernières années, l’automatisation et la dynamicité du problème ont portés beaucoup d’attention [c16, c17, c18, c19, c20] notamment à cause de l’émergence de nombreux types de services qui offrent les mêmes fonctionnalités mais avec des propriétés non-fonctionnelles différentes. L’aspect ouvert et distribué de ce problème a conduit les chercheurs à s’orienter vers les systèmes multi-agents. Les travaux réalisés dans ce contexte reposent essentiellement sur la planification [c21, c22, c23, c24]. On trouve des travaux qui s’appuient sur la planification non-linéaire classique qui supposent que les changements qui subissent dans le système n'étant dûs qu'aux actions produites par les agents [c25, c26, c27, c28, c29], la planification hiérarchique dans laquelle d’accent est mis sur la décomposition des objectifs à atteindre à l'aide d'un réseau hiérarchique de tâches HTN (Hierarchical Task Network) [c30,c31] ainsi que ceux qui reposent sur la planification linéaire qui se basent sur des règles [c32, c33].

L’application des méthodes de planification dans la composition automatique et dynamiques des services Web a montré leur efficacité et leur puissance [c34, c35, c36, c37, c38].  Dans [c39] exemple, chaque service est représenté par un agent dont le rôle est décrit par les paramètres fonctionnels de ce service. Ils considèrent toutes les structures de composition contrairement à plusieurs autres travaux qui ne considèrent que des séquences de services [c40, c41]. La plupart des travaux proposés utilisent des descriptions syntaxiques non suffisantes pour la composition automatique de services, pour cela les descriptions sémantiques sont apparues comme une nouvelle alternative [c42]. La convergence des services Web vers le Web sémantique permet essentiellement rendre la découverte, la sélection, la composition ainsi que l’invocation des services plus explicite en augmentant leur expressivité avec des notations sémantiques.

Cependant, la majorité des travaux proposés reposent sur la centralisation de la composition en utilisant un agent contrôleur (Manager) qui se charge de découvrir, de sélectionner, de composer les Web services et de les orchestrer pendant l’exécution. Ceci n’est plus adapté aux écosystèmes IoT constitués de milliards voire billions d’objets offrant chacun une multitude de service. En plus, de très peu de travaux intègrent les technologies de Fog Computing et des interactions sociales dans leur manière de procéder.

La particularité de notre contribution réside dans le fait d’intégrer ces mécanismes pour la création des applications IoT attrayantes en utilisant une architecture basée sur les avatars. Dans ce travail, on propose un nouveau protocole d’interaction entre avatars autonomes pour la découverte, le filtrage et la sélection de services fournis par les différents objets connectés d’une manière distribuée et décentralisée. Notre protocole d’interaction tire d’avantage des relations sociales qu’un avatar peut avoir les autres avatars de son environnement. De plus, tous les services sont décrits via une description sémantique ce qui qui permet aux avatars de raisonner sur leurs capacités.

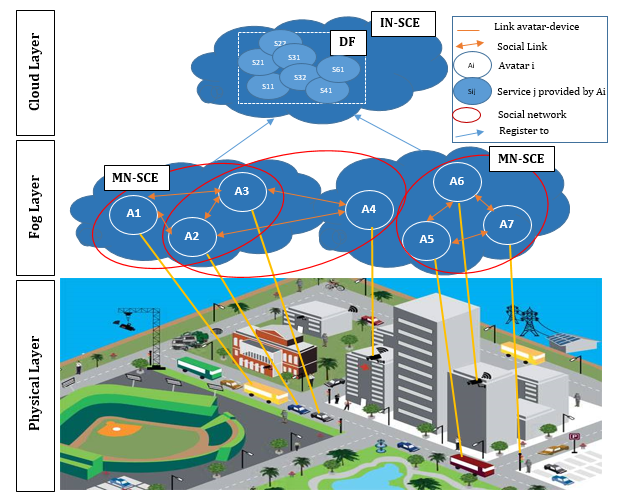
1. **Social Web of Things**

Le SWoT fait référence à un réseau d’objets intelligent capables de s’organiser entre eux pour former des communautés en créant des relations sociales comme le font les humains. Ce paradigme gagne d’avantage du terrain grâce aux avantages tirés de l’intégration de notion de sociabilité aux avatars sur la WoT. Il permet à un avatar donné de découvrir d’une manière autonome les avatars permettant de fournir les services requis et disposant des caractéristiques souhaitées en utilisant ses relations avec ses voisins et les voisins de ses voisins d’une manière distribuée afin de garantir une découverte efficace et évolutive. Il facilite également la navigabilité dans un réseau dynamique constitué d’une très grand nombre d’objets voir des milliards d'objets. Comme il permet de simplifier la gestion de la fiabilité entre les objets appartenant à la même communauté.

Le regroupement social des avatars en communautés peut être guidé par plusieurs critères tels que leurs intérêts en commun, des avantages mutuels qu’ils peuvent échangés ainsi que leur localisation géographique. Atzori et al. [c44] ont défini cinq type de relations social:

1. Parental Object Relationship (POR): ce type de relation est établi entre des objets homogènes issus de même fabricant et appartenant au même lot de production;
2. Co-Location Objects Relationship (C-LOR) : établie entre des objets qu’ils soient homogènes ou hétérogènes situés au même endroit à un diamètre donné ;
3. Co-work Object Relationship (C-WOR): établie entre deux ou plusieurs objets coopérant pour réaliser un objectif en commun ;
4. Ownership Object Relationship (OOR): relation englobant les objets appartenant au même propriétaire;
5. Social Object Relationship (SOR): relation qui se produit lorsque les objets entrent en contact de façon sporadique ou continue lorsque leurs propriétaires se croisent tels que les camarades de la même classe et les membres de la même famille.
6. **GENERIC ARCHITECTURE**

Dans un premier temps, il convient d’élaborer la vue architecturale sur laquelle s’appuie le paradigme de l’IoT pour mieux comprendre son fonctionnement. Nous remarquons dans la Figure 1 que la structure de l’IoT est divisée en trois couches décrites comme suivant:



*Figure 1 : Generic SWoT Architecture*

**PS :** C’est juste pour des raisons de visibilité qu’on a présenté les avatars uniquement au niveau de la couche Fog.

**Physical Layer :**  elle regroupe les divers dispositifs IoT physiques connectés à Internet (les lecteurs, les différents types de capteurs, etc.) qui ont pour fonction de collecter des grandeurs physiques analogiques produites par l’environnement telles que l’humidité et le mouvement par exemple et d’échanger des données entre eux. Si ces dispositifs possèdent les ressources nécessaires pour déployer une passerelle OM2M, les avatars qui les représentent sont déployés directement dedans.

**Fog Layer :** cette couche est physiquement située entre la couche physique et le Cloud. Elle est composée des périphériques réseau proches des devices des utilisateurs ce qui permet de garantir l’intégrité et la sécurité des données et libérer de la bande passante. Ces périphériques jouent le rôle de passerelles MN-CSE selon l’architecture OM2M. Chaque MN-CSE peut héberger un ou plusieurs avatars dans des containers JADE. Ces avatars peuvent constituer des communautés grâce aux relations sociales qui les relient.

**Cloud Layer**: cette couche est constituée de plusieurs serveurs Cloud. Cela permet d’exploiter d’avantage des ressources de calcul et de stockage illimitées de ces serveurs. Elle héberge le serveur IN-CSE de OM2M qui héberge à son tour la main container de JADE où les avatars peuvent publier leurs capabilités et chercher les avatars qui peuvent réaliser une fonctionnalité donnée. Des avatars peuvent être également déployés dans cette couche.

1. **MODÈLE DE SYSTÈME ET HYPOTHÈSES**

Les avatars doivent être capables de s’organiser pour former des communautés sociales d’une manière autonome afin de réaliser des applications IoT qui requirent la collaboration de plusieurs avatars.

Nous supposons que le réseau social des avatars est conçu en superposition de leur réseau de communication, et que tous les avatars sont connectés via les différentes technologies à plusieurs passerelles et qui sont reliées à leur tour à un serveur central. Par conséquent, chaque paire d’avatars possèdent un lien de communication logique entre eux et donc forment un réseau maillé complet. Nous supposons également que chaque avatar du système a au moins un intérêt (confort, météo, etc).

Il convient de noter que les liens sociaux entre avatars ne sont plus statiques, mais dynamiques et changent d’une manière imprévisible à cause de l’apparition et la disparition des périphériques mobiles du réseau. Par conséquent, les avatars doivent mettre à jour périodiquement leurs relations sociales. Chaque avatar doit disposer également d’une mémoire tampon (cache) pour stocker leurs informations sociales ainsi que les messages échangés avec ses homologues.

Le réseau social des avatars peur être représenté par un graphe connexe non orienté pondéré G=<A, E, SD> où A est l’ensemble des avatars, E est l’ensemble *d'arêtes* modélisant des liens sociaux entre eux et SD représente la mesure sociale associée à une arête donnée. Les arêtes sont ajoutées entre avatars en appliquant des algorithmes et des calculs de distance de similarité sociale essentiellement en fonction de leurs intérêts, de leur distance géographique ainsi que leur copropriété.

La distance sociale entre les avatars mobiles Ai et Aj peut-être modélisée par une fonction SD sur l’intervalle [-1,1] tel que SD : A×A→[-1 ,1]. Elle est calculée en se basant sur trois type de relations C-WOR (leurs intérêts), C-LOC (la distance qui les sépare géographiquement les devices) et OOR (on tient en compte l’appartenance au même propriétaire pour augmenter l’aspect sécuritaire). Des coefficients de pondération α, β et δ sont introduits pour équilibrer le poids entre les distances C-WOR, C-LOC et OOR respectivement.

La relation POR n’est pas considérée dans notre travail car le problème d’hétérogénéité n’est plus posé grâce à l’utilisation de la plateforme de services OM2M qui assure l’interopérabilité des différents technologies IoT (CAN, MQTT, Bluetooth, etc). En ce qui concerne la relation SOR, on compte l’utiliser dans la deuxième phase de notre travail qui consiste à introduire les mécanismes d’apprentissage pour l’utilisation des liens sociaux entres les avatars dans la sélection et de la composition de services.

**4.1. Co-work Object Relationship (C-WOR)**

Les avatars qui peuvent coopérer entre eux pour réaliser les différentes tâches composant une application IoT donnée sont classés dans la classe C-WOR. Ces avatars possèdent au moins un intérêt en commun. Afin de mesurer quantitativement la similarité en terme de C-WOR entre avatars, nous représentons les intérêts de chacun d’entre eux avec un vecteur de m dimensions où m représente le nombre de sujets d'intérêts. Chaque élément de ce vecteur représente le degré d'intérêt pour un sujet donné sous forme d’une valeur réelle inclue dans l’intervalle [0,1].

La similarité d’intérêts SI est calculée dans ce travail en utilisant la métrique de similarité du cosinus. Cette dernière est très utilisée pour mesurer la ressemblance en [fouille de textes](https://fr.wikipedia.org/wiki/Fouille_de_textes).

Soit deux vecteurs d’intérêts Ii et Ij des avatars Ai et Aj respectivement. La similarité cosinus [c47] entre ces deux avatars est obtenue par le produit scalaire et la norme des leurs vecteurs Ii et Ij comme suit :

(1)

Où ||I|| est la longueur du I et 0 ≤ SI(Ii, Ij) ≤ 1 tel que la valeur 0 désigne que Ii et Ij sont indépendants et 1 indique qu’ils sont similaires et homologues.

Le mécanisme de suscription/publication des intérêts de chaque avatar dans des pages jaunes au niveau de serveur semble être le plus approprié particulièrement dans les environnements mobiles où avatars peuvent apparaitre et disparaitre d’une manière dynamique. Ce mécanisme permet au avatars de découvrir facilement leurs homologues avec lesquels ils possèdent au moins un intérêt en commun.

**4.2 Co-Location Objects Relationship (C-LOR)**

Les avatars distribués dont les devices se trouvent dans la même zone géographique pendant un certain laps de temps ont tendance à collaborer ensemble dans le but de mener des objectifs en commun.

La similarité d’un point de vue géographique est calculée en utilisant les leurs coordonnées GPS selon l’équation suivante en utilisant la distance à vol d’oiseau D.

Tel que R=6371Km représente le rayon de la terre, LatAi et LatAj sont les latitudes décimales des avatars Ai et Aj respectivement, LongAi et LongAj représentent les longitudes des avatars Ai et Aj respectivement.

Pour le calcul de la similarité de cette distance, on utilise la ration qui prend une grande valeur à chaque fois que la distance est courte.

(2)

**4.3 Ownership Object Relationship (OOR)**

Certains utilisateurs préfèrent de n’utiliser que leurs propres dispositifs pour accomplir leur applications IoT d’une manière plus sécurisée sans se soucier de la dé-vulgarisation de leurs données personnelles. La relation OOR entre deux avatars Ai et Aj est représentée par la fonction suivante :

(3)

**Synthèse**

La distance sociale entre les deux avatars Ai et Aj est calculée comme suivant:

dij= α\*SI(Ai,Aj) + β\*SC(Ai,Aj) + δ\*P(Ai,Aj) (4)

Tel que α, β et δ sont les facteurs de pondération défini par l’utilisateur selon ses préférences.

**Le pseudo algorithme de construction de réseau social de l’avatar Ai**

*INPUT : Interest, α, β, δ, k*

*BEGIN*

*Discover the avatars list A of the system having at least an interest in common with Ai*

*for (Aj ε A) do*

*Calculate the similarity C-WOR between the interests of Ai and Aj by using (1)*

*Calculate the similarity of C-LOC between Ai and Aj avatars by using (2)*

*Calculate the similarity of OOR between Ai and Aj avatars by using (3)*

*Calculate the social distance dij between Ai and Aj using the coefficients α, β et δ and the equation (4)*

*Put the name of the avatar and its social distance in a table T*

*endFor*

*Sort the avatars according to their social distances*

*Add the first k avatars to the neighborhood of the avatar*

*END*

1. **Coordination d’avatars**

Afin que les avatars puissent accomplir leurs objectifs complexes d’une manière collective, ils utilisent un langage de communication expressif et une protocole d’interaction en considérant leurs liens sociaux d’une manière distribuée et décentralisée. Nous supposons dans notre approche que chaque avatar contient dans sa base de connaissances l’ensemble des actions qu’il peut effectuer décrites sémantiquement et qu’il est capable de savoir s’il peut contribuer à la réalisation d’un objectif donné en appliquant des règles SWRL sur sa base de connaissances.

Dans ce travail, nous proposons une nouvelle approche dynamique et distribuée pour la découverte et la composition des services fournis par les différents avatars pour mener un but complexe commun. Chaque avatar traite une partie du but, délègue ce qu’il ne peut pas traiter aux autres avatars voisins qu’il ne connaît pas à priori leurs capacités qui peuvent le propager ensuite à leurs voisins.

Notre approche utilise trois types de rôles:

***Initiateur:*** qui est en charge d’initier le processus de recherches après le déclenchement l’un événement donné et diffuser la requête correspondante au but à satisfaire aux avatars voisins, de recueillir les résultats et d’orchestrer l’exécution. N’importe quel avatar peut prendre le rôle d’initiateur mais souvent il est présenté par un avatar utilisateur (Applicatif).  Chaque avatar peut réaliser une ou plusieurs applications qui sont décrites d’une manière abstraite dans sa base de connaissances ;

***Participant***:  ou fournisseurs de services. Il s’agit d’un avatar impliqué dans la coordination et qui a pour objectif d’interagir avec d’autres avatars pour répondre aux besoins à satisfaire en fournissant un ou plusieurs services demandés ;

***Recommandeur :*** c’est un avatar appartenant au réseau social de l’initiateur mais qui ne possède pas de services utiles pour participer à la résolution de la requête initiée. Cet avatar propage la requête à ses voisins, recueille leurs réponses, filtrer les services proposés et sélectionne les meilleurs parmi eux pour les recommander à l’avatar initiateur.

Pour assurer de telles interactions entre avatars, un langage de communication doit être défini. Les messages échangés sont conformes au standard ACL-FIPA [c46] ont la structure suivante :

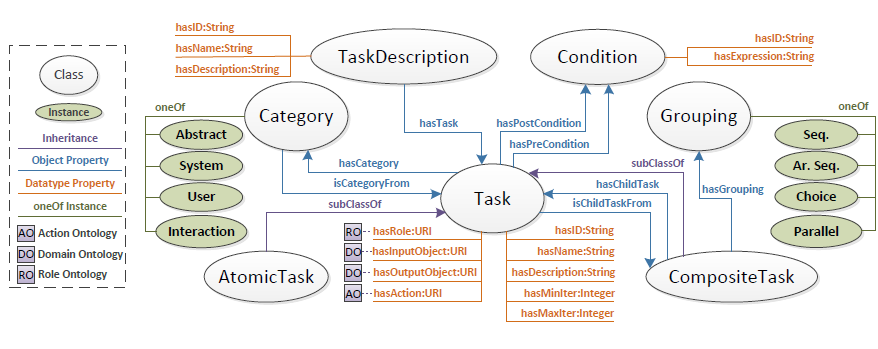
msg = [id, Content, performatif, sender, receivers, in-reply-to]

* id : l’identificateur de la conversation lancée pour résoudre le problème à traiter.
* Content : représente le contenu du message sous forme d’une ou du plusieurs requêtes atomiques C={r1, r2, …,rn} composant l’application. Chacune des requêtes atomiques exprime une tâche abstraite qui doit être réalisée par un service concret.
* Performatif : type d'acte illocutoire de la conversation lancée (INFORM, REQUEST, CONFIRM, etc).
* Sender : représente l’identifiant AID de l’avatar émetteur du message;
* Receivers : représente la liste des identifiants AIDs des avatars destinataires.
* in-reply-to : représente l’AID de l’avatar auquel la réponse doit être retournée.
* TimeOut : représente le délai de validité du message.

Notre approche comporte essentiellement cinq étapes [c48]: la définition abstraite du problème à traiter, la découverte, le filtrage et la sélection des services candidats, la composition des services choisis et enfin l’exécution de plan de composition par les avatars. Ces étapes sont détaillées dans ce qui suit :

* 1. **La définition abstraite**

Le problème global complexe à résoudre peut-être décomposer en plusieurs sous tâches qui peuvent être complexe ou atomiques. Plusieurs approches ont été proposées à cet effet. La description abstraite du flux de travail par le concepteur est l’une des approches les plus courantes vu son coût réduit. Pour ce faire, nous utilisons une ontologie qui permet de décrire les différents tâches composites et atomiques constituant l’application à réaliser et les dépendances logiques entre eux. La description sémantique des objectifs à réaliser fournit un cadre interopérable pour la compréhension des messages échangés entre eux lors de leur collaboration. Nous utilisons la même ontologie proposée par Tiezt et al [c45] avec l’ajout de quelques concepts concernant les structures reliant les tâches abstraites entre eux comme mentionné dans la Figure 2.



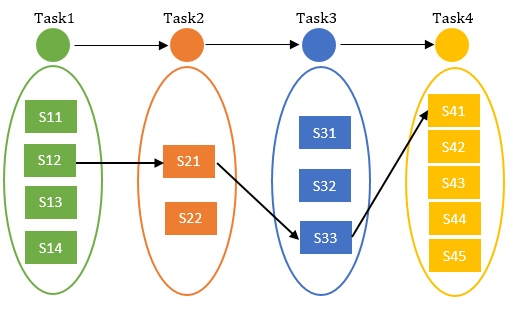
*Figure 2 : Task ontology (DEMISA) [Vincent Tietz et al; 2014]*

Les tâches atomiques composant une tâche composite peuvent être interconnectées via une de ces structures suivantes: séquentielle, parallèle (AND), boucle (Loop) et Conditionnelle (XOR).

L’ensemble des tâches abstraites sont modélisés et présentés par un ensemble Treq={T1, T2,. . . , Tn}.

Chaque tâche abstraite Ti possède un ensemble de services concrets Si={si1, si2,. . . , sim} qui peuvent la réaliser. Les services concrets Si ont les mêmes propriétés fonctionnelles mais se comportent différemment en terme de qualité de service dans la réalisation de même fonctionnalité. L’ensemble des paramètres QoS utilisés sont représentés par l’ensemble Q={q1, q2, .., qk} tel que qm(Sij) représente le m ème critère QoS de j ème service concret de la i ème tâche abstraite. Chaque tâche abstraite possède une contrainte temps réel locale que le service concret doit respecter.

La figure donne une vue globale de la modélisation des tâches abstraites et des services concrets.



Les services concrets constituent l’ensemble des services RESTful fournis par un ensemble d’avatars et qui représentent à leur tour les fonctionnalités des différents devices IoT du système considéré.

* 1. **Découverte des services**

Pour que les avatars puissent interagir et collaborer entre eux, ils doivent être en mesure de découvrir les services répondant aux fonctionnalités de chaque tâche abstraite de l’application. Rechercher un service revient à la recherche d’un ami avatar capable de le réaliser. Pour chaque service, certains types de relation sont privilégiés. Par exemple, si l’avatar a besoin des informations sur son environnement, le type de relation co-localisation est privilégié. Si aucun avatar ami capable de fournir le servir est trouvé, la requête de découverte est acheminée vers les amis des amis de l’avatar exploré. Si plusieurs avatars fournisseurs de service demandé sont trouvés, une sélection sur ces avatars doit être effectué en fonction de la QoS des services trouvés.

Une fois un évènement est déclenché suite à une demande d’utilisateur ou une action interne à l’avatar est produite (par exemple une observation a dépassé un seuil donné), la règle sémantique SWRL correspondante qui se trouve dans la base de connaissances de l’avatar en question est exécutée. La règle sémantique à son tour indique à l’avatar quelle application doit être exécuté. L’avatar cherche la description sémantique de l’application dans sa base de connaissances, puis extrait toutes les tâches atomiques la composant et envoie un message de découverte avec la performative REQUEST à tous ses voisins appartenant à son réseau social. En ce qui concerne les destinataires, ils testent qu’ils peuvent réaliser chacune des requêtes atomiques en fonction de leurs compétences et ils répondent à l’avatar initiateur avec un message de type PROPOSE contenant toutes tâches qu’ils peuvent réaliser avec leurs services concrets.

Après avoir reçu tous les réponses, l’avatar initiateur regroupe les services concrets de chacune des tâches abstraites dans des ensemble. Si pour une tâche donnée ou plusieurs, aucun service concret est trouvé dans le réseau social de l’avatar, ce dernier construit un nouveau message de type RESQUEST dont le contenu est juste constitué des tâches non satisfaites et l’identifiant du message est le même que l’identifiant de la conversation initiale et le transmettre à ses voisins. Les destinataires qui reçoivent le massage REQUEST avec le même identifiant de conversation, ne le traitent pas mais le transmettent à leur tour à leurs voisins pour augmenter le diamètre de recherche.

* 1. **Sélection et composition des services**

L’ensemble des services candidats résultant de la phase précédente possèdent des fonctionnalités similaires et des critères QoS différents. Cette phase permet de sélectionner, parmi l’ensemble de services candidats, les services les plus appropriés pour la composition selon leur propriétés QoS et qui respectent la contrainte temps réel locale pour chaque tâche abstraite. Dans notre travail, nous nous intéressons essentiellement au temps de réponse, la disponibilité ainsi que le coût de services comme paramètres QoS.

Notre approche consiste tout d’abord à filtrer les services candidats en utilisant la fluctuation des QoS, puis créer la composition de services en se basant sur l’algorithme, tout en intégrant les contraintes globales de l’application.

***5.3.1 Filtrage de services selon la fluctuation***

Dans le cas où le nombre de services candidats est très important, il s’avère nécessaire de filtrer ceux qui sont plus fiables par rapport à d’autres et les intégrer dans la composition. Et par conséquent, réduire le nombre de combinaisons possibles pour la composition.

5.3.1.1. Fluctuation de la qualité de service

Dans les approches traditionnelles de composition de services, le choix de services se base sur la moyennes de leurs QoS de leurs dernières invocations. Cependant, un facteur très important est ignoré dans ces approches qui est la variation (fluctuation) de leurs paramètres QoS. Un service possédant des paramètres QoS qui varient beaucoup n’est pas trop souhaitable parce que le résultat de la composition réel peut diverger, ce qui peut produire des compositions peu fiables. Pour calculer la variation des attributs QoS, un coefficient de fluctuationest utilisé.

Définition : Coefficient de variation

Le coefficient de variation [c49] représente le rapport entre l’écart-type et la moyenne des valeurs de l’historique des paramètres de QoS. Ce coefficient est fréquemment utilisé dans la théorie des probabilités pour mesurer la dispersion d’une variable autour de sa moyenne, plus la valeur de ce coefficient est élevée plus sa dispersion est grande. Formellement, il est défini comme suivant :

Soit qi un paramètre QoS et {qi1, qi2,…, qik} l’historique des valeurs de qi. Le coefficient de variation (CV) se calcule de la manière suivante :

CV= S(qi)/E(qi) \* 100% (5)

Tel que :

E(qi)= représente la moyenne de qi

représente l’écart type de qi

***5.3.2 Sélection des services***

La délimitation des destinataires de la requête de découverte aux voisins sociaux et le filtrage des services candidas selon la fluctuation permet de diminuer considérablement le nombre des services candidats et par conséquent le nombre de compositions possibles. Pour cela, nous avons choisis la stratégie de sélection locale gloutonne [c50]. Cette stratégie utilise la fonction score pour calculer l’utilité du service concret considéré tout en respectant les contraintes temps réel locales pour chaque tâche abstraite.

Le score de chaque service Sij représente la somme pondérée des valeurs QoS normalisées selon les préférences de développeur W={w1, w2,…,wn} comme suit :

Score(Sij)= (6)

Tel que :

wi : représente le poids affecté au critère QoS qi

Le principe général de notre algorithme est le suivant :

*INPUT : all sets of candidate services S={S1, S2, ...,Sn}, p*

*BEGIN*

*Discover the avatars list A of the system having at least an interest in common with Ai*

*for (Si ε S) do*

*for (Sij ε Si) do*

*Calculate the coefficient of variation of the Sij service response time values by using (5)*

*Calculate the score of each Sij by applying the score function by using (6)*

*endFor*

*Select the first p services with a small variation*

*Select the service with the highest score, which respects the local QoS constraint of the task Ti*

*If there are several services that have the same value of the maximum score, the service that better respects the real-time constraints is chosen*

*If no service respects the real-time constraints imposed on the application, an execution exception will be thrown*

*endFor*

*END*

1. **Protocole de coordination sociale pour la découverte et la sélection des services**

Tout à long de cette section, nous allons décrire les différentes phases des interactions entre les différents avatars durant le processus de découverte et de sélection des services en se basant sur les relations sociales qui les relient. Ce processus est réalisé grâce à un protocole d’interaction et un algorithme de négociation distribué basé sur les distances sociales entre avatars.

La propagation des différents types de messages (REQUEST, PROPOSE, etc) au sein des réseaux sociaux d’avatars représente une phase fondamentale dans le protocole de coordination à proposer. Dans ce contexte, on peut proposer trois approches pour le mise en place de protocole : approche par délégation, approche Rang-par-Rang et une approche hybride qui représente un mélange des deux autres approches.

**6.1- Approche par délégation**

Cette approche se base sur le principe que les nœuds participants ne reviennent jamais à l’avatar initiateur dans leurs décisions de propagation et ils prennent eux-mêmes l’initiative de continuer la recherche des services qui accomplissent les tâches et les requêtes qu’ils ne peuvent pas résoudre.

Phase 1 : initiation de la coordination

Lorsqu’un évènement survient dans l’état interne d’un avatar donné (un changement brusque de ses données par exemple) ou lors de la réception d’un message déclencheur de son environnement (demande d’un utilisateur), l’avatar raisonne sur sa base de connaissance pour être en mesure de savoir quelles actions sont requises qui correspondent à l’événement déclenché. Si l’avatar est capable d’accomplir la tâche requise tout seul, il la réalise directement sinon il initie une demande de collaboration. Pour cela, il commence par calculer son réseau social selon l’algorithme 1 puis il crée un message de type REQUEST avec la liste des tâches requises sous forme <r1,r2,..,rn> comme contenu du message, un temps de validité timeOut ainsi que un identifiant de conversation conversationID et il le diffuse à l’ensemble des avatars appartenant à son réseau social. L’avatar qui initie la collaboration joue le rôle de l’initiateur.

Phase 2 : traitement des requêtes REQUEST

Lorsqu’un avatar l’initiateur reçoive un message de type REQUEST, il extrait les tâches requises tâche par tâche et vérifie pour chacune s’il possède les services demandés pour l’accomplir. Si pour tâche donnée ri, plusieurs services concrets sont trouvés, l’avatar en question calcule leur fluctuation et pour les services et sélectionne les p premiers ayant une petite variation pour calculer leur score et sélectionne le meilleur parmi eux et l’associé à la tâche avec son score dans un message de type PROPOSE qu’il enverra comme réponse à l’initiateur tout en gardant le même identifiant de la conversation initiale.

Pour les tâches qui ne peuvent être satisfaites par l’avatar en question, ce dernier les met dans un nouveau message REQUEST avec le même conversationID et en mettant à jour le timeOut et le propage à ses voisins et par conséquent, il prend le rôle initiateur. Cet avatar se charge lui-même de recevoir les réponses de ses voisins, de filtrer les services proposés en retour et de sélectionner le meilleur parmi eux selon leur score pour le recommander à l’avatar qui lui a envoyé le REQUEST. L’avatar prend l’initiative de stocker le message dans son cache pour une utilisation ultérieure. Les voisins propagent également à leur tour les requêtes non satisfaites à leurs voisins ainsi de suite.

Si un avatar reçoit un message REQUEST avec un identifiant conversationID déjà existant dans son cache ou avec un timeOut déjà expiré, il ignore le message.

Phase 3 : Traitement des messages PROPOSE

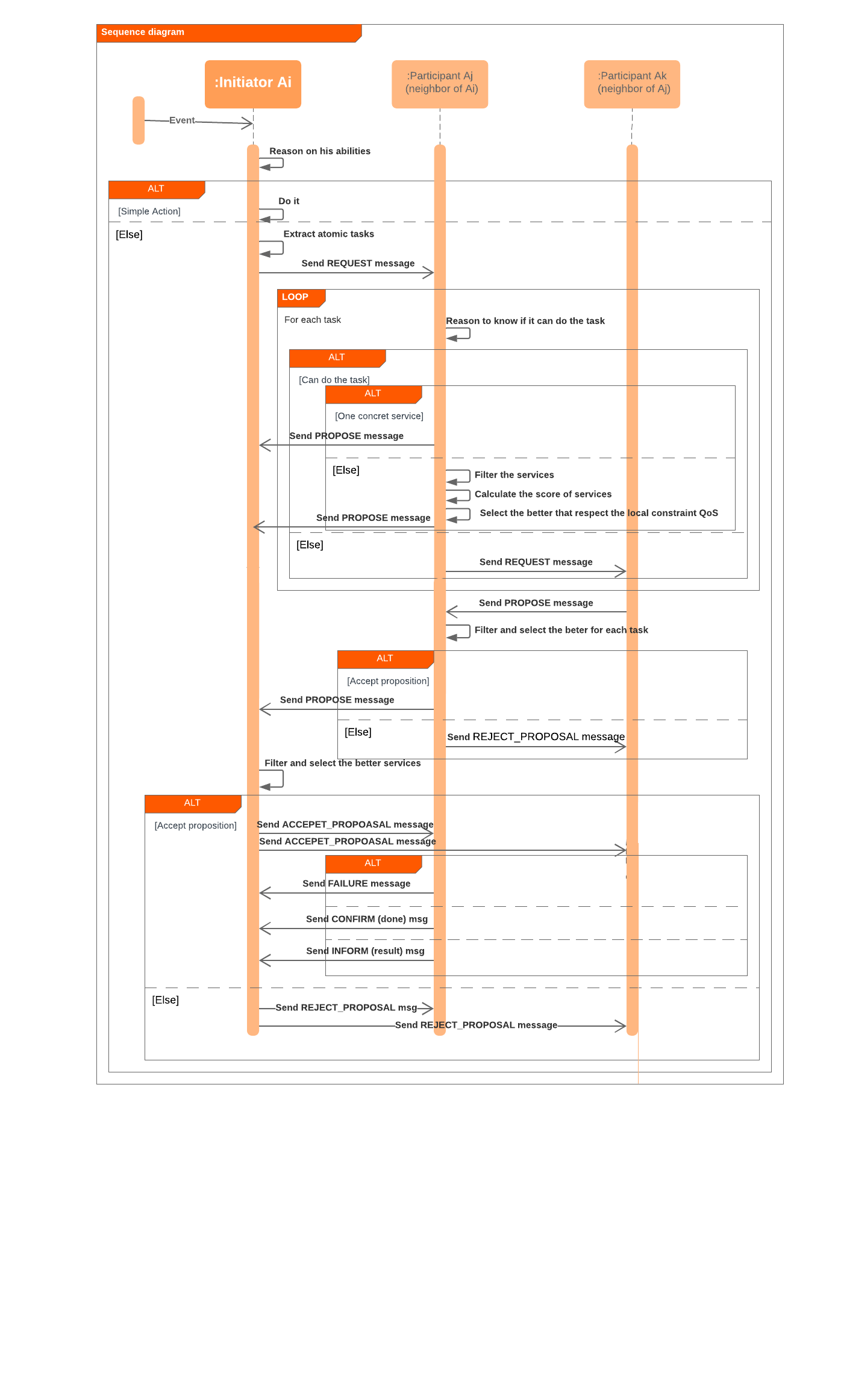
Les avatars expéditeurs des messages REQUEST restent en attente de réponses tant que l’ensemble de leurs requêtes ne sont pas satisfaites et le timeOut n’est pas expiré.

Lorsque l’attente est terminée grâce à la première condition pour une tâche donnée, l’avatar initiateur regroupe l’ensemble de ses services concerts et les trie selon leur score et sélectionne le meilleur parmi eux à condition que la composition résultante satisfasse les contraintes de la QoS locale exigées. L’avatar initiateur envoie un message de type REJECT\_PROPOSAL aux avatars pour lesquels les services concrets n’ont pas été sélectionnés. Concernant les services concrets sélectionnés par le premier avatar initiateur qui cherche à résoudre le problème initial, ce dernier envoie un message ACCEPT\_PROPOSAL à leurs avatars. Le premier avatar initiateur se charge également d’annoncer que la tâche est résolue pour arrêter le processus de la découverte, pour cela, il diffuse un message de type CANCEL à tous les avatars sollicités.

Par contre, si l’attente se termine à cause l’expiration de timeOut, une exception est enlevé et la composition ne peut pas être réalisée.

A la réception d’un message ACCEPT\_PROPOSAL, si l’avatar ne peut plus fournir le service il répond par un message de type FAILURE. Sinon il raisonne sur le service à fournir pour savoir s’il a des dépendances et il a besoin d’informations supplémentaire pour exécuter le service ; si c’est le cas, il reste en attente de ces informations sinon il exécute le service et renvoie le résultat à l’avatar initiateur.

Les principales étapes de ce processus sont illustrées dans le digramme de séquence suivant :



**6.2- Approche Rang-par-Rang**

Cette approche se base sur le fait que la probabilité que le but de coordination peut être satisfait par les avatars appartenant au réseau social de l’avatar initiateur est très élevée. Par conséquent, l’initiateur commence tout d’abord par solliciter ses voisins directs de premier rang, et selon les réponses qu’il a reçu, il décide de demander à ses voisins de solliciter leurs voisins (voisins de voisins de l’initiateur ie de deuxième rang) et ainsi de suite jusqu’à la satisfaction de l’objectif en question ou l’expiration de timeOut.

Phase 1 : initiation de la coordination

Cette phase se déroule de la même manière que l’approche par délégation. La seule différence est qu’un champ supplémentaire Rank est ajouté au message et à l’initiation ce champ est mis à 1. Ce champ représente le niveau de voisinage à solliciter.

Phase 2 : traitement des requêtes REQUEST

Lorsqu’un avatar l’initiateur reçoive un message de type REQUEST, il extrait les tâches requises tâche par tâche et vérifie pour chacune s’il possède les services demandés pour l’accomplir. Si pour tâche donnée ri, plusieurs services concrets sont trouvés, l’avatar en question calcule leur fluctuation et sélectionne les p premiers ayant une petite variation pour calculer leur score et sélectionne le meilleur parmi eux et l’associé à la tâche avec son score dans un message de type PROPOSE qu’il enverra comme réponse à l’initiateur tout en gardant le même identifiant de la conversation initiale.

Pour les tâches qui ne peuvent être satisfaites par l’avatar en question, ce dernier les met dans un nouveau message de type REFUSE avec le même conversationID et le renvoie à l’avatar initiateur.

Phase 3 : Traitement des messages PROPOSE et REFUSE

Lorsque l’initiateur reçoit les propositions de ces voisins directs ou indirects, il regroupe l’ensemble des services concerts de chaque tâche et les trie selon leur score et sélectionne le meilleur parmi eux à condition qu’il satisfasse les contraintes de la QoS locales exigées. L’avatar initiateur envoie un message de type REJECT\_PROPOSAL aux avatars pour lesquels les services concrets qui n’ont pas été sélectionnés. Pour les services concrets sélectionnés, l’initiateur envoie un message ACCEPT\_PROPOSAL à leurs avatars.

En ce qui concerne les messages de type REFUSE, l’initiateur extrait les différentes tâches qui n’ont pas pu être satisfaites et les encapsule dans un message REQUEST avec le même conversationID en augmentant le rang d’un niveau.

A la réception d’un message ACCEPT\_PROPOSAL, si l’avatar ne peut plus fournir le service il répond par un message de type FAILURE. Sinon il raisonne sur le service à fournir pour savoir s’il a des dépendances et il a besoin d’informations supplémentaire pour exécuter le service ; si c’est le cas, il reste en attente de ces informations sinon il exécute le service et renvoie le résultat à l’avatar initiateur.

A la réception d’un message FAILLURE par l’initiateur, et cherche dans son cache s’il existe des services concrets déjà proposés pour cette tâche. Si c’est le cas, il sollicite les avatars en question avec un message REQUEST de rang 1 pour cette tâche, sinon il l’encapsule avec les autres tâches non encore satisfaites (des messages REFUSE) dans un message REQUEST avec le rang en courant.

Phase 4 : étape de propagation

Lorsqu’un voisin reçoit un message REQUEST avec un rang supérieur à 1, il comprend qu’il doit seulement propager le message à ses voisins. Pour cela, il garde tous les champs de message comme reçu, et il change juste le champ Rank en diminuant la valeur reçue de 1.

Si un avatar reçoit le même message REQUEST pour la deuxième fois avec un identifiant conversationID et avec le même rang (ce qui le cas lorsque les avatars ont des amis en communs), il ignore le message. Idem, s’il reçoit un message avec un timeOut déjà expiré.

L’avatar qui a pris la mission de propager un REQUEST se charge lui-même de recevoir les réponses de ses voisins, de filtrer les services proposés en retour et de sélectionner le meilleur parmi eux selon leur score pour le recommander à l’avatar qui lui a envoyé le REQUEST. L’avatar prend l’initiative de stocker le message dans son cache pour une utilisation ultérieure.

Si le timeOut considéré pour la composition au niveau de l’initiateur s’exprime avant qu’elle soit satisfaite, une exception est enlevée et la composition ne peut pas être réalisée.

6.3- Approche hybride

Elle s’agit d’une approche dynamique qui fait intervenir explicitement les comportements des deux approches citées précédemment. Le principe de cette approche est de commencer tout d’abord par adopter l’approche Rang-par-Rang vu que la probabilité que les tâches soient résolues par les voisins direct est très élevée parce qu’ils possèdent des intérêts en communs et se trouvent dans des endroits très proches. Le nombre de rangs à considérés avant d’adopter l’approche par délégation est flexible.

Donc, l’initiateur commence par solliciter son réseau social avec message REQUEST de rang 1, et en dépendant de pourcentage de satisfaction des tâches élémentaires il décide de continuer avec l’approche par rangs ou de changer l’approche.

Si le taux de satisfaction est supérieur à 60%, il continue avec l’approche par rangs parce que le taux signifie que les voisins sont aptes à accomplir l’objectif à réaliser. Sinon, il opte pour l’approche par délégation.

**7- Résultats expérimentaux**

L’architecture proposée dans ce travail comprend un ensemble d’avatars implémentés en utilisant JADE et OM2M sur Eclipse. Ces avatars sont connectés via le réseau Wifi et ont l’habilité de communiquer entre eux d’une manière ad-hoc et asynchrone via le protocole FIPA-ACL. Chacun des avatars publie ses intérêts dans les pages jaunes de main container de JADE pour permettre aux autres avatars de trouver leurs voisins possédant des intérêts en commun.

Les expériences sont effectuées en utilisant un PC Windows 10 avec processeur Intel® Core i5 CPU @ 2.50 GHz avec 8 Go de RAM. Le serveur IN-CSE de OM2M qui héberge le main container de JADE est mis en place sur le PC. Dans nos expérimentations nous allons varier le nombre de passerelles MN-CSE de OM2M considérées entre 2 et 7 afin de voir effet de ce nombre sur le coût de notre approche notamment en termes de temps de réponse et le nombre de messages échangés dans le réseau d’avatars. Les passerelles sont représentées par des images dockers sur le même PC. Chaque image docker héberge à son tour un container JADE. Les avatars sont distribués d’une manière aléatoire entre le serveur et les passerelles considérées.

Pour évaluer et comparer les performances des trois approches citées précédemment, nous utilisons une application composée de 6 tâches abstraites et nous considérons le temps de réponse, le coût et la disponibilité comme contraintes de qualité de service possédant les coefficients de pondération suivant w1=0.4, w2=0.2 et w3= 0.4 respectivement parce que nous nous intéressons essentiellement au temps de réponse et à la disponibilité dans les applications IoT.

Chaque avatar est identifié par un identifiant unique AID sur le serveur OM2M et possède un nombre aléatoire de services concrets entre 2 et 8 services et un nombre d’intérêts entre 1 et 5. Dans cette expérimentation, on possède un ensemble de 20 services concrets S={S1,S2,…,S20}, un ensemble de 6 intérêts I={I1,I2,…,I6}, une ensemble de 5 utilisateur U={U1,U2,..,U5} . Pour chaque avatar, on lui associe un nombre de services concrets grâce à une fonction random sur l’intervalle [2,8]. Une fois le nombre de services est défini, un tirage au sort sans remise est effectué dans S. De la même manière, on associe pour chaque avatar un nombre aléatoire d’intérêts sur l’intervalle [1,5] et ensuite un tirage au sort sans remise dans l’ensemble I. Pour l’appartenance à un utilisateur donné, on affecte chaque avatar à un seul utilisateur de l’ensemble U aléatoirement. En ce qui concerne la localisation, la position de chaque avatar est générée aléatoirement autour de de la position (Latitude: 43.6043 Longitude: 1.4437) avec un diamètre de 15 mètres.

Après avoir défini pour chaque avatar l’ensemble de ses services concrets, ses intérêts, son propriétaire ainsi que sa localisation, nous procédons à la définition des paramètres de calcul de la distance sociale *α, β* et *δ* qui représente les coefficients de pondération des intérêts, de la localisation et l’appartenance au même utilisateur respectivement. Ces coefficients varient selon les scénarios IoT considérés, par exemple dans le cas d’une maison intelligente nous focalisons sur les intérêts et l’appartenance au même utilisateur pour notamment de point de vue sécurité et dans le cas des véhicules connectés la localisation est le critère le plus privilégie.

Au cours de nos expériences, nous cherchons à comparer l’évaluation du temps de réponse, du coût d’exécution ainsi que la charge des messages qui circulent dans le réseau en fonction de nombre d’avatars considérés N de nos trois approches. Avant cela, un autre paramètre important doit être également défini qui est k le nombre d’avatars du réseau social (diamètre de réseau social).

Nous allons varier les coefficients *α, β* et *δ* comme suivant en fixant k à 7 :

Configuration 1*: α* =0.3, *β* =0.1, *δ* = 0.6 et nous allons faire varier N en lui affectant les valeurs suivantes : N=20, N=30, N=40, N=50, N=60 et N=70

Configuration 2*: α* =0.3, *β* =0.6, *δ* = 0.1 et nous allons faire varier N en lui affectant les valeurs suivantes : N=20, N=30, N=40, N=50, N=60 et N=70

Configuration 3*: α* =0.6, *β* =0.2, *δ* = 0.2 et nous allons faire varier N en lui affectant les valeurs suivantes : N=20, N=30, N=40, N=50, N=60 et N=70

Configuration 4*: α* =0.4, *β* =0.4, *δ* = 0.2 et nous allons faire varier N en lui affectant les valeurs suivantes : N=20, N=30, N=40, N=50, N=60 et N=70

Dans chacune des trois premières configurations, nous appuyons sur un coefficient donné en particulier pour voir son effet sur la qualité de réseau social construit dans la découverte des services. En ce qui concerne la quatrième configuration, nous équilibrons les trois critères.

Nous allons faire varier le nombre d’avatars du système ainsi que k comme décrit dans les configurations suivantes en fixant les coefficients *α, β* et *δ à* 0.4, 0.4 et 0.2 respectivement :

Configuration 5 : N=20 et nous allons varier k en lui affectant les valeurs suivantes : k=3, k=6, k= 9 et k=10

Configuration 6 : N=30 et nous allons varier k en lui affectant les valeurs suivantes : k=4, k=8, k=10, k=15, et k=20

Configuration 7 : N=40 et nous allons varier k en lui affectant les valeurs suivantes : k=4, k=8, k=10, k=15, k=20 et k=30

Configuration 8 : N=70 et nous allons varier k en lui affectant les valeurs suivantes : k=8, k=12, k=17, k=20, k=30 et k=40

Configuration 9 : k=6 et nous allons faire varier N en lui affectant les valeurs suivantes : N=20, N=30, N=40, N=50, N=60 et N=70

Configuration 10 : K=6, N=40 et nous allons faire varier le nombre de passerelles considérées entre 2 et 7.

Analyse des résultats : a faire après l’implémentation

Bibliographie

[1] S. Oh, B. On, E.J. Larson, and D. Lee. BF\*: Web Services Discovery and Composition as Graph Search Problem, 6-8, eTechnology, e-Commerce, and e-Services, IEEE International Conference on, 784-786, 2005.

[2] R. Kazhamiakin and M. Pistore, A Parametric Communication Model for the Verification of BPEL4WS Compositions, Formal Techniques for Computer Systems and Business Processes, 318- 332, Trento, Italy. 2005.

[c1] L. Atzori, A. Iera, and G. Morabito, “Understanding the Internet of Things: definition, potentials, and societal role of a fast evolving paradigm,” Ad Hoc Networks, vol. 56, no. Supplement C, pp. 122–140, 2017.

[c2] O. Vermesan, P. Friess, P. Guillemin, H. Sundmaeker, M. Eisenhauer, K. Moessner, F. L. Gall, and P. Cousin, Internet of Things Strategic Research and Innovation Agenda, 2013.

[c3] G. Fortino, A. Guerrieri, W. Russo, and C. Savaglio, “Middlewares for smart objects and smart environments: overview and comparison,” in Internet of Things Based on Smart Objects. Springer, 2014, pp. 1–27.

[c4] Scarrone, ETSI TC Smart M2M Vice-Chairman, o. S. V. C. (2015). M2m/iot standards in etsi and onem2m. Telecom Italia, ALMANAC Workshop, Copenhagen.

[c5] M. Amadeo, O. Briante, C. Campolo, A. Molinaro, and G. Ruggeri, “Information-centric networking for M2M communications: Design and deployment,” Computer Communications, vol. 89, pp. 105–116, 2016.

[c7] Bonomi, Flavio, et al. "Fog computing and its role in the internet of things." Proceedings of the first edition of the MCC workshop on Mobile cloud computing. ACM, 2012.

[c10] N. Temglit, H. Aliane, M. Ahmed Nacer. 2008. «Un modèle de composition des services Web sémantiques»; Volume 11-Numéro spécial CARI2008 -Marc Kokou Assogba.

[c8] A. Socievole and S. Marano, “Exploring user sociocentric and egocentric behaviors in online and detected social networks,” in 2012 2nd Baltic Congress on Future Internet Communications, April 2012, pp. 140–147.

[c9] Benatallah B., M. Dumas, M. Fauvet, and F. Rabhi, 2003. Towards patterns of web services composition, Patterns and skeletons for parallel and distributed computing, Springer Verlag, 2003.

[c11] Jayadev M., and C. William, 2007. Computation Orchestration: A Basis for Wide-area Computing, Journal of Software and Systems Modeling, vol. 6, n° 1, p. 83-110.

[c12] R. Kazhamiakin and M. Pistore, A Parametric Communication Model for the Verification of BPEL4WS Compositions, Formal Techniques for Computer Systems and Business Processes, 318- 332, Trento, Italy. 2005

[c13] Srivastava B, Koehler J (2003) Web service composition—current solutions and open problems. In: Proceedings of ICAPS’03 workshop on planning for web services, Trento, Italy, 2003, pp 28–35 4. [c14] Agarwal V, et al (2008) Understanding approaches for web service composition and execution. In: Proceedings of the 1st Bangalore annual compute conference, India, 2008, pp 1–8 5. Alamri A, Eid [c15] M, Saddik AE (2006) Classification of the state-ofthe-art dynamic web services composition techniques. Int J Web Grid Serv 2(2):148–166

[c16] Bourdon J., 2007. Multi-agent systems for the automatic composition of semantic web services in dynamic environments, Rapport de master, École des Mines de Saint Etienne - G2I & Université Joseph Fourier.

[[c17] D.Wu, E.Sirin, B.Parsia, J.Hendler & D.Nau (2003) “Automatic web services composition usingSHOP2”, In: Proceedings of ICAPS'03Workshop on Planning for Web Services, Trento, Italy.

[c18] S. McIlraith. & T.C.Son (2002) “ Adapting Golog for Composition of Semantic Web Services”, KR- 2002, *France, 2002.*

[c19] S. McIlraith, S.Son & H. Zeng (2001) “Semantic Web Services”, IEEE Intelligent Systems, 16(2): pp46– 53.

[c20] M. Sheshagiri, M. desJardins & T. Finin (2003) “A Planner for Composing Services Described in DAML-S”, In Proc. Of Workshop on Web Services and Agent-based Engineering - AAMAS’03.

[c21] Pistore M., F. Barbon, P. Bertoli, D. Shaparau, and P. Traverso, 2004. Planning and MonitoringWeb Service Composition, Proceedings of the International Conference on Artificial Intelligence, Methodology, Systems, and Applications, p. 106-115.

[c22] B. Medjahed, A. Bouguettaya, and A.K. Elmagarmid, "Composing web services on the semantic web," Journal of Very Large Data Bases, vol. 12, no. 4, pp. 333-351, 2003.

[c23] J. Peer, "A PDDL based tool for automatic Web service composition," Proc. Second Int’l Workshop. Principles and Practice of Semantic Web Reasoning (PPSWR), pp. 149-163, 2004.

[c24] S. Narayanan, and S. McIlraith, "Simulation, verification and automated composition of web services," Proc. 11th Int’l Conf.

World Wide Web (WWW), pp. 77-88, 2002.

[c25] El Falou, Mohamad. Contributions à la composition dynamique de services fondée sur des techniques de planification et diagnostic multi-agents. Diss. Université de Caen, 2010.

[c26] Sheshagiri M., M. Desjardins, and T. Finin, 2003. A planner for composing services described in DAML-S. In Proceedings of the AAMAS Workshop on Web Services and Agent-based Engineering

[c27] Huhns, Michael N., et al. "Research directions for service-oriented multiagent systems." IEEE Internet Computing 9.6 (2005): 65-70.

[c28] Guitton, Julien, and H. Fiorino. "Planification multi-agent pour la composition dynamique de services web." Intelligence (2006).

[c29] Peer, Joachim. "A PDDL based tool for automatic web service composition." International Workshop on Principles and Practice of Semantic Web Reasoning. Springer, Berlin, Heidelberg, 2004.

[c30] Guitton, Julien, and H. Fiorino. "Planification multi-agent pour la composition dynamique de services web." *Intelligence* (2006).

[c31] Wu D., B. Parsia, E. Sirin, J. Hendler, and D. Nau, 2003. Automating DAML-S web services composition using SHOP2. In ISWC’03.

[c32] Edi, Kouassi Hilaire. Affectation flexible des ressources dans la planification des activités industrielles: prise en compte de la modulation d'horaires et de la polyvalence. Diss. Toulouse 3, 2007.

[c33] Medjahed, Brahim, Athman Bouguettaya, and Ahmed K. Elmagarmid. "Composing web services on the semantic web." *The VLDB journal* 12.4 (2003): 333-351.

[c46] O'Brien, Paul D., and Richard C. Nicol. "FIPA—towards a standard for software agents." *BT Technology Journal* 16.3 (1998): 51-59.

[c34] Pistore M., P. Traverso, and P. Bertoli, 2005b. Automated Composition of Web Services by Planning in Asynchronous Domains, Proceedings of the International Conference on Planning and Scheduling, p. 2-11.

[c35] D. Pellier and H. Fiorino. Un modèle de composition automatique et distribuée de services web par planification, Revue d'Intelligence Artificielle ,v23,13-46, 2009.

[c36] B. Medjahed, A. Bouguettaya, and A.K. Elmagarmid, "Composing web services on the semantic web," Journal of Very Large Data Bases, vol. 12, no. 4, pp. 333-351, 2003

[c37] J. Peer, "A PDDL based tool for automatic Web service composition," Proc. Second Int’l Workshop. Principles and Practice of Semantic Web Reasoning (PPSWR), pp. 149-163, 2004

[c38] Z. Maamar, S. K. Mostefaoui, and H. Yahyaoui, "Toward an agentbased and context-oriented approach for Web services composition," IEEE Transactions on Knowledge and Data Engineering, vol.

17, no. 5, pp. 686-697, 2005.

[c39] R. Kazhamiakin and M. Pistore, A Parametric Communication Model for the Verification of BPEL4WS Compositions, Formal Techniques for Computer Systems and Business Processes, 318- 332, Trento, Italy. 2005.

[c40] -G. Silva, F. Pires, and V. Sinderen. An Algorithm for Automatic Service Composition, 1 st International Workshop on Architectures, Concepts and Technologies for service Oriented Computing. pp. 65-74, Barcelona Spain. July 2007.

[c41] S. Oh, B. On, E.J. Larson, and D. Lee. BF\*: Web Services Discovery and Composition as Graph Search Problem, 6-8, eTechnology, e-Commerce, and e-Services, IEEE International Conference on, 784-786, 2005.

[c42] Chaib, Aouatef, Imane Boussebough, and Allaoua Chaoui. "Adaptive service composition in an ambient environment with a multi-agent system." *Journal of Ambient Intelligence and Humanized Computing* 9.2 (2018): 367-380.

[c43] W3C, WEB OF THINGS AT W3C available at <https://www.w3.org/WoT/> , (2017)

[c6] LAAS-CNRS, OM2M connecting things available at <https://www.eclipse.org/om2m/>, (2015)

[c44] L. Atzori, A. Iera and G. Morabito, “SIoT: Giving a Social Structure to the Internet of Things,” Communications Letters, vol. 15, no. 11, pp. 1193-1195, Nov. 2011.

[c45] Tietz, Vincent, et al. "Task-based recommendation of mashup components." *International Conference on Web Engineering*. Springer, Berlin, Heidelberg, 2011.

[c46] O'Brien, Paul D., and Richard C. Nicol. "FIPA—towards a standard for software agents." *BT Technology Journal* 16.3 (1998): 51-59.

[c47] Mei, Alessandro, et al. "Social-aware stateless forwarding in pocket switched networks." *2011 Proceedings IEEE INFOCOM*. IEEE, 2011.

[c48] Sheng, Quan Z., et al. "Web services composition: A decade’s overview." *Information Sciences* 280 (2014): 218-238.

[c49] Kabache, Nassim, Ahmed Khaled Zaidi, and Zoubeyr Farah. *Sélection et Composition de Services Web avec Respect des Contraintes d'Utilisateur et Qualité de Service*. Diss. Université Abderrahmane Mira-Bejaia, 2018.

[c50] Alrifai, Mohammad, and Thomas Risse. "Combining global optimization with local selection for efficient QoS-aware service composition." *Proceedings of the 18th international conference on World wide web*. ACM, 2009.