

Analyse de solutions pour la mise en oeuvre d'applications DDS sur réseaux grande distance

Akram Hakiri, Pascal Berthou, Slim Abdellatif, Michel Diaz, Thierry Gayraud

▶ **To cite this version:**

Akram Hakiri, Pascal Berthou, Slim Abdellatif, Michel Diaz, Thierry Gayraud. Analyse de solutions pour la mise en oeuvre d'applications DDS sur réseaux grande distance. CFIP 2012, Oct 2012, Anglet, France. pp.105. hal-00747432

HAL Id: hal-00747432
https://hal.science/hal-00747432

Submitted on 31 Oct 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Analyse de solutions pour la mise en oeuvre d'applications DDS sur réseaux grande distance

Akram Hakiri[1,2], Pascal Berthou[1,2], Slim Abdellatif[1,2], Michel Diaz[1,2], Thierry Gayraud[1,2]
[1]CNRS ; LAAS ; 7 Avenue du Colonel Roche, F-31077 Toulouse Cedex 4, France
[2]Univ de Toulouse ; UPS, INSA ; F-31077 Toulouse Cedex 4, France
Email : {Hakiri, Berthou, Slim, Diaz, Gayraud}@laas.fr

***Résumé*—Stimulées par la croissance de leur utilisation tant dans le domaine militaire que dans le domaine civil, les applications DDS connaissent une forte croissance dans les réseaux à large envergure. Cependant, certaines caractéristiques de ces réseaux comme les longs délais, la limitation de la bande passante et l'interdiction de l'utilisation de certains protocoles dans les réseaux opérationnels comme le multicast réseau, rendent difficile voire impossible la mise en place d'applications DDS. Nous présentons dans ce travail une proposition pour l'interconnexion d'applications DDS sur des réseaux distants : dans un premier temps, nous analysons la possibilité d'utiliser le service de routage de la couche middleware DDS à travers la mise en oeuvre d'un "pont-fédéré" DDS permettant d'interconnecter des domaines DDS distincts sur des réseaux étendus. Ensuite, nous proposons le concept de "Proxy DDS" qui en comparaison avec la solution précédente permet de réduire le volume de trafic échangé sur le réseau.**

***Index Terms*—DDS, Service de Routage, Pont-Fédéré, Proxy DDS, Multicast.**

I. Introduction

De plus en plus d'applications DDS (Data Distribution Service) [1] sont déployées sur des réseaux étendus comme l'Internet pour bénéficier du service de distribution de données qu'offre le middleware DDS comme par exemple les applications Peer-to-Peer [2], les applications multimedia [3] [4], les application de contrôle/commande [5], etc.

En effet, DDS présente une architecture producteur/consommateur qui permet aux applications de communiquer tout simplement, par publication, l'information dont elles disposent et de souscrire à l'information dont elles ont besoin. Le modèle conceptuel de la distribution de données dans DDS repose sur une abstraction d'un espace global de données typées partagé entre des processus applicatifs producteurs (publisher) qui produisent certaines de ces données et des processus consommateurs (subscriber) qui en consomment certaines [6]. Un participant peut publier et s'abonner simultanément à des informations identifiées par nom de Topic. A partir de ces déclarations d'intention (à produire ou à consommer un topic), le middleware DDS met automatiquement en relation les producteurs et consommateurs qui se partagent le même topic ; DDS se charge alors de livrer les différentes valeurs (également appelés échantillons) produites aux consommateurs qui se sont déclarés intéressés par le topic.

Une application souhaitant produire un type de données doit attacher un DataWriter à un publisher. Le publisher est en charge de la distribution effective des données alors que le DataWriter est le moyen pour l'application d'indiquer au publisher la présence d'un nouvel échantillon. Un processus applicatif souhaitant transmettre un nouvel échantillon utilise donc un DataWriter pour activer au niveau du publisher la dissémination de la valeur de la donnée qui se fera en fonction de la QoS prévue.

Du côté consommateur, un subscriber réceptionne les échantillons publiés des Topics auxquels il a souscrit. L'application utilise de son côté un DataReader (un par type de données) pour récupérer les échantillons reçus. Une application souhaitant consommer un type de données (spécifié par un topic) doit donc attacher un DataReader à un subscriber. Ce dernier engage, pour chaque type de données consommées, une phase de souscription qui lui permet de s'associer aux publishers respectifs. Cette souscription peut concerner tous ou partie des échantillons d'un topic en jouant sur les caractéristiques d'un topic, à savoir le nom, la clef éventuelle (souscription à une instance) ou le contenu/valeur des échantillons.

DDS fournit la possibilité de spécifier pour chaque type de données plusieurs paramètres de QoS (débit, latence, durée de validité,...), qui permettent de réaliser une application distribuée basée sur le besoin et la disponibilité de chaque type de données (nous ne traitons pas les QoS DDS dans ce travail).

L'interconnexion d'applications DDS à travers des réseaux distants nécessite une communication multicast réseau pour réduire le volume de trafic échangé. En conséquence, pour généraliser l'intégration d'applications DDS sur des réseaux étendus, il peut-être nécessaire d'ajouter d'autres domaines DDS distants. La possibilité d'utiliser plusieurs domaines permet de séparer les données circulant dans le réseau de celles d'autres applications.

Cependant, l'intégration de DDS sur des réseaux distants est confronté à de nombreux problèmes de déploiement. En effet, les messages de découverte des participants qui devront être envoyés en multicast sont bloqués, le multicast IP est bloqué au niveau des routeurs de bordure pour des raisons de sécurité et de performance de routage. En plus, DDS par défaut ne permet pas l'échange de Topics partagés avec d'autres domaines DDS. Il est alors impossible de réaliser une communication entre deux domaines DDS distincts.

L'objectif de ce papier est d'analyser les solutions existantes permettant la mise en place d'applications DDS s'exécutant sur des noeuds distants, basées sur le service de routage [7] proposé par la couche middleware NDDS [8], et ensuite de proposer le concept de proxy DDS qui en comparaison aux solutions existantes réduit le volume de trafic échangé sur le réseau grande distance.
Ce papier est organisé comme suit : la section II traite l'utilisation du service de routage de DDS dans le pont-fédéré pour résoudre certaines de ces questions. La section III présente le proxy DDS que nous avons proposé pour résoudre les problèmes rencontrés lors du déploiement du "pont-fédéré". La section IV décrit la plateforme de test et évalue les performances de la solution proposée. La section V présente la conclusion de ce papier et décrit les perspectives de ce travail.

II. Analyse des solutions basées sur le pont-fédéré

Dans cette section, nous présentons la première solution qui consiste à enrichir le service de routage de DDS pour implémenter le pont-fédéré. Le pont-fédéré permet découpler spatialement les entités DDS situées sur des réseaux distants afin de déployer les applications DDS existantes de manière transparente. Il permet d'une part l'interconnection des domaines DDS distincts situés sur des réseaux étendus (Domain-Bridge) et d'identifier les différents Topics (par leurs nom, clé et type) échangés dans les différents espaces de partage de données (GDS) en jouant le role d'un pont de Topics pour assurer l'échange des données entre différentes applications DDS.

A. Présentation du DDS Routing Service (DDS-RS)

Le service DDS-RS propose des fonctions de bridge 'DDS-to-DDS' entre applications pour permettre la transformation de données. Ceci permet d'utiliser les applications DDS sans les modifier même si elles ont été développées en utilisant des définitions d'interface incompatibles. C'est souvent le cas lors de l'intégration d'applications existantes dans des systèmes développés indépendamment. Il permet également d'offrir les fonctionnalités suivantes :

- ✠ Domain Bridging : permet de créer un pont entre des domaines DDS. Le middleware seul ne permet pas d'assurer un échange de Topic entre deux applications DDS qui utilisent des espaces de partages globaux différents. Ce service de routage assure alors la communication entre un DataWriter et un DataReader situés dans des domaines DDS différents.
- ✠ Topic Bridging : Le service classique de distribution de données se fait par l'échange d'une même instance d'un Topic entre un DataWriter et DataReader, alors que ce service permet aux différentes entités d'échanger des Topics qui portent des noms et des instances différents. Il permet donc d'assurer une interopérabilité entre des Topics différents.
- ✠ Data Transformation : Ce service est capable de transformer des Topics en d'autres. En effet, on peut le configurer de telle sorte qu'il consomme certains Topics et les retransmette en changeant des valeurs d'échantillons de ces Topics. Cette fonction peut être utilisée pour empêcher les données privées d'être publiées dans un réseau étendu.
- ✠ Configuration XML : Le comportement du "DDS-RS" peut être configuré à l'aide de fichiers XML pour faciliter sa manipulation.
- ✠ Full QoS Configuration : Paramètre le service DDS-RS pour la prise en compte de la qualité de service par la configuration de profils de QoS dans des fichiers XML.

Pour permettre à des applications DDS de communiquer sur des domaines DDS distincts, nous introduisons la notion de "pont-fédéré" pour se référer à un pont DDS qui utilise le service DDS-RS. Ce pont-fédéré est réalisé avec les APIs de DDS sans introduire de nouveaux protocoles qui rajoutent de la complexité, sans pour autant perdre l'avantage de l'interopérabilité souhaitée. Dans le suite de cette section, nous nous focalisons sur les deux possibilités de configuration du pont-fédéré, à savoir le mode "Domain Bridging" et le mode "Topic Bridging".

B. Solution basée sur le Domain-Bridge

1) Principe de Fonctionnement: La solution basée sur le "Domain-Bridge" permet d'interconnecter des espaces de partage de données DDS différents. Elle consiste à définir de manière statique les différents participants en spécifiant leurs adresses IP respectives dans une variable d'environnement appelée `"NDDS_DISCOVERY_PEERS"`. En conséquent, le DDS-RS regarde dans cette variable toutes les adresses des machines distantes et envoie les messages *Discovery* en unicast à chaque participant. Ce pont, montré sur la Figure 1, se comporte comme un unique fédéré différent des autres applications DDS. Il s'assure que tout évènement ayant lieu sur un simulateur est propagé vers les autres fédérés.

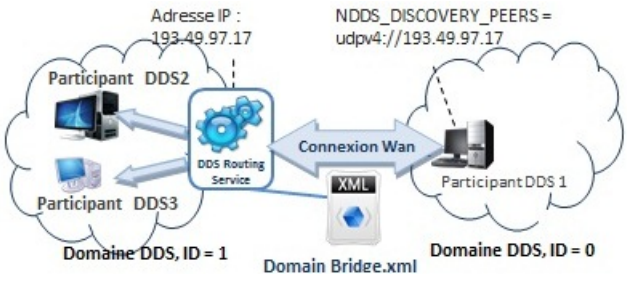


FIGURE 1: Mise oeuvre du service de routage de NDDS

Comme le montre la Figure 1, le participant DDS1 est configuré de telle sorte qu'il puisse découvrir le pont-fédéré pour assurer la distribution des Topics entre le participant DDS1 sur le domaine DDS0 et les participants DDS2 et DDS3 situés dans le domaine DDS0. Ainsi, le participant DDS1 transfère toutes les données présentes dans le domaine DDS 0 vers le domaine DDS 1.
Le pont-fédéré peut découvrir les adresses IP des différentes

```
<routing_service name="defaultBothWays">
   <domain_route name="TwoWayDomainRoute">
        <participant_1>
           <domain_id>0</domain_id>
        </participant_1>
        <participant_2>
           <domain_id>1</domain_id>
        </participant_2>
         <session name="Session1">
            <auto_Topic_route name="AllForward">
               <publish_with_original_info>true</publish_with_original_info>
               <input participant="1">
                  <allow_Topic_name_filter>*</allow_Topic_name_filter>
    <allow_registered_type_name_filter>*</allow_registered_type_name_filter>
                  <creation_mode>ON_DOMAIN_AND_ROUTE_MATCH</creation_mode>
               </input>
               <output>
                  <allow_Topic_name_filter>*</allow_Topic_name_filter>
    <allow_registered_type_name_filter>*</allow_registered_type_name_filter>
                  <creation_mode>ON_DOMAIN_AND_ROUTE_MATCH</creation_mode>
               </output>
            </auto_Topic_route>
         </session>
    </domain_route>
 </routing_service>
```

FIGURE 2: Configuration du service de routage pour connecter de domaines DDS différents

machines par la configuration d'un fichier XML. La Figure 2 nous permet de mieux comprendre la configuration et le principe du service. Ce fichier contient en plus de la description des différents participants, une description des Topics à transférer durant la session et le comportement à suivre pour les router vers différents domaines DDS. Les sessions DDS contiennent (entre autres) les informations suivantes : pour la Session1 nous définissons le domaine d'entrée comme "Input" suivi du participant qui va générer les différents Topics, et le domaine de sortie comme "Output", vers lequel les Topics vont être redirigés. Pour la Session2 [1], nous définissons les opérations dans le sens inverse, c'est à dire les éléments nécessaires pour transférer les Topics présents dans le domaine DDS1 vers le domaine DDS0. Cette opération nous permet donc d'assurer une communication bidirectionnelle entre les deux domaines. De cette manière, le pont-fédéré basé sur le "Routing Service" permet alors d'assurer la découverte des entités et l'échange des Topics entre deux domaines DDS différents.

2) Analyse de la solution dans le cas multi-domaine DDS: Pour analyser cette solution, nous nous focalisons sur la configuration du pont-fédéré comme domain-bridge illustrée sur la Figure 3. En effet, Le pont-fédéré est lancé et exécuté sur la machine le participant DDS2 pour se comporter à la fois comme consommateur et pont avec une configuration correspondant à "Domain Bridge" avec deux domaines DDS distincts sur deux réseaux IP différents.

Nous avons considéré une application DDS lancée sur le participant DDS1 dans le domaine DDS0 qui contient un

1. N'est montrée dans la Figure 2 pour des raisons de l'espace

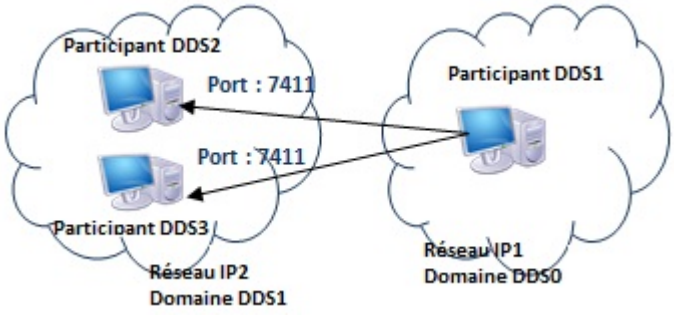


FIGURE 3: Test du pont-fédéré avec le Domain Bridge

producteur qui émet un grand nombre de messages (en DDS cela correspond à un producteur du Topic "Message" qui contient juste une variable clé : "UserID" de type long et une autre variable pour le contenu du message "Content" de type séquence de caractère) et de l'autre côté dans le domaine DDS1 deux récepteurs (participant DDS2 et participant DDS3) qui se souscrivent à ce même Topic "Message".

Nous avons constaté que les Topics envoyés par le participant DDS1 depuis le domaine DDS0 sont reçus par toutes les machines consommatrices (le participant DDS2 et le participant DDS3) situées dans le domaine DDS1. Cependant, cette opération d'envoi mènent à la création de deux sockets sur le port 7413 et sur le port 7411 depuis la machine source (le participant DDS1) vers le pont-fédéré comme le montre la Figure 3. En conséquence, cette solution reste partielle (même si elle permet d'acheminer le trafic DDS entre deux domaines DDS distincts se trouvant sur deux réseaux IP différents) car

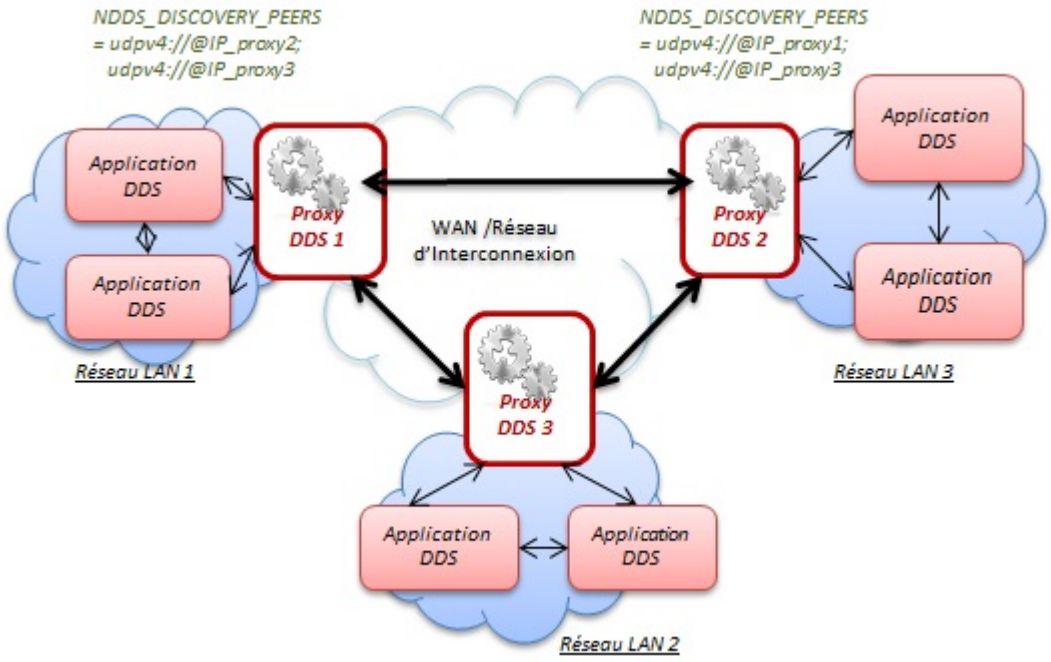


FIGURE 4: Déploiement de Proxies DDS dans un réseau grande distance multi-domaines IP

le flux envoyé par le producteur des Topics est dupliqué par le nombre de machines consommatrices découvertes par le service "Discovery" de DDS.

III. LE PROXY DDS

A. Introduction

Dans la section précédente, nous avons montré que l'utilisation du service de routage de DDS dans le pont-fédéré ne permet pas d'optimiser les flux échangés par les applications DDS. Dans cette section, nous proposons le proxy DDS, qui en plus fonctionnalités réalisées par le pond-fédéré, permet d'une part la réduction le volume de trafic DDS et d'autre part répond à la problématique de communication en multicast. Nous détaillons dans la suite de cette section les services offerts par le proxy DDS ainsi que son principe de fonctionnement.

B. Services offerts par le proxy DDS

A l'inverse du pont-fédéré qui résulte d'une configuration du DDS Routing Service (DDS-RS), le proxy DDS est une solution logicielle que nous avons implémentée en C++ et indépendante du DDS-RS. Ce paragraphe va détailler les services offerts par le proxy DDS.

1) Relier différentes applications DDS à travers un réseau WAN: Le proxy DDS permet de relier des applications DDS situées sur des réseaux grande distance. Il relaie les messages de tous les participants de son domaine IP aux autres proxies DDS du réseau. Cette configuration nécessite alors de déployer autant de proxies DDS que de domaines IP sur une machine à côté du routeur de bordure. L'architecture de déploiement permettant d'assurer la communication entre les différents proxies DDS distants est montrée par la Figure 4. Elle nécessite de plus configurer le service de découverte ("*NDDS_DISCOVERY_PEERS*") avec toutes les adresses IP des différents proxies.

2) Utilisation d'un seul domaine DDS: Le proxy DDS est composé de différentes entités qui communiquent dans un seul domaine DDS, mais pouvant être dans des domaines IP différents. Il consomme alors tous les Topics auxquels il est abonné dans son réseau IP et retransmet les Topics destinés aux consommateurs distants vers les autres proxies DDS distants respectifs. Les autres proxies DDS vont redistribuer tous ces Topics reçus dans leur domaine, et à travers ce mécanisme toutes les applications DDS situées dans des réseaux différents pourront partager le même domaine DDS et considérerons le réseau WAN comme support d'un espace de partage global. La Figure 5 montre le déploiement de cette architecture.

C. Principe de fonctionnement du proxy DDS

Le Proxy DDS est une application qui se lance sur une machine et s'exécute explicitement comme un service DDS. Il adopte comme technologie de base le middleware DDS en utilisant les API et les services nécessaires du RTI DDS. Ce proxy est composé de toutes les entités de base de DDS, et son développement et son déploiement dépendent du besoin de l'application mise en oeuvre. Cela nécessite une spécification de tous les Topics échangés entre les différentes entités de l'application. Pour mieux comprendre le fonctionnement du proxy DDS, nous présentons les différentes étapes du déroulement sur un exemple.

Mise en place d'un Proxy DDS: Nous considérons une application qui souhaite émettre un seul Topic "**T1**" d'un réseau IP à un autre à travers une connexion gérée comme un seul et unique domaine DDS, et deux proxies installés à l'extrémité de chaque réseau. Comme le montre la Figure 6, cette application

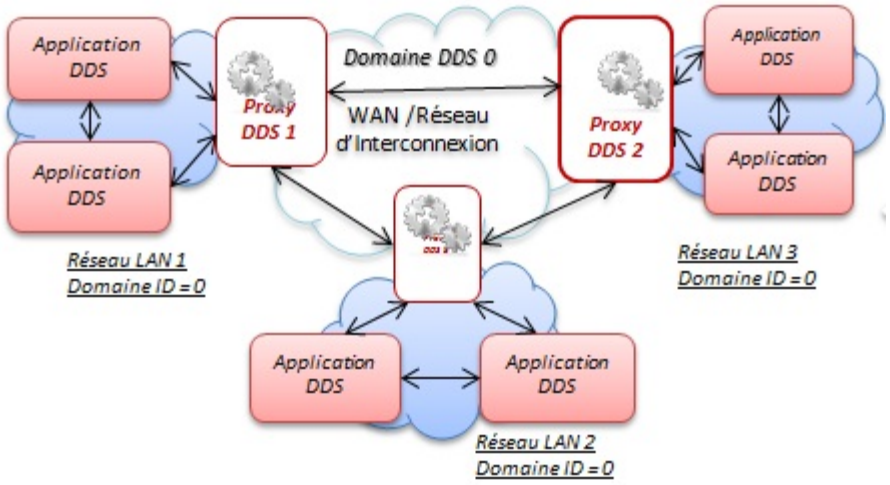


FIGURE 5: Déploiement du proxy DDS sur un seul domaine DDS

dispose d'un producteur géré par un DataWriter de "**T1**" et deux consommateurs gérés par deux DataReaders du même Topic.
Les étapes illustrées sur cette figure montrent le scénario d'échange de données entre les différents participants, qui se déroule comme suit :

- Étape 1 : Le producteur de l'application publie dans le réseau local 2 des échantillons du Topic **T1** en multicast.
- Étape 2 : Le proxy DDS1 s'inscrit au Topic **T1** et consomme tous les échantillons présents dans le domaine DDS1 (le Subscriber de **T1** consomme uniquement les échantillons publiés en multicast).
- Étape 3 : A l'aide d'une file d'attente, le proxy DDS1 redistribue le Topic **T** consommé dans le domaine DDS1 en le renommant en un autre Topic **T1'** afin d'éviter que **T1** soit consommé une autre fois après avoir été redistribué dans le domaine DDS1, et par la suite éviter une boucle locale de consommation et production des échantillons du même Topic au niveau du Proxy DDS1.
- Étape 4 : De l'autre côté du réseau, un autre proxy DDS2 commence à consommer le Topic **T1'** dès sa présence dans le domaine DDS2 (les échantillons entre les deux proxies sont échangés en Unicast puisqu'ils sont transmis à travers le réseau WAN), ensuite les proxies transforment **T1'** en **T** et le redistribue en multicast dans le réseau local 1.
- Étape 5 : Finalement, les consommateurs de l'application reçoivent en multicast tous les échantillons publiés par le proxy de leur réseau ce qui correspond au Topic **T1** envoyé par le producteur de l'application dans le réseau local 2.

IV. ETUDE DE CAS

Après avoir décrit le principe de fonctionnement du pont-fédéré et du proxy DDS, nous présentons dans cette section une étude de cas de mise en oeuvre d'une application de simulation distribuée sur DDS. Cette application consiste

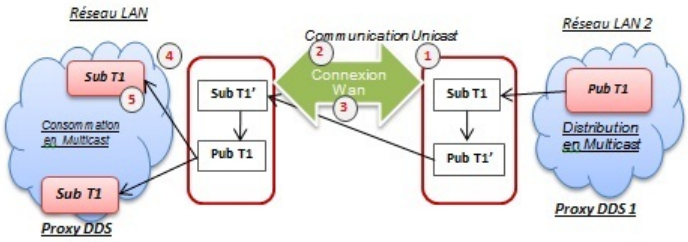


FIGURE 6: Interconnexion des deux proxies DDS

à développer une plateforme qui permet de mettre des simulateurs terrestres sur DDS en réseaux grande distance et de partager le même champ d'opérations virtuel (par exemple une ville, un aéroport, un abord de stade, etc.) et ainsi répondre à une demande de formation collective dans un environnement virtuel le plus proche possible de la réalité. Nous décrivons la plateforme de test et les résultats expérimentaux pour la validation de nos prototypes.

A. Plateforme de test

Une plateforme d'expérimentation a été conçue et mise en place au LAAS-CNRS. Cette plate-forme, appelée "Laasnextexp" [9] et montrée à la Figure 7, est indépendante du réseau du LAAS et elle est raccordée directement à Renater [2], et à travers Renater aux autres réseaux européens de la recherche. De cette manière, Laasnextexp permet de réaliser des tests dans un environnement réel multi-technologies, multi-domaines ainsi que dans un environnement émulé.
La plate-forme globale est composée de trois domaines avec des adresses publiques interconnectés par le biais de plusieurs équipements réseaux : trois routeurs JuniperM7i, six commutateurs (switches) CISCO Catalyst 2960G, 2970 et WS-C6504-E. Elle peut fonctionner en IPv4 ou IPv6 et accéder

2. http:www.renater.fr

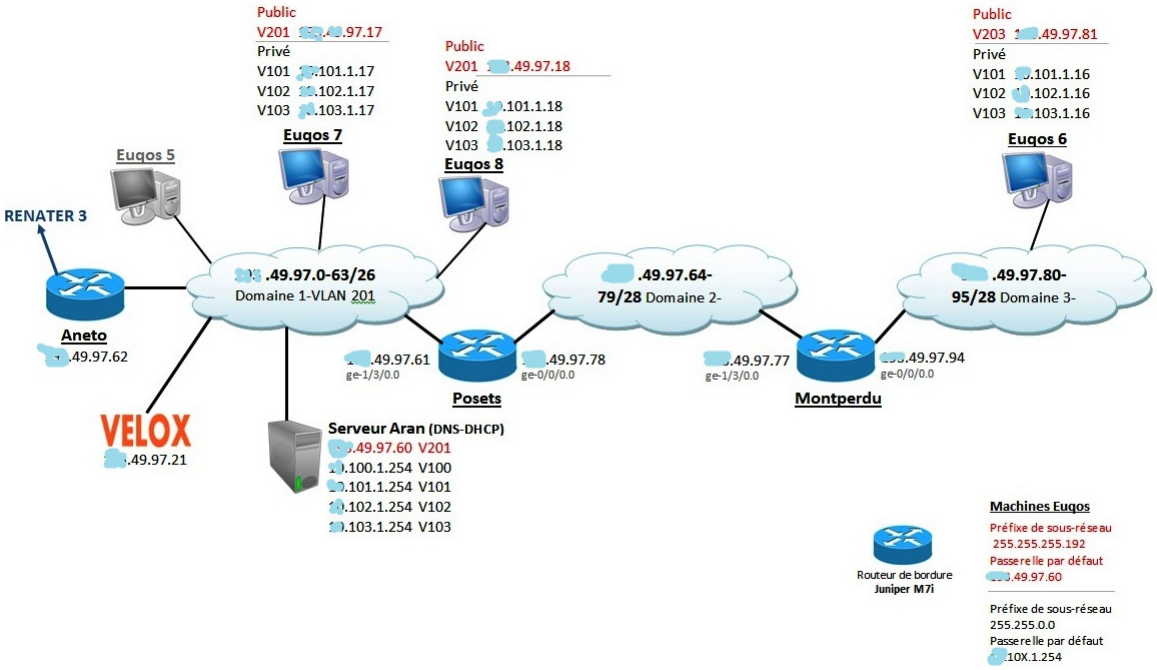


FIGURE 7: Laasnetexp testbed

aux services de multicast. Les machines qui forment la plateforme sont des DELL PowerEdge 750, 850, 1850, 3850, 6650 et 6850.

B. Evaluation du pont fédéré

1) Principe de Test: Afin de vérifier si le volume du trafic transmis du producteur dépend du nombre de machines réceptrices et du nombre de consommateurs dans chacune, nous avons effectué le test suivant. Ce test consiste à publier des Topics (les carreaux sur la Figure 8) du côté de l'un des producteurs et déclarer plusieurs consommateurs de ces même Topics. Ces consommateurs sont situés dans des réseaux IP différents et des domaines DDS distincts.

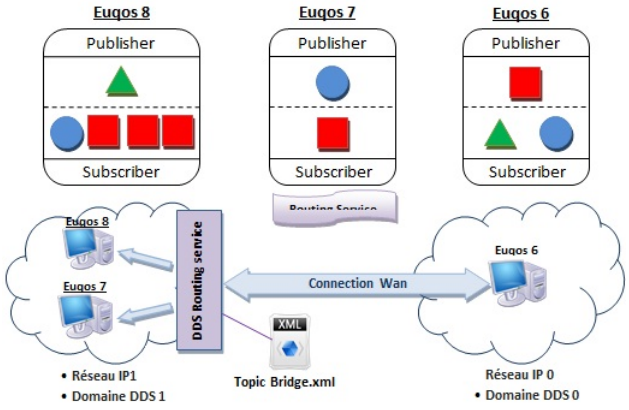


FIGURE 8: Scénario de communication du pont-fédéré le Topic Bridge

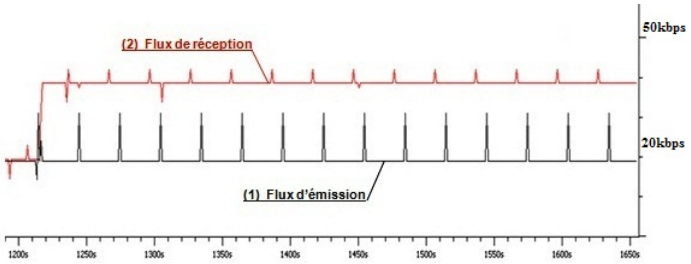


FIGURE 9: Les flux de données captés sur Euqos 6

2) Résultats et Analyses: Les résultats de ce test sont présentés dans la Figure 9 et la Figure 10. La Figure 9 représente les flux de données envoyés et reçus par la machine Euqos6 :

- Le trafic (1) représente le flux de données envoyé d'Euqos 6 vers le pont-fédéré avec un débit de 20Kbps et correspond à la distribution d'un seul Topic (Carreau).
- Le trafic (2) représente le flux de données reçus du pont-fédéré DDS avec un débit de 40Kbps et correspond à la consommation de deux Topics (Cercle et Triangle).

La Figure 10 représente les flux de données envoyés et reçus par le pont-fédéré exécuté sur Euqos7 :

- Le trafic (1) représente le flux de données reçu d'Euqos6 qui est similaire au trafic (1) de la Figure précédente.
- Le trafic (2) représente le flux de données envoyé vers Euqos6 qui est similaire aussi au trafic (2) de la Figure précédente.
- Le trafic (3) représente le flux de données envoyé d'Euqos7 vers Euqos8 avec un débit de 80Kbps et correspond à la distribution de deux Topics et 4 instances (3 carreaux

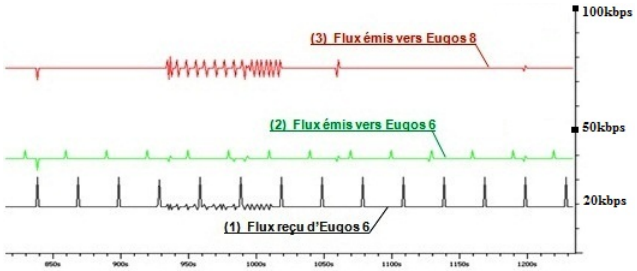


FIGURE 10: Flux de données capté par le pont-fédéré avec le Topic Bridge

et un cercle).

D'après ces analyses, nous pouvons retenir que le volume de données envoyé du domaine DDS 0 (depuis Euqos6) vers le domaine DDS 1 (Euqos7 qui contient en plus du consommateur, le pont-fédéré) ne dépend ni du nombre de participants DDS locaux ni du nombre de consommateurs dans le site distant, mais juste des types de Topics envoyés. Nous remarquons aussi que le transfert des Topics d'un site à un autre commence dès que le pont-fédéré s'inscrit à ces Topic et qu'il les duplique par la suite dans son réseau à destination des participants consommateurs.

Nous pouvons donc constater qu'il est possible d'optimiser le volume de données échangées sur les liens physiques entre des applications DDS situées dans deux réseaux IP distants et deux domaines DDS différents. Cependant, le problème est que cette solution n'est pas capable d'offrir les mêmes performances dans le cas d'un seul domaine DDS. En effet, nous avons remarqué que le volume de données envoyé d'Euqos6 vers Euqos7 est deux fois plus grand quand ils sont dans un seul domaine DDS.

C. Evaluation du proxy DDS

1) Principe de Test: Pour évaluer le proxy DDS, d'abord nous avons réalisé une application DDS distribuée qui assure un échange de données dans un seul domaine DDS. Deux Topics ont été utilisés : "UserInfo" et "CarInfo". Côté producteur de l'application on considère un Publisher du Topic ≪UserInfo≫ et un Subscriber du Topic ≪CarInfo≫, et côté client de l'application on considère un Subscriber de ≪UserInfo≫ et un Publisher de ≪CarInfo≫.

Nous avons déployé le Proxy DDS pour mettre en réseau l'application de test. Ce Proxy est responsable de l'échange des deux Topics entre Euqos6 et les subscribers de l'application situées dans deux réseaux distants.

Le scénario utilisé pour ce test est présenté dans la Figure 11 : Considérons l'architecture de la plateforme LaasNetExp pour ce test. Le serveur de l'application est lancé sur la machine (Euqos 6) du domaine 3 et les clients sur les machines (Euqos 7 et 8) du domaine 1. Par la suite, le Proxy DDS est installé sur les deux réseaux (précisément sur Euqos 6 et 7).

Le scénario de test consiste à lancer sur :

– Euqos6 : Un Publisher de "UserInfo", un Subscriber de "CarInfo" et le Proxy DDS1,

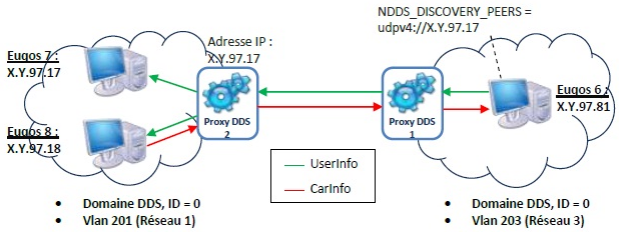


FIGURE 11: Scénario de test du Proxy DDS

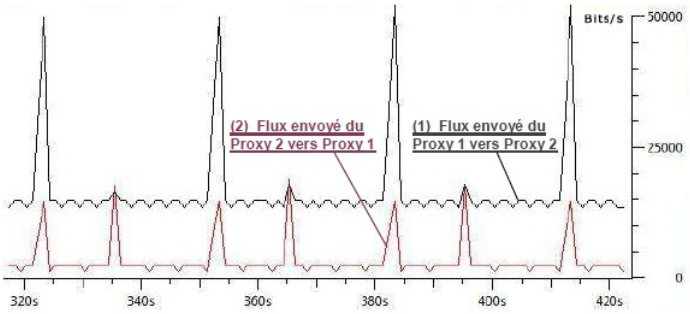


FIGURE 12: Courbes du trafic échangé entre les deux proxies DDS

– Euqos7 : Deux Subscribers de "UserInfo" et le Proxy DDS 2,
– Euqos8 : Deux Subscribers de "UserInfo" et un Publisher de "CarInfo"

2) Résulats et Analyses: Pour analyser les résultats obtenus à partir des tests, nous nous focalisons sur la Figure 12 qui illustre les différents flux échangés : entre Euqos6 et Euqos7 qui correspond au flux transmis entre les deux Proxy DDS 1 et 2, le flux retransmis du Proxy DDS 1 vers Euqos8 et finalement le flux transmis d'Euqos8 vers le Proxy DDS 1. Ces flux de trafic sont présentés sur les figures 12 et 13.

La Figure 12 illustre le trafic transmis entre les deux Proxy DDS distants. La courbe (1) : représente la distribution d'une seule instance du Topic "UserInfo" envoyé du proxy 1 vers le proxy 2 avec un débit moyen de 15Kbps. La courbe (2) : représente la distribution d'une seule instance du Topic "CarInfo" envoyé du proxy 2 vers le proxy 1 avec un débit moyen de 2,5Kbps.

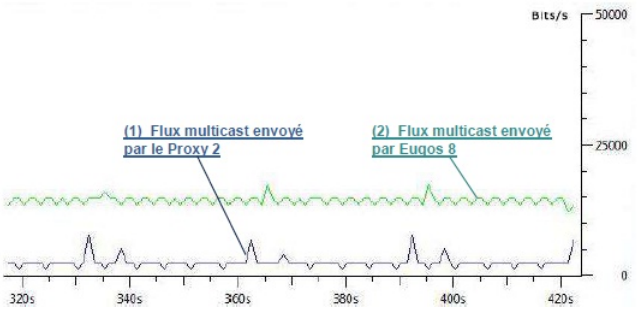


FIGURE 13: Courbes du trafic échangé entre un proxy DDS et un subscriber sur EuQoS8

La Figure 13 illustre le trafic transmis en multicast dans le domaine DDS1 entre le proxy DDS 2 et les participants DDS dans son domaine (EuQoS 7 et 8). La courbe (1) représente la redistribution du Topic "UserInfo", envoyé par le proxy DDS 2 sur le réseau avec un débit moyen de 15Kbps. La courbe (2) représente la distribution du Topic "CarInfo" en multicast, envoyé par Euqos 8 sur le réseau avec un débit moyen de 2,5Kbps.
Les résultats obtenus ont confirmé que le volume de données envoyé par un proxy DDS (flux des Topics DDS de l'application productrice), à travers le réseau d'interconnexion ne dépend pas du nombre de participants DDS (subscriber), mais dépend uniquement du profil de production des Topics envoyés (Type et taille).
En conséquence, le proxy DDS représente une solution efficace pour l'optimisation de la bande passante de flux d'applications DDS. Il permet aussi de réduire le coût du transfert de données partagées dans entre plusieurs domaine DDS distincts interconnectés à travers un réseau multi-domaine. Comme prédit, le trafic sur le réseau est réduit à sa bande passante minimale.

V. Conclusion

Dans ce travail, nous avons montré qu'il est possible d'interconnecter des applications DDS situées sur des domaines DDS distincts en assurant leur interfonctionnement en réseaux grande distance. Pour cela, nous avons d'abord analysé la possibilité d'utiliser le service de routage que fournit la couche middleware DDS par l'implémentation d'un pont-fédéré qui permet de lier des domaines DDS distincts à travers un réseau IP multi-domaines. Ensuite, nous avons proposé un proxy DDS qui en le comparant avec le pont-fédéré, permet d'optimiser la bande passante en réduisant le trafic DDS échangé en son expression minimale. Toutefois, bien que ces solutions améliorent notre réponse à la problématique d'interfonctionnement d'applications DDS, elles n'offrent qu'une réponse imparfaite au problème de gestion de la QoS. Nos travaux futurs se focaliseront sur l'extension du proxy DDS vers une architecture de communication à QoS garantie en réseaux IP multi-domaines.

Références

[1] O. Object_Management_Group, "Data Distribution Service for Real-Time Systems Specification, DDSv1.2," in *http://www.omg.org/spec/DDS/1.2/*, 2009.

[2] A. Corradi and L. Foschini, "A DDS-compliant P2P infrastructure for reliable and QoS-enabled data dissemination," in *Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing*, IPDPS '09, pp. 1–8, 2009.

[3] V. Lopez, M. Jose, P. Javier, and J. Lopez-Soler, "DDS/SIP Interworking : A DDS-SIP Gateway," in *Real-Time and Embedded Systems Workshop*, May 2010.

[4] G. V. Marisol, B. V. Pablo, and E. A. Iria, "Adaptive Real-Time Video Transmission over DDS," in *8th IEEE International Conference on Industrial Informatics (INDIN)*, pp. 130–135, july 2010.

[5] J. Darby, M. L.S., N. Curran, and P. B. Armen, "Applying Publish-Subscribe to Communications on the Move Node Control," in *Lincoln Laboratory Journal*, vol. 16, 2007.

[6] J. Schlesselman, G. Pardo-Castellote, and B. Farabaugh, "OMG Data Distribution Service (DDS) : Architectural Update," in *Military Communications Conference, 2004. MILCOM 2004. 2004 IEEE*, vol. 2, pp. 961–967, oct.-3 nov. 2004.

[7] R. T. I. RTI Innovation Inc, "RTI Data Distribution Service Routing Service, version 4.5c," in *http://community.rti.com/content/page/documentation*, 2011.

[8] R. Innovation, "NDDS Middleware," 2004. RTI Innovation publish.

[9] O. Philippe, Y. L. Pascal, B., and G. David, "LaasNetExp : une plateforme expérimentale pour l'émulation et les tests en réseaux," in *Colloque Francophone d'Ingénierie des Protocoles, CFIP*, 2008.