DDS et SDN en symbioses pour les applications dynamiques

Armel Francklin Simo Tegueu, Pascal Berthou, Slim Abdellatif, Thierry Villemur, Faten Mkacher

2015 International Conference on Protocol Engineering (ICPE) and International Conference on New Technologies of Distributed Systems (NTDS).

DDS[a] and SDN[b] in symbiosis for dynamic applications

Armel Francklin Simo Tegueu[1,2,*], Pascal Berthou[1,3], Slim Abdellatif[1,2],Thierry Villemur[1,4] and Faten Mkacher[1,2]

[1]CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France

[2]University of Toulouse, INSA, LAAS, F-31400 Toulouse, France

[3]University of Toulouse, UPS, LAAS, F-31400 Toulouse, France

[4]University of Toulouse, UT2J, LAAS, F-31100 Toulouse, France

***Abstract*—The emergence of dynamic applications (whose data flows and their associated Quality of Service (QoS) requirements vary over time) poses new challenges to the underlying communication system and in particular to the communication network. The DDS (Data Distribution Service) middleware allows applications to dynamically and finely express their QoS needs. However, they do not map these changing requirements at the network level and usually resort to network resource over-provisioning. To address this problem, we propose a solution based jointly 1) on the Software Defined Network (SDN) paradigm, which provides network flexibility; and 2) on the MAPE (Monitoring, Analysis, Planning and Execution) framework, which introduces autonomic properties into the allocation of network resources in an SDN / OpenFlow context.**

I. INTRODUCTION

Les nouvelles applications ont souvent transformé l'Internet et sa structuration, imposant une architecture de communication compatible avec les besoins émergents. Le Web a imposé une architecture client/serveur, souvent asymétrique que les réseaux d'« égal à égal » ont ensuite tenté de rééquilibrer. Les besoins d'échanges de gros volumes d'information issus des réseaux sociaux ont accéléré l'émergence des IXP (Internet eXchange Points). Les applications dynamiques, comme la simulation distribuée, qui répondent aujourd'hui à un besoin de l'industrie, risquent à leur tour d'imposer un nouveau virage à l'Internet. Cet article propose une architecture de communication répondant à cette problématique, en se basant sur les paradigmes de communication producteur/consommateur et des réseaux programmables.

La première partie présente plus en avant la problématique relative à l'apparition d'applications distribuées dynamiques qui est rendue possible par les capacités croissantes des réseaux de communication. La deuxième partie abordera un état de l'art qui traitera des architectures de communication permettant d'offrir des services adaptatifs nécessaire à leur support. Mais aussi une rapide présentation des technologies sous-jacentes à notre proposition sera faite et concernera le middleware DDS ainsi que les réseaux SDN. Pour expliciter plus en avant le contexte, nous donnerons un exemple d'application dynamique que nous utiliserons comme support pour la présentation de l'architecture ensuite détaillée. Finalement, la dernière section, se focalisera sur deux des modules de l'architecture et présentera succinctement les gains obtenus.

II. PROBLEMATIQUE

De plus en plus d'applications émergentes à l'instar des applications interactives de simulation distribuées [9] sont des applications dynamiques dans la mesure où les flux de données qu'elles échangent et les besoins stricts de qualité de service (QoS, pour « Quality of Service ») en termes de délai et de bande passante évoluent dans le temps. Cette classe de service se démarque de celles définies par l'ITU-T Y.1541 [1] et I.356 [2] et de l'IETF RFC 4594 [3] de par son caractère dynamique. Les middlewares ont suivi cette nouvelle tendance. Parmi eux, le Service de Distribution de Données (DDS) s'est particulièrement illustré[4], non seulement par son support très riche et varié de paramètres QoS, mais également en fournissant une API qui permet aux applications d'exprimer dynamiquement leurs exigences en terme de communications. Les services ou les composants applicatifs qui partagent le même espace d'échange de données, décrivent une sorte d'overlay applicatif dont la topologie et la capacité des liens sont variables. La prise en compte de ces exigences au niveau réseau, suppose que ce dernier soit 1) suffisamment flexible pour être reprogrammé afin d'instancier à la volée ces overlays puis suivre leurs évolutions, 2) autonome pour suivre les changements de contexte liés aux modifications réseau et agir en conséquence afin de maintenir le service. C'est le talon d'Achille des réseaux actuels car, en pratique, ce sont plutôt des réseaux « overlays » statiques et surdimensionnés qui sont déployés (ce qui s'avère coûteux et dans certains cas difficile à mettre en place) ou sinon, à défaut, il est demandé à l'application de s'adapter aux performances rendues par le réseau.

Dans ce travail, en reposant conjointement sur le paradigme Réseau Guidé par du Logiciel (Software Defined Network (SDN)), dont l'une des finalités est d'offrir de la flexibilité

[a] Data Distribution Service

[b] Software Defined Network

978-1-4673-9265-5/15/$31.00 ©2015 IEEE

aux réseaux, et sur un module d'extension du contrôleur SDN, basé sur le Framework MAPE (Monitor-Analyze-Plan-Execute) intégrant les aspects autonomiques, nous adressons le problème de fournir des services réseau à QoS dynamique. Ce travail vise en particulier le cas des applications dynamiques de type DDS (comme par exemple les environnements de simulation distribuée multi-utilisateurs). En effet, grâce à son modèle de coopération en « Publier/Souscrire» et la multitude de paramètres de QoS associés, DDS permet aux applications de décrire leurs besoins en trafic et en QoS avec un niveau de détail beaucoup plus fin que ce que l'on peut obtenir avec d'autres approches (telles que Group based policy). Sur la base de cette description détaillée des attentes dynamiques de l'application, nous cherchons à définir des services réseau qui soient capables de coller à ses besoins tout en utilisant au mieux les ressources réseau.

III. ETAT DE L'ART

A. Travaux antérieurs

Internet est basé sur un service réseau très simple dit best-effort, ne fournissant aucune différenciation entre plusieurs flux réseaux et ne permettant aucune garantie sur le délai et la bande passante. Les applications utilisant ce service sont des applications élastiques, c'est à dire qu'elles peuvent s'adapter aux conditions variables de bande passante disponible et aux variations du délai des paquets. L'émergence des applications multimédia a suscité les travaux visant à proposer différents modèles d'architecture pour la gestion de la QoS. Les techniques traditionnelles de contrôle de la QoS comme la réservation de bande passante avec le protocole Resource Reservation Protocol(RSVP)[6] sont principalement statiques et souffrent du passage à l'échelle.

Pour pallier ce manque de dynamique, dans [7], [8], [9], [10], les auteurs tirent profit des réseaux SDN/OpenFlow qui offrent un contrôle centralisé et une meilleure visibilité des ressources réseaux disponibles. Ces nouvelles architectures répondent mieux aux exigences de gestion de QoS en termes de granularité, reconfigurabilité, autonomicité, et sont mieux adaptées aux besoins stricts et dynamiques de QoS. Bueno et al.[7] proposent une architecture pour la réservation dynamique de ressources basée sur le Framework OpenNaaS. L'interface offerte à l'application pour effectuer les demandes est spécifique et implantée en langage REST. Bari et al.[8] proposent une architecture similaire pour la mise en place et le maintien d'accord du niveau de service (Service-Level Agreement (SLA)) dans un réseau programmable par OpenFlow. Les SLAs sont décrits et proposés par l'administrateur du réseau. Tomovic et al. [10] intègrent en supplément un module de planification et de calcul de route basé sur un algorithme de Dijskstra. Dans ces trois architectures une part importante est faite au monitoring pour permettre une adaptation dynamique. Les auteurs de [9] se concentrent sur l'interface offerte aux applications pour exprimer leurs besoins. Ils mettent en exergue les biais des approches concurrentes qui offrent des APIs possédant trop peu d'abstraction pour être adaptées au besoin d'un développeur d'application. L'article décrit principalement les concepts d'une API qui devrait leur être offerte. Les architectures proposées dans [7], [9], spécifient une nouvelle interface que devront utiliser les développeurs d'application pour spécifier les besoins en termes de communication. Cette approche demande donc une adaptation pour les applications existantes et un développement spécifique pour les nouvelles, minimisant ainsi leur adoption.

Notre solution DDS/SDN propose une intégration non intrusive ne nécessitant pas d'extension du middleware DDS ni de l'application, ce qui permet ainsi un rapide déploiement de l'existant. De plus, elle intègre les concepts de monitoring et de planification cités précédemment.

B. Technologies

1) Data Distribution Service (DDS)

Data Distribution Service (DDS)est un standard de type middleware producteur consommateur, spécifié par l'Object Management Group (OMG), qui cible les applications temps-réel distribuées avec des exigences telle que la distribution de gros volumes de données avec des latences faibles. Le middleware DDS fournit les mécanismes nécessaires pour la prédictibilité et le contrôle des ressources tout en offrant la modularité, le passage à l'échelle, l'évolutivité, la fiabilité et la robustesse de la communication. La Figure 1 illustre une vue d'ensemble de DDS.

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 créent et envoient certaines de ces données et des processus consommateurs (subscriber) qui les reçoivent. Un participant peut publier et s'abonner simultanément à des informations identifiées par le nom de Topic. A partir de ces déclarations d'intention (à produire ou à consommer un topic), DDS met automatiquement en relation les producteurs et les consommateurs qui se partagent le même topic et se charge alors de livrer les différentes valeurs produites (également appelées échantillons) à destination des consommateurs qui se sont déclarés intéressés par le topic.

L'architecture de DDS repose sur deux niveaux d'interface de service. Au plus bas niveau, la couche Data Centric Publish Subscribe (DCPS) contient cinq modules : Infrastructure, Topic, Publication, Abonnement et Domaine. Elle offre aux applications des APIs très importantes qui leur permettent d'accéder au service de distribution de données en respectant la QoS prédéfinie avec un contrôle intégré sur les ressources. Le module Infrastructure définit les classes abstraites qui servent de point d'entrée aux autres modules. Il contient plusieurs classes dont celles qui nous intéresseront : la classe DCPSEntity modélisant les entités DCPS (Topic, Publisher, Subscriber, DataReader, DataWriter, …), la classe QoSPocily modélisant la famille importante de paramètres QoS (Tableau 1), la classe Listener (DataReaderListener, …) modélisant le point d'accès aux données et aux notifications en mode asynchrone.

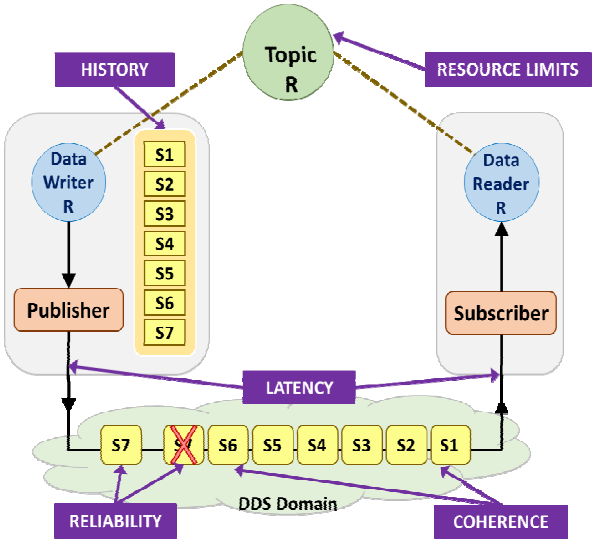

Figure 1: **Vue d'ensemble de DDS**

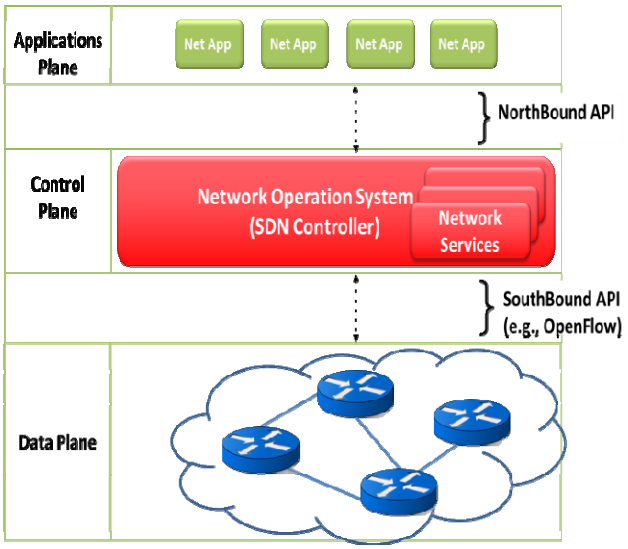

Figure 2: Architecture SDN

DDS offre donc aux applications un support varié de paramètres de QoS répartis sur plusieurs niveaux middleware, système d'exploitation et réseau (délai, débit, …) et pouvant évoluer au cours du temps selon la logique applicative.

Toutefois, la sémantique d'expression de QoS de DDS diffère de celle communément utilisée pour décrire des exigences au niveau réseau. En effet, DDS propose cinq classes de paramètres de QoS (Tableau 1), dont seules celles liées aux aspects temporels (Data Timeliness) et de ressources (Resources) sont aisément transposables en paramètres de priorités ou de débits classiquement utilisés au niveau de la couche réseau.

Classe DDS	*Paramètres de QoS*
Data avalaibility	Durability ; Durability Service ; Lifespan ; History
Data Delivery	Presentation, Reliability, Partition, Destination Order, Ownership, Ownership Strength,
Data Timeliness	Deadline, Latency Budget, Transport Priority,
Resources	Time Based Filter, Resource Limits,
Configuration	User_Data, Topic_Data, Group_Data

Tableau 1: **Classes des paramètres de QoS définis dans DDS**

2) SoftwareDefined Network (SDN) et OpenFlow

Software Defined Network(SDN) fait référence à un nouveau paradigme réseau qui prône le renvoi des fonctions de contrôle réseau, historiquement localisées au sein des équipements réseau, vers des entités centrales (contrôleurs SDN), et le pilotage à distance des fonctions d'acheminement de ces équipements. La Figure 2 résume les principes des réseaux SDN. Dans cette architecture, les équipements de transmission ne mettent en œuvre que la fonction d'acheminement.

Les règles d'acheminement associées à chaque flux de données décrites sont communiquées aux équipements de transmission par une entité distante appelée contrôleur SDN. Le contrôleur utilise une interface dite Sud (SouthBound Interface), pour décrire les règles définies et inter-opérer avec ces équipements de transmission. Ces règles sont traduites selon la spécification de cette interface. Actuellement, l'interface Sud la plus aboutie est OpenFlow, maintenue et standardisée par le consortium industriel Open Networking Fondation (ONF). Les fonctionnalités d'OpenFlow permettent d'implanter une architecture complexe de QoS, un monitoring plus fin ainsi qu'une indépendance vis-à-vis des fournisseurs de commutateurs réseau.

Le service offert par le contrôleur SDN peut être comparé à un système d'exploitation réseau (Network Operating System), dans la mesure où il masque l'état distribué du réseau en fournissant aux applications de contrôle une vue globale du réseau. Les programmes de contrôle utilisent cette interface pour interroger et programmer le réseau et éventuellement pour être prévenus lorsqu'un évènement se produit.

L'approche SDN offre la possibilité d'inculquer la flexibilité aux réseaux (propriété qui manque aux réseaux contemporains) et de leur permettre de répondre à la volée à l'évolution des besoins applicatifs et de leur environnement opérationnel.

Avant de présenter l'architecture complète, utilisant le pouvoir d'expression de DDS et la flexibilité des SDN, le paragraphe suivant détaille une étude de cas permettant d'exprimer quels sont les besoins et les contraintes des applications émergentes que nous visons.

IV. ETUDE DE CAS

Cette section présente une application de simulation distribuée impliquant plusieurs simulateurs de véhicules et dont le but est de permettre la formation à la conduite en groupe[12] tel que l'escorte d'un convoi de fonds ou l'escorte présidentielle. Ces simulateurs sont interconnectés par des réseaux longue distance via des liaisons VPN et des réseaux locaux. La

Figure **3** présente une illustration simplifiée du cas étudié.

Ces simulateurs (participant DDS) évoluent dans un monde virtuel muni d'un système de coordonnées géographique, modélisant un environnement de simulation (ville, campagne, route) et s'échangent les informations d'état (Topic DDS {position, direction et vitesse}). Toutes ces informations sont échangées dans le Domaine DDS.

La spécificité de cette application réside dans la quantité d'information transmise, puisqu'elle vise la simulation haute-fidélité, afin de reproduire au mieux un exercice réel. Ainsi, les simulateurs produisent périodiquement une quantité importante de données, représentant par exemple la position de chacune des roues, au lieu d'un système classique qui distribue uniquement la position du véhicule et estime la position de ses roues. Cette dernière solution n'est pas suffisante pour représenter par exemple un véhicule en état de perte d'adhérence et de glisse.

Quand le nombre de véhicules est important, il n'est plus possible de tout diffuser sur le réseau, d'autant plus si les données traversent des réseaux peu dimensionnés. Idéalement, plus les véhicules sont proches entre eux dans le monde virtuel, plus le volume d'informations échangées entre eux doit être important, une distribution plus sporadique pour des véhicules très éloignés n'altérant pas le rendu visuel de la simulation. Par contre, si un véhicule accélère ou ralentit (changement de régime), il doit informer les véhicules qui le suivent en leur communiquant une grande quantité d'informations d'état fiables et cohérentes en peu de temps et avec un faible temps de latence afin que ceux-ci puissent réagir le plus rapidement possible. Le réseau doit donc être configuré pour supporter les exigences de cette phase de changement de régime tout au long de la simulation. Ces exigences strictes imposent une isolation de trafic et un ajustement dynamique de ressources associées à ce trafic. Une solution possible est de surdimensionner les liaisons, ce qui est couteux et très peu efficace par rapport à une autre qui consiste à ajuster plus finement les ressources réseau au besoin, sous réserve que le réseau soit dynamiquement programmable.

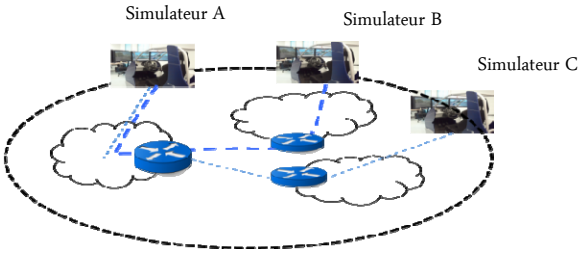

Figure 3: Simulateurs de véhicule distribués

V. ARCHITECTURE PROPOSEE

Notre solution repose et tire profit des possibilités offertes par les réseaux de type SDN en proposant une fonction réseau (Network Service Allocator) conçue selon le cycle autonomique, capable de déployer et de maintenir un réseau virtuel applicatif (VNet), et dont la topologie (ensemble des paires producteur-consommateur) et la capacité (définie par la QoS souscrite par le consommateur) des liens (VLink) peuvent être reprogrammées dynamiquement en fonction de l'évolution des évènements observés (déclaration du Topic, abonnement et désabonnement, modification de la QoS souscrite) dans le domaine DDS. Cette section présente les détails fonctionnels et techniques des composants de notre architecture illustrée à la Figure 4et leurs interactions. La Figure 5présente l'aspect dynamique de notre architecture.

A. DDS Handler

L'objectif de ce module est de suivre l'évolution de l'overlay décrit par les processus applicatifs et d'interpréter les évènements observés, en produisant les opérations: de création, de modification ou de suppression du VNet ou d'un VLink. Par exemple, le départ du producteur d'un topic peut se traduire par la libération des ressources mobilisées pour supporter la communication entre ce producteur et les consommateurs associés à ce topic. Pour cela, il exploite le service de monitoring fourni aux applications par DDS. Cette approche est transparente pour l'application et le middleware, renforçant l'adoption et l'intégration de notre architecture dans les systèmes existants. Par ailleurs elle nécessite que le bloc fonctionnel exécute le middleware DDS en tant que participant observateur du domaine DDS, afin de suivre et d'interpréter l'évolution des besoins de l'application.

B. Network Monitor

L'objectif de ce module est de suivre l'évolution de la topologie physique du réseau, des caractéristiques des liens (capacité, charge et délai) et des nœuds (capacité, charge). Il surveille en temps réel les performances, en termes de bande passante et de délai, des chemins réservés aux flux applicatifs ayant fait la demande, et déclenche éventuellement une réallocation de ressources lorsque certains changements surviennent. Le framework OpenFlow Notification [11] défini une liste d'évènements auxquels souscrit ce module.

La surveillance de la bande passante réservée aux flux se fait en utilisant la technique de monitoring en mode actif.

Cette technique consiste à interroger périodiquement les statistiques des entrées actives des tables associées à ces flux, grâce au concept de compteur. Des messages particuliers permettent cette interrogation via l'API Nord du contrôleur SDN. La charge s'évalue en sommant les bandes passantes consommées par les flux le traversant, bandes passantes obtenues en interrogeant les statistiques des ports physiques extrémités de ce lien.

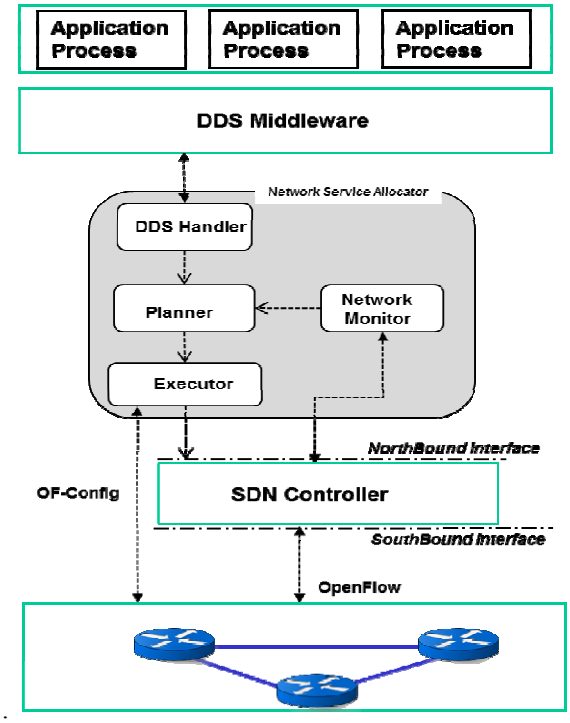

Figure 4: Architecture DDS-SDN

C. *Planner*

L'objectif principal de ce module est de calculer à la volée les chemins physiques (avec les ressources réseau à allouer le long de ces chemins) support de chaque VNet (qui se compose d'un ou plusieurs VLinks). Chaque lien est caractérisé par une bande passante et un délai maximal à garantir et peut être de type point-à-point ou point-multipoints. Notre algorithme considère deux types de ressources : classiquement, les ressources de transmission (capacité des liens de transmission) et les ressources de commutation au niveau des commutateurs SDN traversés. La prise en compte de ce dernier type de ressources est primordiale puisqu'avec les technologies actuelles, l'opération de commutation dans un réseau SDN est consommatrice en temps, ce qui impacte la taille maximale des tables d'acheminement des commutateurs SDN. La fonction objective sur laquelle repose l'algorithme du *planner* cherche à minimiser la quantité de ressources (transmission et commutation) allouée à chaque VNet, ainsi qu'à équilibrer la charge entre les différents nœuds et liens de l'infrastructure physique. De ce fait, elle favorise l'admissibilité des prochaines demandes d'allocation de ressources. L'algorithme que nous proposons repose sur une formulation selon un programme linéaire en nombres entiers[14].

D. *Executor*

L'objectif de ce module est d'alimenter le réseau avec des règles OpenFlow, de les modifier ou de les supprimer en réponse au besoin exprimé par le module précédent. Plus précisément, il traduit un besoin en une liste de blocs d'instructions rédigées conformément à l'API Nord du contrôleur. Ce contrôleur déploie les blocs à chaque commutateur concerné via le protocole OpenFlow. Il définit finement les flux grâce aux douze champs de la spécification OpenFlow Switch. Il instancie un contexte QoS pour chaque flux en créant dynamiquement une nouvelle classe de service via le protocole OF-Config[12]. Il crée et configure les files d'attentes associées à un port, permettant de contrôler, de lisser et de garantir un débit minimal et maximal et un délai maximal de transit.

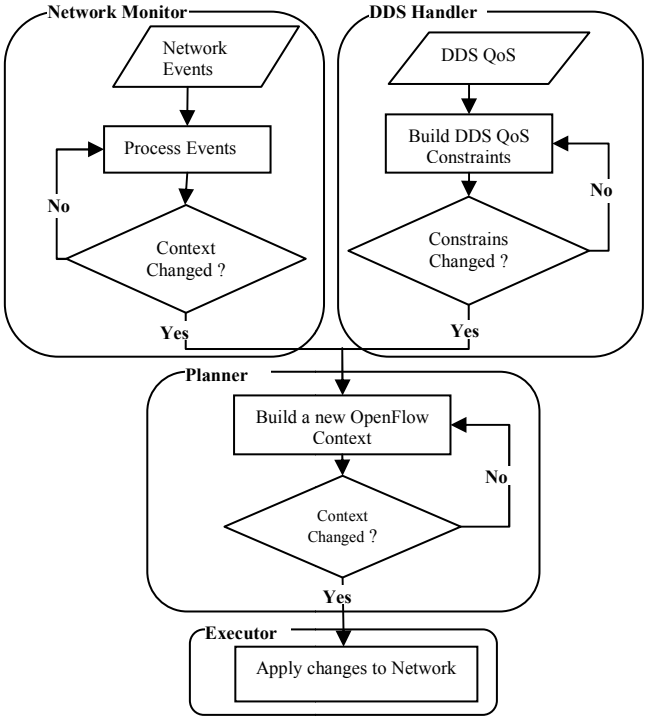

Figure 5: Dynamique de l'architecture avec son flux d'information

VI. RESULTATS

Cette section présente les résultats de mesure de performances de l'algorithme d'allocation de ressource du module *Planner* et de la mise en œuvre des fonctions de contrôle de trafic et de surveillance de débit d'un flux respectivement à la charge des modules *Executor* et *Network Monitor.*

A. *Performances de l'algorithme d'allocation de ressources du modulePlanner*

Comme décrit précédemment, l'objectif du module *planner*, consiste à partir d'une topologie réseau connue, d'un ensemble de contraintes, et de requêtes de constitution de réseaux virtuels, à trouver le placement optimal du VNet sur la topologie physique. Ce module a fait l'objet d'un précédent

travail dont les détails algorithmiques et expérimentaux ont été développés dans [14].Les résultats ont montré que notre algorithme a un taux d'admissibilité de 5 à 10% meilleur que celui produit par l'heuristique de plus court chemin la plus performante.

B. *Monitoring de la bande passante et du délai de transit d'un flux de bout en bout*

La Figure 6 présente à la fois le résultat du bloc *Network Monitor*, capable de mesurer le débit d'un flux applicatif à 10 Mbps (Débit démission) qui dépasserait son contrat de service (Débit crête 1 Mbps), ainsi que le résultat du bloc *Executor* qui met en place un régulateur de trafic limitant le débit à la valeur désirée. La dernière courbe, montre l'évolution de la gigue aussi mesurable par le bloc *Network Monitor*. Cette fonctionnalité de contrôle de trafic est permise par les fonctions de *metering* offertes par OpenFlow supportées depuis la version 1.3.0. Notre implémentation se base sur OFSoftSwitch13[15], seule plateforme logicielle à implémenter cette fonctionnalité aujourd'hui.

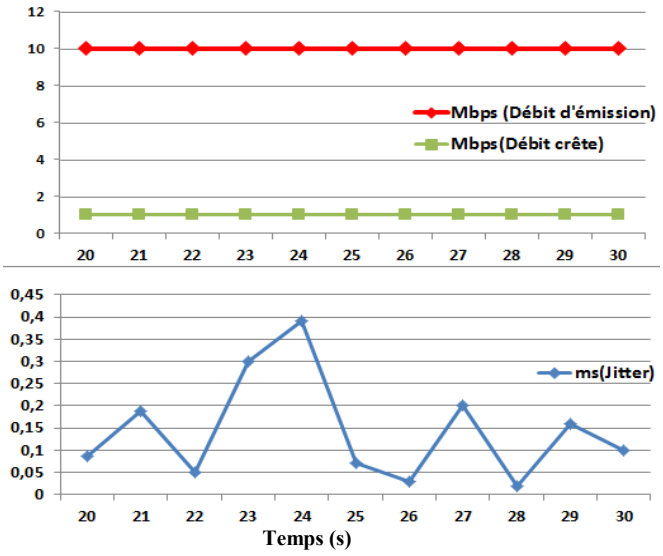

Figure 6: Contrôle et monitoring de trafic

VII. CONCLUSION

Cet article a présenté notre approche d'intégration DDS/SDN visant à répondre efficacement aux exigences de communication dynamiques de certaines applications émergentes.DDS donne aux applications la possibilité d'exprimer de manière fine et dynamique leurs besoins. Le logiciel de contrôle réseau que nous proposons dans ce travail permet de répercuter ces besoins au niveau réseau en provisionnant des services réseau dynamiques capables de répondre aux besoins applicatifs exprimés au niveau DDS.

Certaines parties de notre plate-forme d'intégration telles que le *Network Monitor* et le *Planner* doivent être améliorées et étendues. On envisage d'étendre le *Planner* par une fonction de migration de ressource et d'améliorer son algorithme d'allocation de ressource en modélisant plus finement la ressource de commutation par la considération des opérations de Matching liées au contexte OpenFlow.

Les performances présentées dans cet article sont une première étape. Il est nécessaire de pousser plus en amont cette étude au travers de l'établissement d'un "benchmarking" plus poussé, pour comparer ces résultats avec d'autres méthodes/architectures proposées dans la littérature.

En dernier lieu, afin d'étendre la généralisation de notre plate-forme, nous prévoyons de nous intéresser à d'autres cas d'études comme des applications massivement multiutilisateurs (les jeux vidéo par exemple).

Remerciements

Ce travail a été partiellement financé par l'Agence Nationale de la Recherche (ANR) et la Direction Générale de l'Armement (DGA) dans le cadre du projet ANR DGA ADN (ANR-13-ASTR-0024)

REFERENCES

[1] Y.1541, *Network Performances Objectives for IP-based Services*, ITU-T, January, 2005
[2] I.356, B-ISDN ATM layer cell transfer performance, ITU-T, May, 1996
[3] J. Babiarz, K. Chan and F. Baker, *Configuration Guidelines for DiffServ Service Classes*, RFC 4594, June, 2006
[4] Jose-Luis Poza-Luján, Juan-Luis Posadas-Yagüe, and José-Enrique Simó-Ten, *A Survey on Quality of Service Support on Middleware-Based Distributed Messaging Systems Used in Multi Agent Systems*, International Symposium on DCAI, AISC 91, pp. 77–84.
[5] S. Shenker, C. Partridge and R. Guerin, *Specification of Guaranteed Quality of Service, RFC 2212, September*, 1997
[6] L. Zhang, S. Berson., S. Herzog, and S. Jamin, (1997). *Resource reservationprotocol (rsvp) – version 1 functional specification*. RFC 2205
[7] I. Bueno, J. Aznar, E. Escalona, J. Ferrer, and J. Antoni Garcia-Espin, *An opennaas based sdn framework for dynamic qos control*, in Future Networks and Services, 2013 IEEE SDN for Future Networks and Services, Nov 2013, pp. 1–7
[8] Md. BariF.,S.R Chowdhury, R. Ahmed, R. Boutaba, *PolicyCop: An Autonomic QoS Policy Enforcement Framework for Software Defined Networks*, Future Networks and Services (SDN4FNS), 2013 IEEE SDN for Future Networks and Services, vol., no, pp.1,7, 11-13 Nov. 2013.
[9] GorlatchS.,HumernbrumT., GlinkaF., *Improving QoS in real-time internet applications: from best-effort to Software-Defined Networks*, Computing, Networking and Communications (ICNC), 2014 International Conference on , vol., no., pp.189,193, 3-6 Feb. 2014
[10] S. Tomovic,N. Prasad,I. Radusinovic, *SDN control framework for QoS provisioning,*Telecommunications Forum Telfor (TELFOR), 2014 22nd , vol., no., pp.111,114, 25-27 Nov. 2014
[11] Open Networking Foundation, *OpenFlow Notification Framework, ONF Technical Specification*, TS-014, V.1.0, Oct. 2013
[12] Open Networking Foundation, OpenFlow Management and Configuration Protocol, *ONF Technical Specification*, TS-016, V.1.2, Jan. 2014
[13] Akram Hakiri, Pascal Berthou, Thierry Gayraud, *Design of Low Cost PC-based Simulators for Education and Training Purpose Using DDS,* International Conference on Computer as a Tool, EUROCON 2011, Apr 2011, Lisbonne, Portugal. pp.103, 2011
[14] M. Capelle, S. Abdellatif, M.J. Huguet, P. Berthou, *Online Virtual Links Resource Allocation in Software-Defined Networks*, IFIP Networking 2015, Toulouse, France, 20 – 22 May 2015.
[15] CPqD, *OFSoftSwitch 13,* [Online]. Available: http://cpqd.github.io/ofsoftswitch13/