Supporting SIP-based End-to-End Data Distribution Service QoS in WANs

Akram Hakiri[a,b], Pascal Berthou[a,b], Aniruddha Gokhale[c], Douglas C. Schmidt[c], Gayraud Thierry[a,b]

[a] *CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France*
[b] *Univ de Toulouse, UPS, LAAS, F-31400 Toulouse, France*
[c] *Institute for Software Integrated Systems, Dept of EECS*
Vanderbilt University, Nashville, TN 37212, USA

Abstract

Assuring end-to-end QoS in enterprise distributed real-time and embedded (DRE) systems is hard due to the heterogeneity and transient behavior of communication networks, the lack of integrated mechanisms that schedule communication and computing resources holistically, and the scalability limits of IP multicast in wide-area networks (WANs). This paper makes three contributions to research on overcoming these problems in the context of enterprise DRE systems that use the OMG Data Distribution Service (DDS) quality-of-service (QoS)-enabled publish/subscribe (pub/sub) middleware over WANs. First, it codifies the limitations of conventional DDS implementations deployed over WANs. Second, it describes a middleware component called Proxy DDS that bridges multiple, isolated DDS domains deployed over WANs. Third, it describes the NetQSIP framework that combines multi-layer, standards-based technologies including the OMG-DDS, Session Initiation Protocol (SIP), and IP DiffServ to support end-to-end QoS in a WAN and shield pub/sub applications from tedious and error-prone details of network QoS mechanisms. The results of experiments using Proxy DDS and NetQSIP show how combining DDS with SIP in DiffServ networks significantly improves dynamic resource reservation in WANs and provides effective end-to-end QoS management.

Keywords: **Bridge-Federate model; Proxy DDS; SIP; NetQSIP; QoS Framework; DiffServ; Common Open Policy Service (COPS).**

1. Introduction

Current trends and challenges. Regional smart power grids or air traffic management systems are examples of enterprise distributed real-time and embedded (DRE) systems that disseminate high volumes of data with low end-to-end latency and jitter. These types of systems often comprise multiple end-to-end application flows that may require various quality-of-service (QoS) properties affecting CPU, memory, and network resources. They also operate in distributed and heterogeneous environments that process data from large number of terminals, media sources, and real-time data feeds [15] that originate over diverse geographic locations connected via wide-area networks (WANs).

In enterprise DRE systems the right answer delivered too late becomes the wrong answer [61]. Key challenges faced when fielding these systems thus include distributing a high volume of messages per second while simultaneously meeting requirements

Email addresses: hakiri@laas.fr (Akram Hakiri), berthou@laas.fr (Pascal Berthou), a.gokhale@vanderbilt.edu (Aniruddha Gokhale), d.schmidt@vanderbilt.edu (Douglas C. Schmidt), gayraud@laas.fr (Gayraud Thierry)

for scalability and low/predictable latency, controlling trade-offs between latency and throughput, and maintaining stability during bandwidth fluctuations. Moreover, end-system mechanisms must work within and across different communication access points, network domains, and inter-domain links to ensure end-to-end QoS requirements are met.

Over the past decade, standards-based middleware has emerged that addresses many enterprise DRE system challenges, including distributing high volume data scalably/predictably and minimizing end-to-end latency/jitter. In particular, the OMG *Data Distribution Service* (DDS) [44] defines a standard architecture for real-time, data-centric publish/subscribe (pub/sub) middleware capabilities that are used in many DRE systems to provide efficient and predictable dissemination of time-critical data [36]. Due to its flexible programming abstractions and rich support for QoS policies/mechanisms capable of controlling the end-to-end properties of data distribution [27], DDS is a popular technology for building enterprise DRE systems [2].

For example, DDS defines a set of *network scheduling policies* (*e.g.*, end-to-end network latency budgets), *timeliness policies* (*e.g.*, time-based filters to control data delivery rate), *temporal policies* to determine the rate at which periodic data is refreshed (*e.g.*, deadline between data samples), *network priority policies* (*e.g.*, transport priority can be used to set the DSCP field for DiffServ in the underlying data transport), and other policies that affect how data is treated in transit with respect to its reliability, urgency, importance, and durability.

Although standards-based DDS implementations have been used to develop many efficient and dependable DRE systems in *local-area networks* (LANs), conventional implementations of the DDS standard incur the following limitations that impede their use for *enterprise* DRE systems that run in *wide-area networks* (WANs) that comprise multiple network domains.

Limitation 1: Lack of communication between isolated DDS domains. Conventional DDS implementations run within an isolated DDS *domain*[1] traditionally deployed within a single multicast-enabled LAN. For example, individual power stations and power distribution plants of a smart grid could operate within their own DDS domains that are isolated from each other. As DRE systems expand into enterprise DRE systems (such as a regional smart power grid or air traffic management system) there is a need to interconnect these independent and isolated DDS domains.

The DDS standard, however, makes no mention about interconnectivity among isolated DDS domains. In fact, DDS domains were devised to isolate the individual communications within domains from each other, which was necessary if multiple independent domains operate within overlapping network boundaries. In enterprise DRE systems, these DDS domains are often distributed across WANs. Communication between isolated DDS domains—even if were addressed by the DDS standard—is hard due to the challenge of delivering the topics along with their expected end-to-end QoS requirements to participants across a WAN and between isolated DDS domains.

Limitation 2: Lack of IP multicast support in WANs. Conventional DDS implementations are oriented towards LANs and virtual LANs (VLANs).[2] For example, in a regional smart grid, DDS will try to discover various publisher (*e.g.*, sources of anomaly events at power generation facilities) and subscriber (*e.g.*, power distribution facilities) participants using conventional IP multicast [21], where the network equipment supports Ethernet multicast prefixes reserved for IP multicast groups [20].

IP multicast is often not deployed in multi-domain WANs, however, so conventional DDS implementations may be unable to distribute topics between participants running in isolated DDS domains over WANs. In particular, when DDS discovery messages are blocked by edge routers for security and/or per-

[1] A DDS is a data-space consisting of publishers and subscribers that communicate asynchronously and anonymously with each other via typed messages called topics.

[2] A VLAN is a single layer-2 network that may be partitioned to create multiple distinct domains that are mutually isolated.

formance reasons they do not reach subscribers in remote networks in a WAN. In the smart grid example, for instance, topics published by a publisher in one LAN may not reach subscribers in a geographically distributed LAN.

Limitation 3: Lack of DDS QoS provisioning mechanisms in WANs. The DDS standard only defines QoS mechanisms that control end-system properties, such as OS-level parameters and tuning network parameters for the connecting link. DDS implementations tend to support these features well only in the context of LANs/VLANs, and are limited in their support of key QoS policies over WANs. For example, events generated in one air traffic management domain may require real-time dissemination to another geographically distributed air traffic management domain. In conventional DDS implementations, however, it is hard to achieve these effects since they provide proprietary—and often incomplete and inefficient solutions—for provisioning/controlling end-to-end QoS over WANs, which impedes QoS assurance when participants span multiple network domains.

Overcoming the three limitations outlined above is essential to assure the end-to-end QoS of DDS-based enterprise DRE systems running over WANs.

Solution approach → ***Non-invasive DDS Extensions that Leverage IP Multimedia Core Network Subsystem (IMS) [64] technologies to ensure end-to-end QoS***. To address *Limitation 1* (lack of communication between isolated DDS domains) we developed *Proxy DDS*, which is a set of cooperating software components that enable efficient and seamless communication between DDS participants in isolated domains, and provides signaling. Proxy DDS components scalably decouple DDS participants in time/space and minimize traffic exchanged between these participants.

To address *Limitation 2* (lack of multicast support in WANs), the Proxy DDS provides proxy-to-proxy communication over WANs, thereby supporting scalable multicast. Likewise, to address *Limitation 3* (lack of QoS provisioning mechanisms in WANs), we integrated the Proxy DDS with *NetQSIP*, which is a *Session Initiation Protocol* (SIP)-based [32, 10] QoS framework that uses *Differentiated Service* (DiffServ) [7] policies to enforce network-level differentiated QoS for each DDS application data flow in a WAN.

Our prior work [30] presented a framework called *Velox* that conducted QoS negotiation and resource reservations across network elements in a WAN to meet scheduling requirements of DDS applications. Velox incurs several reliability and maintainability limitations, however, *e.g.*, it cannot recover from failures dynamically and must be manually reconfigured by human operators whenever a failure occurs. Moreover, Velox does not support dynamic QoS reconfigurations in DDS over WANs.

To overcome these limitations, this paper describes how combining Proxy DDS and NetQSIP provides an alternate strategy for supporting DDS QoS policies over WANs that does not introduce new capabilities at the network layer (as Velox did), but instead leverages standard SIP to achieve these goals. Although SIP has traditionally been used to support dynamic QoS-enabled IMS applications, this paper demonstrates how it can be used to support dynamic QoS management for DDS applications running in enterprise DRE systems. In particular, this paper makes the following contributions to research on assuring end-to-end QoS of enterprise DRE systems using DDS over a SIP-based WAN:

- It elaborates upon the limitations outlined above of deploying conventional DDS implementations over WANs.
- It describes how the Proxy DDS and NetQSIP framework resolve limitations with conventional DDS implementations over WANs by using SIP and DiffServ to optimize pub/sub communication and assure end-to-end DDS application QoS requirements. In addition to bridging isolated DDS domains and ensuring the propagation of the DDS QoS to different network domains, NetQSIP enables self-network reconfiguration after failures by adapting applications to network load.
- It empirically evaluates the capabilities of our enhanced DDS implementation in a WAN to determine how well (1) Proxy DDS optimizes the bandwidth between remote participants and (2)

the NetQSIP framework can achieve lower end-to-end delay.

Our solution preserves the DDS programming model so developers can seamlessly use the new capabilities of Proxy DDS and NetQSIP without changing their application designs. Moreover, since SIP is a standard and universally available, our solution can easily be adopted by different DDS implementations and support interoperability.

Paper organization. The remainder of this paper is organized as follows: Section 2 compares our research on the Proxy DDS and NetQSIP with related work; Section 3 describes how Proxy DDS and NetQSIP support effective and scalable inter-DDS-domain communication, end-to-end QoS signaling, and resource management in WANs; Section 4 analyzes the results of experiments that evaluate the capabilities of the Proxy DDS and NetQSIP in a WAN testbed; and Section 5 presents concluding remarks and lessons learned.

2. Background and Related Work

Conventional techniques for providing network QoS to applications incur several key limitations, including a lack of mechanisms to (1) specify deployment context-specific network QoS requirements and (2) integrate functionality from network QoS mechanisms at runtime. This section evaluates limitations with existing attempts to resolve these problems and outlines how our research on Proxy DDS and NetQSIP address them.

To clarify the specific challenges (*i.e.*, integrating DiffServ-based QoS provisioning mechanisms into DDS middleware) addressed in this paper, this section first provides an overview of OMG DDS and then describes the limitations of the *DDS Routing Service* (DDS-RS) [55], which is a content-aware middleware service designed to bridge DDS data spaces. We then compare and contrast Proxy DDS and NetQSIP with related work on general middleware-based QoS management and network-level QoS management.

2.1. Overview of the OMG Data Distribution Service (DDS)

The OMG DDS specification [44] defines a standard pub/sub architecture and runtime capabilities that enable applications to exchange data asynchronously, anonymously, and dependably in data-centric DRE systems. DDS provides efficient, scalable, predictable, and resource-aware data distribution via its *Data-Centric Publish/Subscribe* (DCPS) layer, which supports a global data store where publishers write and subscribers read data, respectively. DDS' modular structure and flexibility stems from its support for

- *Location-independence*, via anonymous pub/sub,
- *Redundancy*, by allowing any numbers of readers and writers,
- *Real-time QoS*, via its 22 QoS policies summarized in Table 1,
- *Platform-independence*, by supporting a platform-independent model for data definition that can be mapped to different platform-specific models (*e.g.*, C++ running on VxWorks or Java running on Real-time Linux), and
- *Interoperability*, by specifying a standardized protocol [43] that allows implementations from different DDS vendors to exchange data between distributed publishers and subscribers.

DDS specifies several types of DCPS entities. A *domain* represents the set of applications that communicate with each other. A domain acts like a *virtual private network* [3, 57], allowing DDS entities in different domains to communicate anonymously, regardless of whether they reside on the same machine, in the same process, or on another host in the network. A *domain participant factory* creates and destroys *domain participants*, each of which provide

- A *container* for all DDS entities for an application within a single domain,
- A *factory* for creating publisher, subscriber, and topic entities, and

Table 1: DDS QoS Policies

DDS QoS Policy	Description
User Data	Attaches application data to DDS entities
Topic Data	Attaches application data to topics
Group Data	Attaches application data to publishers, subscribers
Durability	Determines if data outlives the time when written or read
Durability Service	Details how durable data is stored
Presentation	Delivers data as group and/or in order
Deadline	Determines rate at which periodic data is refreshed
Latency Budget	Sets guidelines for acceptable end-to-end delays
Ownership	Controls writer(s) of data
Ownership Strength	Sets ownership of data
Liveliness	Sets liveness properties of topics, data readers, data writers
Time Based Filter	Mediates exchanges between slow consumers and fast producers
Partition	Controls logical partition of data dissemination
Reliability	Controls reliability of data transmission
Transport Priority	Sets priority of data transport
Lifespan	Sets time bound for "stale" data
Destination Order	Sets whether data sender or receiver determines order
History	Sets how much data is kept to be read
Resource Limits	Controls resources used to meet requirements
Entity Factory	Sets enabling of DDS entities when created
Writer Data Lifecycle	Controls data and data writer lifecycles
Reader Data Lifecycle	Controls data and data reader lifecycles

- A domain's *administration services*, such as allowing an application to ignore information about particular DDS entities.

DDS is topic-based, which allows strongly-typed data dissemination since the type of the data is known throughout the DRE system. A DDS *topic* describes the type and structure of the data to read or write, a *data reader* subscribes to the data of particular topics, and a *data writer* publishes data for particular topics. *Publishers* manage one or more data writers, whereas *subscribers* manage one or more data readers. Publishers and subscribers can aggregate data from multiple data writers and readers for efficient transmission of data across a network. Topic types are defined via the OMG *Interface Definition Language* (IDL) [45] that enables platform-independent type definitions.

DDS provides a rich set of QoS policies (listed in Table 1) and various properties of DDS entities can be configured using combinations of these policies. Each QoS policy has ∼2 attributes, with most attributes having an unbounded number of potential values (*e.g.*, an attribute of type character string or integer). The DDS specification defines which QoS policies are applicable for certain entities, as well as which combinations of QoS policy values are semantically compatible. DDS thus provides a wide range of QoS capabilities that can be configured flexibly [34] to meet the needs of topic-based DRE systems that have diverse QoS requirements.

For communication between DDS participants to occur efficiently, QoS policies on a publisher must be compatible with corresponding policies on a subscriber. DDS ensures this compatibility via its *Requested/Offered* (RxO) contract model, where a subscriber can specify a requested value for a particular QoS policy to be set in compatible manner between the corresponding participants. If the RxO contract matches, that QoS policy must be set both at the publishing and subscribing sides.

In the DDS RxO model the publisher declares information it has and specifies the topic and the offered QoS contract for that topic; its associated listener is then alerted of any significant status changes. The subscriber declares information it wants and specifies the topic and requested QoS contract; its associated listener is then alerted of any significant status changes. For example, if the subscriber requests to receive data reliably while publisher defines a best-effort policy, communication will not happen as requested. If the RxO does not match, the compliance of QoS policy on both sides is not requested.

2.2. DDS Routing Service

A notable earlier effort aimed at spatially decoupling DDS entities in the context of WANs is the *DDS Routing Service* [55, 37] (DDS-RS). DDS-RS establishes a content-aware interconnection service

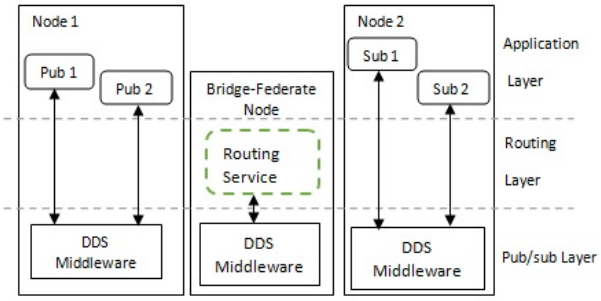


Figure 1: Layered Bridge-Federate Architecture

to enable deploying existing application in transparent manner. It allows "DDS-to-DDS" bridging between different DDS domains using transparent data transformation mechanisms (*i.e.*, applications can be reused without source code changes, even if they were developed using incompatible interface definitions).

DDS-RS also provides features (such as domain-bridging and topic-bridging) to allow data model compatibility by enabling seamless communication between data-spaces. Its domain-bridging mode enables an application running in one data-space to access a different data-space. It creates a bridge between data writers and data readers in two distinct DDS domains, even with incompatible topics; the topic-bridging allows sharing of distinct topics (with different keys, names and data types) between DDS participants. It can also act as a filter by transforming data content in topic *A* to topic *B*.

One solution to match distinct DDS-domains is to leverage existing capabilities of the DDS-RS and enhance them. We therefore implemented a Bridge-Federate using the DDS-RS API that performs both domain-bridging and topic-bridging. As described in Figure 1, the bridge-federate is a software component that adopts a layered architecture to make it possible the interconnection of disjoint DDS domains. It acts as a single federate that differs from other DDS participants to forward DDS messages between disjoint DDS domains.

Although the Bridge-Federate enables forwarding data from one domain to the other, it has several limitations that preclude it from fulfilling the bandwidth and the data-delivery requirements of enterprise DRE systems in WANs, as described in Section 4.2.1. In particular, the Bridge-Federate does not optimize bandwidth utilization between DDS participants, so it can incur scalability problems if the number of DDS participants increases significantly (as is often the case in WAN-based enterprise DRE systems).

In addition, the Bridge-Federate enables a federation-like communication model, where a central federate matches remote participants and forces all the traffic to pass through it, thus it does not provide any mechanism for failover from failure. Moreover, since DDS topics in a WAN may be routed over an internet, which is composed of different network technologies (*e.g.*, wired and wireless) with different link capacities (*e.g.*, satellite channel, mobile antenna, optical carrier, etc.), providing QoS end-to-end between different network domains is hard to be achieved and cannot be resolved with the bridge-federate based on the DDS-RS.

In contrast, our Proxy DDS approach provides more scalable and QoS-aware communication between isolated DDS domains. Proxy DDS also optimizes the utilization of network resources and enhances the scalability, as shown in Section 4.2.2. In addition, Proxy DDS incurs no single point of failure, *i.e.*, if one proxy fails all other proxies continue working correctly, and a notification is sent to alter application participants about this failure.

2.3. Related Work on DDS-based and Pub/Sub QoS Management

Enterprise DRE systems increasingly use DDS middleware to disseminate data over large-scale networks [2, 12]. To ensure end-to-end interoperability without requiring proprietary bridges [38], the OMG defined a *DDS Interoperability* (DDSI) protocol [43] that enables seamless QoS-enabled communication between DDS implementations from different suppliers. DDSI was designed to operate in stable and controllable environments (such as LANs and on-board embedded systems) since it enables plug-and-play connectivity to simplify discovery of participants. The DDSI discovery may be sufficient for small to medium networks, but not for less deterministic environments, such as WANs, where the DDSI

multicast discovery packets will be dropped by the edge networks and hence will not reach participants on remote end-systems.

Mechanisms for matching DDS domains using DDS-ESB (Enterprise Service Bus) integration architecture was proposed in [47]. Their approach enabled an integrated architecture between DDS middleware and ESB to allow exchanging information between DRE systems and the enterprise system. Although DDS-ESB focused on using DDS as a web service transport to increase interoperability between DDS implementations, it did not increase the scalability of DDS-based enterprise DRE systems. In contrast, the work we present in this paper does not use web services as a transport mechanism between disjoint DDS domains, but instead enhances the standard SIP protocol to ensure interoperability of DDS implementations and increase scalability in WANs.

The Instant Message framework [49] was proposed to integrate DDS with instant messaging services and to enable publishing DDS topics across existing IMS networks. This approach, however, uses a best-effort based point-to-point architecture that is not well-suited for the many-to-many communication model of DDS. In contrast, the work presented in this paper provides a QoS-enabled distributed architecture, which enables dynamic resource management under network load for many-to-many communication, *i.e.*, all Proxy DDS components communicate with each other at the same time.

Finally, many pub/sub standards and technologies (*e.g.*, Web Services Brokered Notification [42] and the CORBA Event Service [17]) have been developed to support distributed event-based systems [22]. These standards and technologies, however, do not provide fine-grained and robust QoS support, but instead focus on issues related to monitoring run-time application behavior in a scalable manner. They therefore do not address key open issues, such as end-system QoS policies for reserving the network resources and simplifying the control of the network behavior [28].

In contrast, the work presented in this paper enables more effective control of applications in enterprise DRE systems via the Proxy DDS (which maps the bandwidth, latency, and jitter QoS requirements to the underlying WAN as described in Section 3.2.2) and NetQSIP (which allocates the network resources as required by DRE applications without over-provisioning the network equipments).

2.4. Related Work on Other Middleware and Network-level QoS Management

Most applications of DDS in DRE systems either use LANs or small-scale WANs whose QoS properties are relatively stable. In these environments, DDS middleware uses IP multicast to allow publishers and subscribers to operate in a scalable peer-to-peer fashion. In large-scale WANs (such as the Internet or enterprise intranets), however, IP multicast is often disabled for performance and security reasons.

Overlay networks have been studied as an alternate approach to shield applications from the heterogeneity present in WANs. For example, [29] propose integrating an open-overlay framework as part of the middleware, thereby enabling end-systems to support multiple overlays running composite protocols to allow dynamic reconfigurability. Hermes [48] provides an overlay broker for event-based middleware to perform the content-based routing. Likewise, Tiny DDS [8] is a pub/sub middleware that operates on top of an Overlay Event Routing Protocols (OERP) layer for event routing. Moreover, [14] propose a *distributed hash table* (DHT)-based overlay network to perform scalable and efficient discovery scheme for DDS entities over WANs.

In this related work, however, discovery messages are sent to a limited number of remote nodes, even communicating in peer-to-peer fashion. This work also has a several limitations with enabling end-to-end QoS provisioning, scalability, and performance of the communication. In contrast, the work presented in this paper does not modify the middleware layer to add an overlay network. Instead, the Proxy DDS components provide diverse DDS QoS profiles for LANs and WANs and provides proxy-to-proxy communication over WANs, thereby supporting scalable multicast, as described in Section 3.2.1.

Authors in [33] proposed to adopt an overlay network that allows *Application-Level Multicast* (ALM). Members belonging to the same multicast group are interconnected through an overlay network [35],

where end-systems rather than routers are responsible for replicating messages that are dispatched over several distinct out-going links [26]. ALM over overlay networks has several drawbacks, however, including lower performance than IP multicast, scaling limitations, and degraded network resources caused by increasing traffic load.

Prior middleware solutions for network QoS management focus on how to add network QoS services for CORBA-based communication [19, 18]. A large-scale event notification infrastructure for topic-based pub/sub applications has been suggested for peer-to-peer routing overlaid on the Internet [54]. Those approaches can be deployed only in a single-domain network, however, where one administrative domain manages the whole network.

Extending network QoS solutions to the Internet can result in traffic specified at each end-system being dropped by the transit network infrastructure of other domains [6]. For example, [63] presented a network communication broker to enable per-class QoS for multimedia collaborative applications. Even if this network broker enhanced the QoS allocation by differentiating the traffic processing at the network edges, it supported neither mobility service management nor scalability since it adds complicated interfaces to both applications and middleware for the QoS notification.

The peer-to-peer architecture described in [4, 16] is designed to ensure data delivery with expected QoS-levels to reliably disseminate events and balance data distribution load in non-multicast-enabled WAN deployments. Likewise, [62] proposes a service-oriented architecture to integrate remote DRE systems over the Internet based on the data-centric publish-subscribe model. These prior efforts, however, do not support network resource provisioning required to ensure end-to-end QoS. Finally, [23] focused on priority reservation and QoS management mechanisms that can be coupled with CORBA at the OS level to provide flexible and dynamic QoS provisioning for applications running in DRE systems.

The work reported in this paper enhances prior research on QoS management at the network layer by integrating QoS along two key dimensions: (1) the *horizontal direction*, *i.e.*, between different adjacent layers in the application, middleware, and network, and (2) the *vertical direction*, *i.e.*, within a particular layer. In particular, our NetQSIP framework maps application flow requirements into the DDS layer to allow end-to-end QoS provisioning.

Our prior work [30] on Velox used an MPLS tunneling mechanism to propagate DDS QoS over an overlay model. Velox statically created a logical tunnel between remote DDS participants to conduct QoS negotiation and resource reservations. Any time communication failed, however, the user had to reconfigure the network manually. Moreover, application requirements could not change during runtime communication between participants.

This current paper enhances our prior work on Velox and overcomes its limitations by defining the Proxy DDS, which communicates with the other proxies without using any tunneling mechanisms. As described in Section 3.2.1, our Proxy DDS components do not require an overlay model to match distributed participants through the network. Moreover, our NetQSIP framework provides dynamic QoS management and simplifies resource management by using SIP to automate end-to-end QoS provisioning, which described in Section 3.2.3.

3. Supporting DDS QoS Policies over WANs

This section focuses on the following two capabilities we developed to support DDS QoS policies over WANs:

- **Interconnecting isolated DDS domains via the Proxy DDS**, which matches isolated DDS domains to distribute topics and reduce the traffic exchanged between remote DDS participants in a WAN.
- **Simplify network resource management via the NetQSIP policy-driven QoS framework**, which leverages the Proxy DDS, SIP, and the *Common Open Policy Service* (COPS) protocol for *Diffserv Resource Allocation* (COPS-DRA) [56] to automate end-to-end QoS provisioning and mapping DDS sessions from application participants to the underlying WAN.

3.1. Supporting DDS QoS Policies Efficiently and Scalably in WANs

Below we describe several challenges that arise when attempting to use conventional DDS implementations for enterprise DRE systems running in WANs.

3.1.1. Challenge 1: Mapping DDS domains to network-level mechanisms

Context. DDS middleware provides QoS policies and mechanisms for DRE systems that run in stable and controllable LANs. As these environments grow in complexity and scale, it becomes necessary to deploy DDS-based DRE systems in WANs. End-to-end QoS requirements for these enterprise DRE systems still require bandwidth, lossless delivery, and interoperability even among disjoint DDS domains. In particular, DDS implementations should disseminate messages from publishers to subscribers under bandwidth fluctuations without overloading the capacity of the WAN, even between disjoint DDS domains.

Problem. The DDS domains of enterprise DRE systems are deployed over different network-domains, where each one is administrated only by a single provider, on which the policies and administration can differ, *i.e.*, one provider may not have the full control over the network. It is hard to obtain the expected QoS at the overlay level, and they may have various restrictions (such as not allowing multicast traffic, using different protocols and policies that depend on how routing tables are created, etc.) that may affect QoS.

The DDS standard does not specify how to distribute topics between independent data spaces. For example, OpenSplice DDS (`www.prismtech.com/opensplice`) provides network partitions that enable WAN communications, but does not propagate QoS across a WAN. Likewise, RTI Connext DDS (`www.rti.com/products/dds/`) provides a unicast service for propagating QoS policies across a WAN, but does not provide a mapping of these policies in edge routers, which require advanced APIs for queue management. In both cases, multiple DDS domains are not supported unless custom gateways/bridges are used, which still does not resolve WAN-level QoS problems, such as scalability and delivery guarantees end-to-end.

One approach is to use point-to-point IP encapsulated tunnels, such as VPN-MPLS [50, 41, 1]. Our prior work on the Velox framework [30] used MPLS tunneling mechanism to propagate DDS QoS over an overlay model. Although this work demonstrated how standard DDS QoS policies can be supported end-to-end in WANs, the Velox framework had the following limitations:

- It did not support multiple DDS domains with inter-domain traffic; rather it assumed that all participants shared the same global data-space through different IP network domains.
- Its QoS management was *static*, *i.e.*, the network configuration was done only once during system initialization and hence it could not support dynamically changing application requirements.
- It used the *Multi-Protocol Layer Switching* (MPLS) [41] tunneling mechanism to propagate DDS QoS over an overlay, which lacks robust means to ensure end-to-end semantic consistency of QoS policies and requires manual user intervention during initial configuration, as well as reconfiguration after failures.

Solution summary. Section 3.2.1 describes how we address *Challenge 1* by introducing the Proxy DDS, which is a software component for enabling efficient and seamless communication between participants in isolated domains. The Proxy DDS scalably decouples DDS participants in time/space and minimizes traffic exchanged between these participants.

3.1.2. Challenge 2: Enabling scalable multicast-based DDS discovery in WANs that cross IP network domain boundaries

Context. The DDS discovery service is used by DDS applications for automatic publication and subscription discovery and enables QoS-enabled DRE system architectures without the need for any centralized broker. This service periodically sends *heartbeat* discovery messages to remote nodes to check whether

these nodes are still alive and able to receive data (*e.g.*, processes discovery events generated by underlying DDS middleware and notify DDS participants whenever change occurs in the discovery service). Communication between nodes uses the standard DDS request/offered (RxO) contract in which each participant presents compatible QoS parameters to enable sending and/or receiving of data.

Problem. To enhance DDS applications QoS over WANs, DDS' discovery mechanisms should enable the efficient and scalable exchange of topics between distinct data spaces. Typical LAN-based implementations of DDS uses IP multicast, which does not scale to WANs. DDS multicast services in WANs must therefore deal with the following deployment problems:

- Multicast discovery messages are often blocked in edge routers and do not reach remote networks since IP multicast is often disabled whether for safety or performance reasons.
- IP multicast forces routers to maintain per-group state (*e.g.*, members to the group) that (1) is highly variable over time and (2) imposes scalability and reliability penalties. IP multicast often incurs performance limitations on routers (*e.g.* it fails to filter incoming messages beyond a few dozen multicast groups), so IP multicast does not scale to a large number of groups [60].
- The deployment of IP multicast requires all routers to be appropriately configured, which makes it quite impractical in large scale or open settings, *i.e.*, where one does not have full control over the networking environment.

Solution summary. Section 3.2.2 describes how we address *Challenge 2* by describing NetQSIP, which is a middleware framework that bridges the gap between DDS applications and SIP to allow communication between distributed participants over isolated DDS domains. It also supports end-to-end QoS signaling to all distributed participants to leverage the limitations of IP multicast without affecting the discovery protocol.

3.1.3. ***Challenge 3: DDS QoS provisioning with the DDS domain overlays that subsume multiple network domains is hard***

Context. To simplify the development and evolution of applications for enterprise DRE systems, the details of interacting with low-level network QoS mechanisms for provisioning and resource allocation should be encapsulated within a middleware signaling service API. One way to implement such a signaling service involves using the *Session Initiation Protocol* (SIP) [53, 10] to request only the desired resources for the DDS application and avoid the over-provisioning of WAN resources [40].

Problem. The following problems must be addressed to use SIP effectively as the basis for a high-level signaling service for DDS:

- **Semantic gap between SIP and DDS.** SIP uses predefined *Session Description Protocol* [31] (SDP) messages to allow the communication between SIP-compliant clients. These messages use specific media attributes for SIP that do not include any fields for the standard DDS QoS policies.
- **Respecting the DDS requested/offered (RxO) contract model.** DDS QoS policies should respect the RxO contract model that matches publisher QoS needs to subscriber QoS capabilities. DDS implementations use the DDS RxO contract model described in Section 2.1 to ensure that communication only occurs when a QoS policy requested from one participant is compliant with a QoS policy offered by another participant. SIP by itself does not support the RxO model.
- **Choice of the exact DDS QoS policies.** Most DDS QoS policy settings are not modifiable at run-time and must therefore be designated before compiling the DDS applications themselves. Only QoS policies that control simultaneously topics, data readers, and data writers can therefore be used to control the end-to-end path. These policies must be generic for all DDS implementations to allow interoperability between dif-

ferent middleware vendors. Since SIP does not understand the semantics of DDS, it cannot natively support such QoS configurations.

- **Incompatibility of a SIP codec with a DDS session.** SIP allows applications to select the appropriate codec to adapt their bandwidth with the network load. SIP clients must therefore agree on the codec they want to use to adapt their sending rate to fulfill the required QoS. The caller sends a codec map as part of its initial INVITE message so the receiving application can select the appropriate codec and adapt their bandwidth to the network load. In contrast, however, DDS has no notion of a codec to adapt the publish rate. It is therefore necessary to adapt DDS QoS profiles to SIP signaling mechanisms with respect to their traffic profile. In particular, a DDS signaling service needs to specify the network QoS support and the session management that can provide the different DDS QoS properties.

- **Lack of resource reservation mechanisms in edge routers.** Although SIP can transport the QoS requests (including bandwidth, latency, and class of service) sent by DDS applications within their signaling messages to notify the network about the application's QoS requirements [59], it cannot reserve the resources to fulfill these requirements. Advanced QoS provisioning mechanisms are therefore needed to implement QoS requests in edge routers of each network domain.

Solution summary. Section 3.2.3 describes how we address *Challenge 3* by integrating Proxy DDS and NetQSIP so that Proxy DDS components carry QoS policy messages in SIP control messages. NetQSIP simplifies resource allocation by automating end-to-end QoS provisioning and provides service-level differentiation required by applications in DRE systems.

3.2. Supporting Enterprise DRE Systems over WANS using Proxy DDS and SIP Signaling

To address the challenges described in Section 3.1, we now describe the structure and functionality of Proxy DDS and NetQSIP. Proxy DDS enables efficient and scalable communication between isolated domains in a WAN. NetQSIP simplifies DDS resource management by automating end-to-end QoS provisioning over WANs. NetQSIP uses Proxy DDS to match distinct and isolated DDS domains over different IP network domains, SIP to support end-to-end QoS signaling, and the COPS protocol as a QoS manager for dynamic resource allocation over DiffServ infrastructure.

Below we outline how the challenges described in Section 3.1 are addressed in the remainder of this section:

1. The Proxy DDS matches isolated DDS domains and optimizes the bandwidth usage between remote participants, as described in Section 3.2.1.
2. DDS QoS policies are mapped into SIP messages to carry the QoS requests from the application to the network, as described in Section 3.2.2.
3. NetQSIP uses these SIP messages to reserve QoS mechanisms in edge routers and provide end-to-end resource reservation, as described in Section 3.2.3.

*3.2.1. **Resolving Challenge 1: Create a Proxy DDS to Match Isolated DDS Domains***

The Bridge-Federate presented in Section 2.2 successfully interconnects DDS-based DRE systems situated in isolated DDS domains. This solution only partially solves *Challenge 1*, however, due to excessive flow duplications when the number of participants increases, as shown by our empirical results in Section 4.2.1. To address this problem, therefore, we enhanced the Bridge-Federate functionality via a new component we called the *Proxy DDS*, which does not use the DDS-RS approach (described in 2.2).

The Proxy DDS provides the same functionality as the Bridge-Federate model (*i.e.*, it matches isolated DDS domains). It also optimizes DDS bandwidth, even if used over heterogenous IP network domains. Instead of using a single software component

(the DDS-RS-based Bridge-Federate model), Proxy DDS can deploy many proxies in many IP network domains, thereby providing the following capabilities:

Capability 1: Matching isolated DDS domains over different network domains and optimizing end-to-end bandwidth utilization. Each Proxy DDS component subscribes to all topics sent by participants in its IP domain, selects those of interest (*i.e.*, topics that will be sent to other remote participants in remote DDS domains belonging to remote IP network domain), and retransmits them to the (virtual) global data space, as shown in Figure 2.

To receive DDS data, each proxy subscribes to those topics from the global data space and publishes (replicas of) them in its IP domain for interested subscribers. Each Proxy DDS component performs the deployment and configuration of the DDS implementation with respect to application requirements (such as latency, bandwidth, class of service, etc.). It specifies only the topics exchanged in the virtual global data space with remote proxies.

Capability 2: Supporting Multiple QoS profiles for LAN and WAN. Each Proxy DDS communicates with both the local participants (in its network domain) and the other remote proxies over a WAN. It therefore uses the following distinct DDS QoS profiles:

- The first QoS profile ensures consistency between DDS participants in the same IP domain, *i.e.*, the QoS profile should be consistent between all the participants and the Proxy DDS in the same DDS domain (LAN QoS-profile in Figure 2),
- The second QoS profile enables communication between different DDS proxies to distribute data in an inter-proxies manner (Proxy-to-Proxy QoS-profile shown in Figure 2).

Capability 3: Ensure interoperability between the DDS QoS-Policies and network QoS policies. Each Proxy DDS uses the DDS *Transport_Priority* QoS to mark the DiffServ DSCP field of the packets sent to other remote proxies. For example, assume that Proxy DDS_1 receives a topic A from the participants that belong to its IP domain with the DSCP value equal to 46, which is the IP transport priority that corresponds the *Expected Forward Per-Hop Behavior* (EF-PHB). Proxy DDS_1 then transforms and forwards the messages corresponding to topic A with the DSCP value 10, which corresponds to the *Per-Hop Behavior Assured Forward (PHB-AF11).*

3.2.2. ***Resolving Challenge 2: Interface Mapping between DDS and SIP***

To bridge the semantic gap between DDS QoS policies and SIP control messages, we developed NetQSIP, which is middleware that resides between the DDS application and the underlying network service plane (SIP session layer). NetQSIP helps resolve *Challenge 2* described in Section 3.1.3 by extending the semantics of the SIP media flow attribute (the so-called "a" attributes) included in the SIP/SDP message (a set of lines of the form $< attribute >=< values >$), as shown in Figure 3. Both the publisher and the subscriber use the information carried within these attributes to negotiate with the underlying network on how the QoS reservation should be done using a new predefined precondition.

As discussed in Section 3.1.3, SIP was designed to support codec negotiation between clients. DDS does not support the notion of codecs, but instead it has several QoS parameters that may change at runtime. Those QoS parameters are adapted in the context of SIP to enable dynamic publish rates for DDS-based DRE applications.

For each DDS topic, NetQSIP defines several QoS profiles (stored in its topic map) that can replace the codec map used by SIP. As an enhancement to the standard SDP in [51], we devised new DDS QoS attributes and syntax to incorporate into the signaling procedure. The DDS publisher and the subscriber exchange SIP messages that include the resource reservation demands in the topic map and the current status to support the QoS demands in each direction. The "*qos-dds*" token described in Table 2 specifies the QoS policies carried within the SIP/SDP INVITE message for the dynamic network QoS allocation.

Table 3 also shows the new "qos-dds" token we created to support the specific DDS-based QoS re-

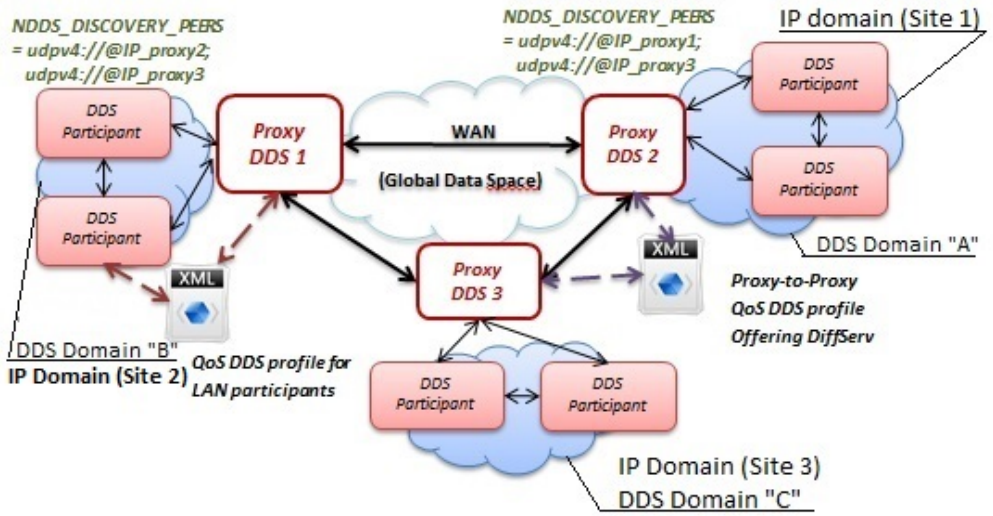


Figure 2: Matching Disjoint DDS Domains Over Different Network Domains

Field	Description
Deadline	Data reader expects a new sample updating the value of each instance at least once every deadline period. Data writer indicates that the application commits to write new value for each instance managed by this data writer at least once every deadline period. The Deadline is a duration "0 0".
Latency	The delay from data writing until its delivery is inserted in the receiver's application cache and the receiving application is notified of the fact. The Latency Budget is duration "0 0".
Reliability	the reliability of the service. Could be Reliable ("R") or Best Effort ("BE").
Priority	Transport Priority is a hint to the infrastructure used to set the priority of the underlying transport used to send data in the DSCP field for DiffServ. This value is presented as an integer.

Table 2: Encoding the DDS QoS policies in the "qos-dds" Fields

quirements, which contains the Offer/Answer SDP extension specified in RFC3264 [51].

Table 3: SIP INVITE Message with the New "a" Including the qos-dds Token

```
INVITE sip:ahkiri@laas.org SIP/2.0
Via:SIP/2.0/UDP 193.49.97.17
Via:SIP/2.0/UDP 193.49.97.81
From:sip:ahkiri@iptel.org
To:sip:ahkiri@laas.org
CSeq: 1 INVITE
o = akram 53655765 2353687637 IN IP4 193.49.97.18
s = -
c = IN IP4 ahkiri@laas.org
t = 0 0
b = AS:512
a=qos-dds-send: current 0 20000000 0 20000000 R 46 send
a=qos-dds-recv: current 0 20000000 0 20000000 R 32 recv
```

The "qos-dds" token identifies a QoS mechanism that is supported by the entity generating the session description. A token that appears in a "qos-dds-send" attribute identifies DDS QoS policies supported by data writers to help the resource reservation for traffic sent by publishers generating SDP messages. A token appearing in a "qos-dds-recv" attribute identifies the DDS QoS policies that can be

```
a = status-type SP qos-dds SP direction-tag
    status-type = ("desired" "current" "acceptable" "unacceptable")
    SP = space
    qos-dds = ("deadline" "latency" "reliability" "priority")
    direction-tag = ("send" "recv" "send-recv")
    status-type:
     - desired: DDS participant indicates the DDS QoS he wants to use.
     - current: informs the remote DDS participant about the QoS DDS
     to be used in the current session.
     - acceptable: indicates that the DDS QoS is acceptable and
      compliant with the R/O contract.
     - unacceptable: indicates that the current DDS QoS policies
     are not compliant with the R/O contract (reject the communication).
    direction-tag :
     - send: DDS participant indicates it desire to send topics (Publish mode).
     - recv: DDS participant indicates he wants to receive topics (Subscribe mode).
     - send-recv: DDS participant indicates that he wants to send and receive topics
       (both Publish mode and Subscribe mode are supported).
```

Figure 3: The New SDP Attributes for DDS with Preconditions

supported by data readers to reserve the resources for traffic coming from DDS publishers. These extensions remain transparent to the edge router and work seamlessly in the context of DiffServ networks.

We used the SDP session description attributes presented in Figure 3 to describe the new offer/-answer extension that supports signaling for DiffServ. These attributes describe a DDS communication request for total bandwidth of 512 kbits per second (designated in line "b") with the following "qos-dds-send" and "qos-dds-recv" attributes of the DDS media session:

Offer/answer behavior. When using "qos-dds-send" and "qos-dds-recv" attributes, an offer/answer negotiation is done between the publisher and the subscriber to allow endpoints to load a list of DDS QoS profiles. Participants negotiate the direction in which those profiles are exchanged with respect to both preconditions [9] and DDS changeable table parameters [44]. Participants also use other QoS parameters (*e.g.*, the bandwidth, jitter, delay parameters described in RFC3312 [11]) to negotiate per-Class QoS setting for DiffServ.

Offer behavior. The publisher includes "qos-dds-send" flow information in the publication direction to inform subscribers about the DDS QoS profiles supported by publishers. Similarly, a participant can use "qos-dds-recv" attributes to specify which kind of QoS policies can be supported at subscribers.

Answer behavior. After receiving an offer from a remote participant with the "qos-dds-send" attributes, NetQSIP translates those attributes into network QoS settings for the resource reservation process. These attributes correspond to the DDS QoS policies within the QoS profiles supported in the subscriber direction. A participant uses these attributes in a "qos-dds-recv" attributes in the answer. For example, when a participant receives an offer with "qos-dds-recv" attributes, the answerer uses the corresponding QoS translation mechanisms it supports in the publisher direction and then includes them in the "qos-dds-send" attributes.

3.2.3. ***Resolving Challenge 3: Supporting PubSub QoS properties over WAN***

To support QoS signaling and resource provisioning over WANs, the Proxy DDS uses NetQSIP to transport DDS QoS policies within its control messages. These messages include the information (*e.g.*, bandwidth, latency, priority, etc.) about applica-

tion requirements. DDS applications interested in publishing data can specify the media description by sending SIP control message to the network. If their requests are accepted, then the required resources are allocated, and they can securely exchange data. These features are part of the NetQSIP framework, which provides:

- A *resource allocator* (defined as a part of the Proxy DDS) to use these messages for signaling the QoS to the network on behalf of an application;
- A *policy-based network configurator* to enforce the resource allocation in the network infrastructure.

Enhancing the Proxy DDS for QoS Signaling. In addition to the functionality presented in Section 3.2.1, the Proxy DDS maps DDS sessions between an application and the underlying network. We enhanced the Proxy DDS to include the DDS QoS policies using the JAIN SIP [25] [24] proxy. We also extended it to parse the new "a" attribute that support the QoS policies described in Section 3.2.2.

The enhanced Proxy DDS acts as a resource allocator that communicates with the DDS participant via the SIP/DDS interface and intercepts the DDS QoS policies from the SDP message. It also communicates with the QoS configurator to provision key traffic control, classification, and shaping properties of the priority queue. In addition, it infers the session characteristics (such as bandwidth, the priority level (DSCP), the sender/receiver IP addresses/-Ports) from an XML file (an excerpt of which is shown in listing 1).

Listing 1: QoS Policies Provided by DDS Middleware

```
<?xml version="1.0"?>
  <NetworkInterfaceAddress>193.49.97.81
  </NetworkInterfaceAddress>
  </General>
     <Partitioning>
      <GlobalPartition Address="unicast"/>
        <GlobalPartition Address=
   "193.49.97.17"/>
     </Partitioning>
 <Channels>
  <Channel default="true" enabled="true"
   name="BestEffort" reliable="false">
   <PortNr>54100</PortNr>
        <Resolution>10</Resolution>
        <FragmentSize>554</FragmentSize>
        <AdminQueueSize>4000</AdminQueueSize>
         <Sending>
          <DiffServField>18</DiffServField>
          <MaxBurstSize>500</MaxBurstSize>
          <QueueSize>400</QueueSize>
          <ThrottleLimit>1024</ThrottleLimit>
          <MaxRetries>10</MaxRetries>
          <RecoveryFactor>3</RecoveryFactor>
        </Sending>
        <Receiving>
        <ReceiveBufferSize>1000000</ReceiveBufferSize>
         <Scheduling>
       <Priority>60</Priority>
       <Class>Realtime</Class>
       </Scheduling>
        </Receiving>
</Channel>

<Channel enabled="true" name="Reliable"
              reliable="true">
          <PortNr>54110</PortNr>
        </Channel>
      </Channels>
```

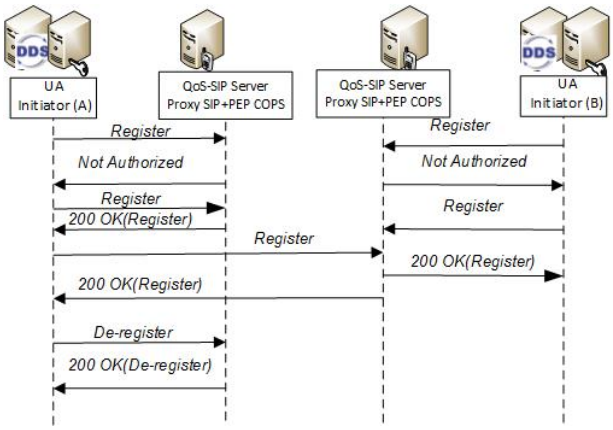


Figure 4: Session Register and Deregister with proxies

For example, since SIP does not allow sending data in multicast, we configured the source and the destination IP addresses in the attributes "NetworkInterfaceAddress" and "GlobalPartition," respectively. The attribute "PortNr" specifies the port number on which each DDS participant receive data. Likewise, the attribute "DiffServField" allows specifying the value that will be used to mark the DSCP field, which may change at runtime if a participant decides to change it. The attributes "MaxBurstSize" (topic size in bytes) and "Resolution" (in ms) describe the publish rate (*i.e* throughput calculated at runtime

by the DDS middleware) at which the publisher can send DDS topics. The publish rate is defined by the relation: $rate = (MaxBurstSize \times 8 \times (\frac{1000}{Resolution}))$ (b/s).

The application contacts the Proxy DDS during the registration phase to access to the registrar, as shown in Figure 4.

The registrar receives two the QoS requests sent by the publisher and the subscriber. It then forces all signaling messages to pass across it with the header "record route" and intercept those messages to analyze the information about the session. After the publisher is configured to make and receive calls, NetQSIP processing begins with the registration phase in the registrar (user registration in the SIP location database).

We considered the case where the registration is done by both remote SIP proxies of each network: the publisher sends a register request to the proxy belonging to its network domain which intercepts it and adds the address of record (AddressOfRecord) to enforce messages to pass through the registrar. After the destination address is found in its local database (local database to avoid the usage of a DNS), it redirects the message to the destination. The remote proxy (belonging to the same network domain of the subscriber) intercepts this message, checks if the subscriber belongs to its domain, and forwards the INVITE message to it if it belongs to the domain.

Network QoS configurator for resource reservation. To reserve QoS in the edge-router, we developed a resource configurator based on the COPS-DRA [56] protocol as part of the NetQSIP framework. NetQSIP also integrates the resource allocator in the Proxy DDS to translate the DDS QoS policies in the SIP/SDP messages into network QoS, and negotiate the DDS QoS policies on behalf of the application. These translation requests are submitted to the resource configurator which configures the priority queues of the edge-router based on the DSCP marking for IP packets with the *Transport_Priority* DDS QoS setting. Below we describe the QoS negotiation during the signaling process.

Session initiation. The publisher (shown as DDS participant "A" in Figure 5) sends the INVITE message including a list of DDS QoS profiles with the source/destination IP addresses, the ports number, the DDS QoS attributes, the DSCP field descriptor, and the media descriptor.

This message indicates the direction for exchanging DDS topics between remote participants (forwarding and back to topic flows with "qos-dds-sendrecv" token). When the publisher wants to change the DDS QoS profile, it sends a message including "qos-dds-send" token with the "desired" status. If the publisher wants to receive the DDS QoS profiles, it sends message including "qos-dds-recv" token also with the "desired" status. In this case, the following two situations can occur: **Situation 1.** The remote DDS participant "B" (subscriber) receives the list of DDS QoS profiles (topic map) and intersects it with its existing profiles. For DDS to match the publisher with the subscriber, they both need to agree the RxO contract, which indicates their QoS profiles are compliant. They should find the common parameters in the QoS profiles (*i.e.,* DDS QoS parameters described in Table 2 and carried by SIP invite message). Since several QoS profiles are grouped in topic map, publishers and subscribers only need to select the compliant ones, (*i.e.*, the publisher profile should be compliant with the subscriber profile) via the following steps:

1. If subscriber "B" supports at least one DDS QoS profile satisfying the RxO contract, it will can reply with an "acceptable" status. It therefore selects the first DDS QoS profile compliant with the RxO contract and compares the IP addresses and port numbers on which it can receive DDS topics as described in the XML file shown in Listing 1. As described in Figure 5, the subscriber then sends the response "183 Session In Progress" with the "acceptable" status to its Proxy DDS, which forwards the reply to the remote publisher "A". The proxy forwards this message to the publisher and indicates that the session has been established in both directions.
2. Proxy DDS next sends a reservation request via the NetQSIP resource allocator, which triggers

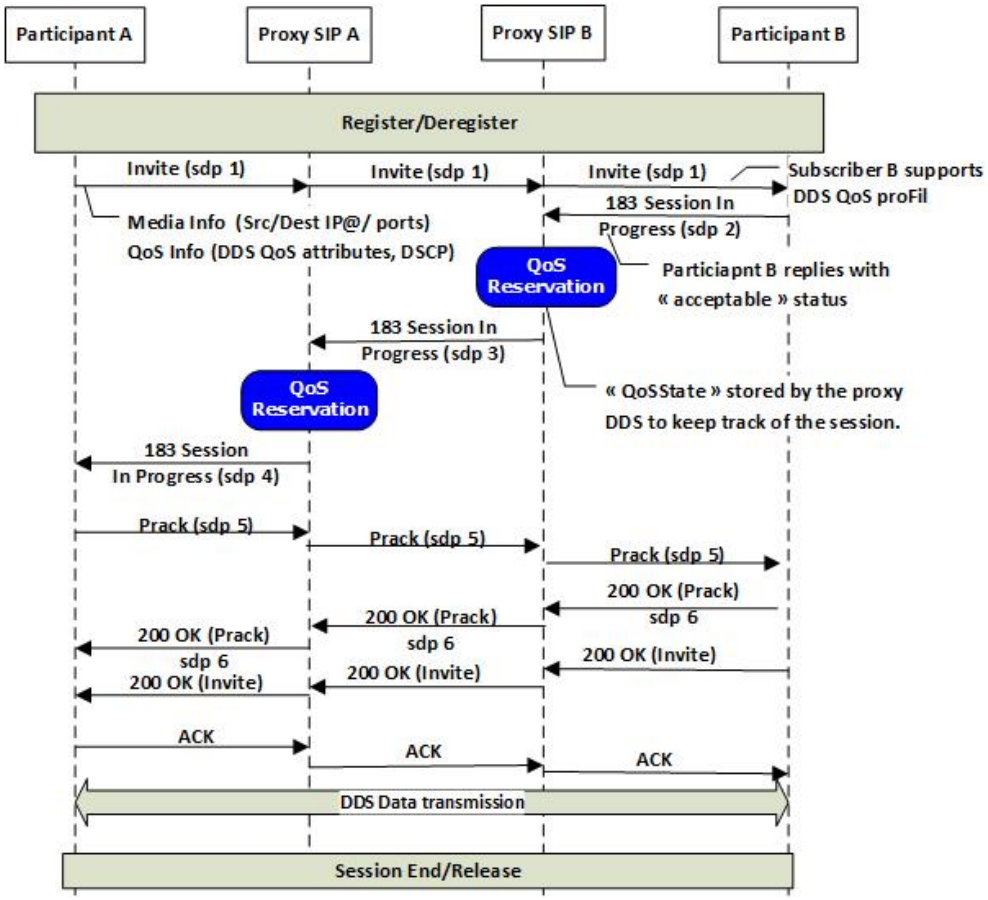


Figure 5: QoS Reservation within SIP/DDS Session

the resource reservation process as described in Figure 6. The NetQSIP resource configurator intercepts this request to negotiate the required QoS with the edge router. This information—called "QoSState"—is stored by the proxy DDS to keep track of the session.

3. Proxy DDS then sends a QoS request (embedded within the SDP message) to the resource allocator in the remote domain "B" to request the QoS in both directions. The publisher subsequently sends a "Prack" [52] message to confirm the 183 SESSION IN PROGRESS has been received and waits for subscriber "B" to send the 200OK message (for Prack).

4. The session establishment is confirmed when both proxies (for participants "A" and "B") exchange 200OK and ACK messages. The publisher starts transmitting data to remote subscribers and the data path is established.

The "qos-dds" attributes include the DSCP priority tag the publisher has added to SIP/SDP INVITE messages (described by the *Transport_Priority* DDS QOS policy). The resource allocator sends the request message (REQ message) to the resource configurator to install the desired QoS. The resource configurator analyzes the request (consults its database for verification), performs the DiffServ resource allocation, and replies to the resource allocator with a

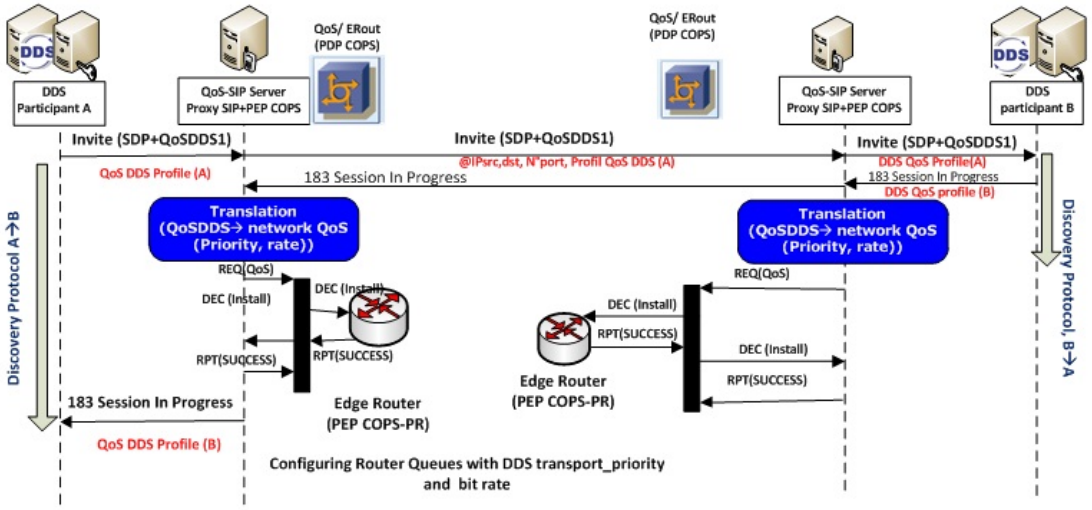


Figure 6: Session Establishment with DDS QoS Support

decision message (DEC) message.

The resource configurator uses the DSCP value to classify DDS traffic regarding their priority in each edge-router. An XML parser translates the following QoS settings: latency budget, deadline in the "a" field into bandwidth, latency, and jitter. The resource allocator subsequently generates a report message "RPT" to indicate the success of the decision. The 183 SESSION IN PROGRESS message is also sent to the publisher's Proxy DDS to indicate successful resource allocation in its direction.

The DDS discovery protocol performs discovery in both direction. The neighbor discovery is provided from the information (IP addresses) provided by the proxy and registrar near to each DDS participant, so that the interconnection between distinct DDS domains can be easily performed. NetQSIP matches DDS data writers and data readers by sending pub-/sub declarations embedded within DDS messages, which include a *Globally Unique ID* (GUID), QoS policies, etc.

Situation 2. Two cases must be considered if the subscriber does not support any DDS QoS profile satisfying the required QoS (*e.g.*, *Transport_Priority*=0). The first case is shown in Figure 7 where no DDS QoS profile can be used to implement the network QoS. This case corresponds to the default best-effort service without QoS that can be established at the network level. In the second case there is a violation of the DDS requested/offered (RxO) contract between the publisher and subscriber. Since session establishment cannot occur the NetQSIP proxy thus sends a rejection message back to the first DDS publisher with the status token fixed at "Unacceptable" to reject the communication.

Session Liberation The session termination is shown in Figure 8. When the publisher wants to finish the session, it sends a BYE message to the remote subscriber, which replies with a 200OK message (for BYE). The deregistration procedure from the proxies is performed similarly to the registration phase: a REGISTER message, almost similar to the registration message, is sent with the the "Expires" field set to 0. When the publisher sends a BYE message to its nearest Proxy DDS component, this component sends the QOS FREE message to the resource allocator to request the liberation of the resources. The resource configurator receives this message and sends an UNINSTALL message to release the allocated resources at

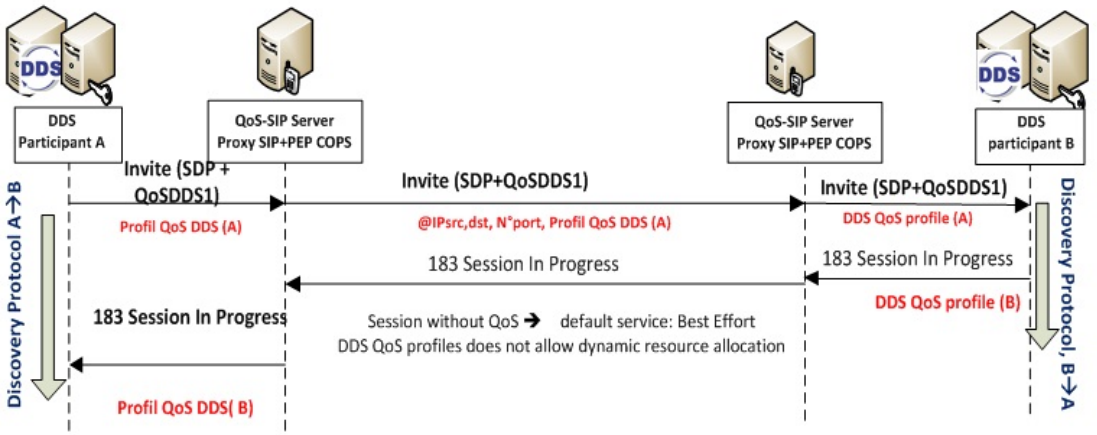


Figure 7: Session establishment without QoS (Best-Effort service)

the end of session. After releasing the resources, the subscriber sends the 200 OK message to reply to the BYE message.

4. Experimental Results and Discussions

This section analyzes the results from experiments we conducted to evaluate the efficacy of the Proxy DDS components and NetQSIP framework presented in Section 3.2. Section4.2 evaluates the performance of the Proxy DDS against the Bridge-Federate in the best-effort QoS architecture described in Section 4.2.1. Section4.3 evaluates NetQSIP in terms of its timing behavior, overhead, and end-to-end latencies observed in different scenarios.

4.1. Hardware and Software Testbed and Configuration Scenario

The performance evaluations reported in this paper were conducted in the Laasnetexp testbed shown in Figure 9. Laasnetexp consists of a server and 38 dual-core machines that can be configured to run different operating systems, such as various versions of Windows and Linux [46]. Each machine has four network interfaces per machine using multiple transport protocols with varying numbers of senders, receivers and 500 GB disks. The testbed also contains four Cisco Catalyst 4948-10G switches with 24 10/100/1000 MPS ports per switch and three Juniper M7i edge routers connected to the RENATER network [3].

To serve the needs for the emulations and real network experiments, two networks have been created in Laasnetexp: a three-domain real network (suited for multi-domain experiments) with public IP addresses belonging to three different networks, as well as an emulation network.

In our evaluation scenario, a number of real-time DDS applications sent their monitored data to each other so that appropriate control actions are performed by the military training and Airbus Flight Simulators we used. Figure 9 shows several simulators deployed on EuQoS5-EuQoS8 blades communicating based on the Opensplice DDS middleware implementation [4]. To emulate network traffic behavior, we used a traffic generator that sends UDP traffic over the three domains with configurable bandwidth consumption.

[3] `http:www.renater.fr`
[4] `http://www.prismtech.com/opensplice`

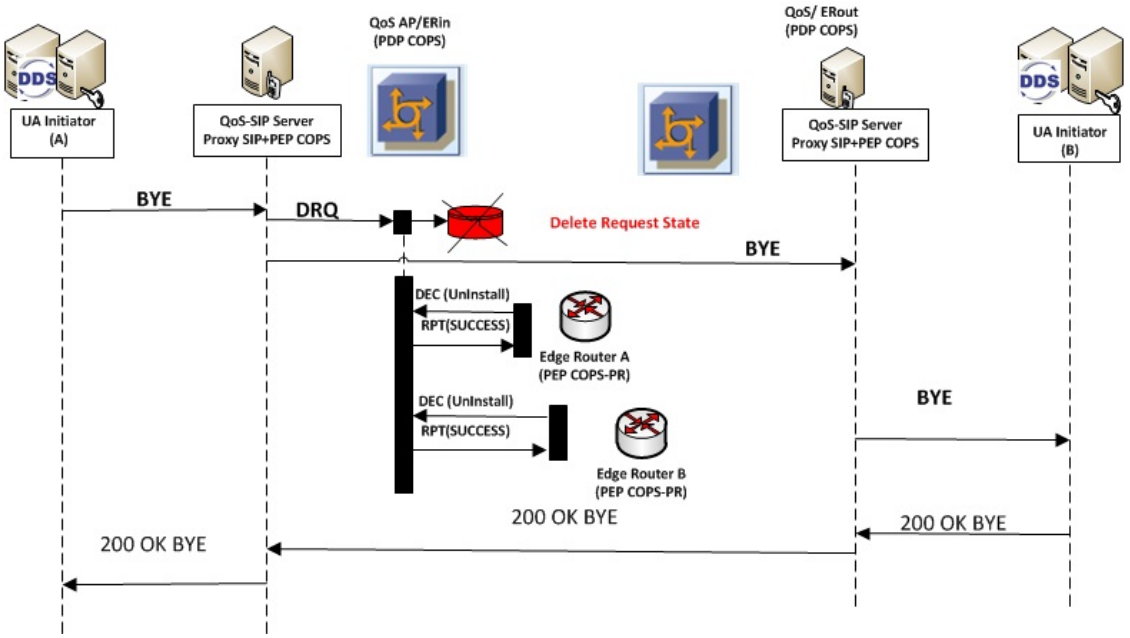


Figure 8: Session Termination and Resources Release

4.2. Evaluation of the Proxy DDS in a Best-Effort WAN

Below we present the results of experiments conducted to evaluate the Bridge-federate in Section 4.2.1 and the proxy DDS in Section4.2.2. These results evaluate the bandwidth and the overhead of both solutions and show the impact of using the Proxy DDS to replace the Bridge-Federate implementation described in Section 2.2 .

4.2.1. Evaluating the Bridge-Federate Model

Below we describe the results of evaluating the performance and overload of using the Bridge-Federate model to interconnect disjoint DDS domains in Domain-Bridge model and compare these results with an implementation of the Topic-Bridge model.

Evaluation of the Domain-Bridge Model.

Rationale. The Domain-Bridge model enables matching application running in a data space by creating a bridge between data writers and data readers in two isolated DDS domains. It ensures that all events occurring in DDS domain "A" (with domain_id "0") should be notified to other participants in remote DDS domain "B" (with domain_id 1). Each DDS participant is described in a global XML-based configuration file, as shown in Listing 2.

Listing 2: Bridge-Federate Configuration as a Domain-Bridge

```
<?xml version="1.0"?>
<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>
```

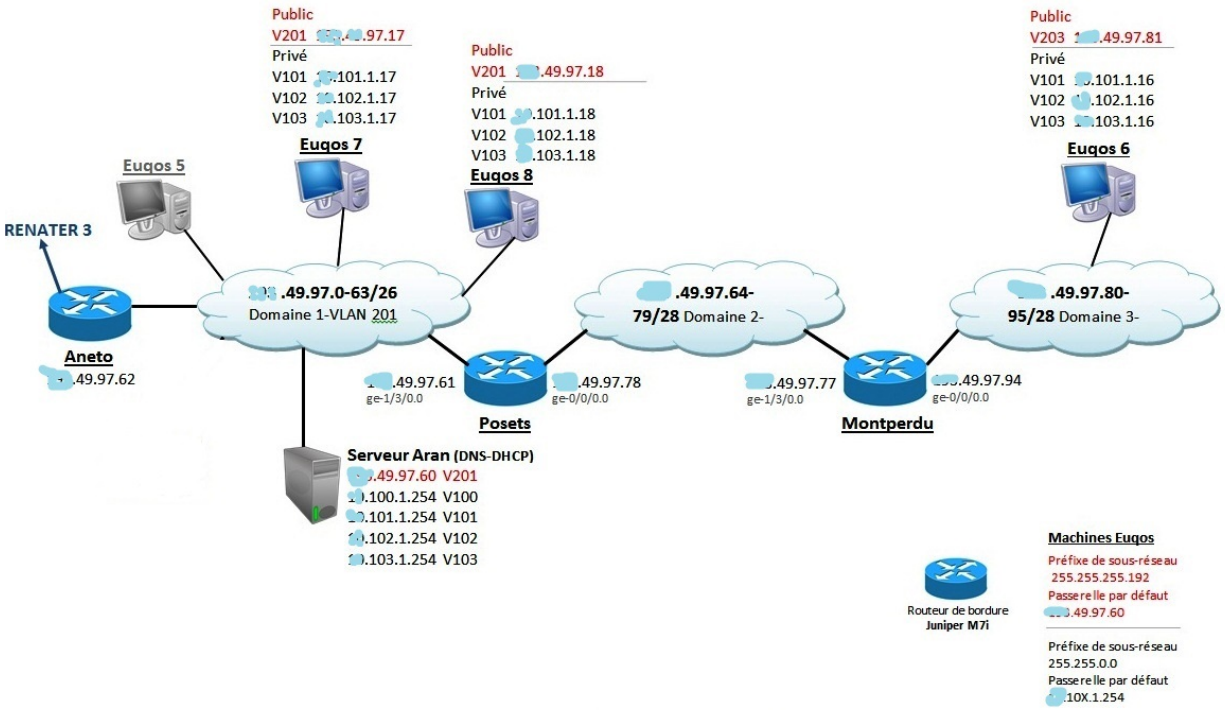


Figure 9: Laasnetexp testbed

```
 </auto_topic_route>
</session>

<session name="Session2">
 <auto_topic_route name="AllBackward">
  <publish_with_original_info>true</publish_with_original_info>

<input participant="2">
<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>
```

The example in Listing 2 includes the topics to transfer during the DDS session and the behavior to follow to route them to different DDS domains.

The Bridge-Federate model creates a virtual network link (called a *global routing domain*) between DDS domain A to DDS domain B to transfer all topics from one data-space to its neighboring domains. It also allows a bi-directional connection between several DDS sessions. These DDS sessions contain the following information:

- Session 1 includes the input domain (with domain_id "0") as "Input", comprising all participants generating topics, and the output domain (with domain_id "1") as "Output", to which topics will be redirected and

- Session 2 defines the reverse operations, *e.g.*, the necessary information to transfer topics from domain "B" (with domain_id "1") to domain "A" (with domain_id "0").

Analysis This test considers a publisher sending DDS topics (with a key "UserID" and a "Content-type" sequence of bytes) to remote subscribers as shown in Figure 10. The experiments described in

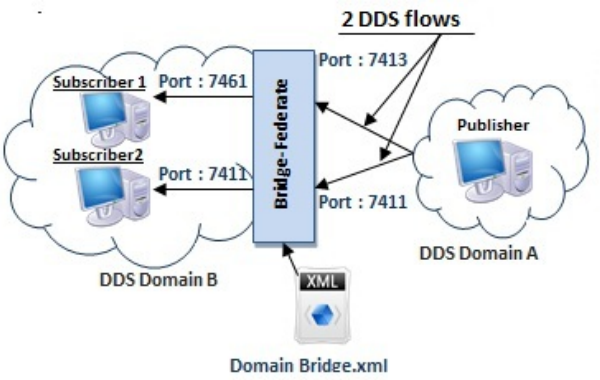


Figure 10: Domain-Bridge Deployment Between Disjoint DDS Domains

Figure 11 showed that all topic instances sent by a publisher in DDS domain A) are forwarded to every subscriber on DDS domain B). As we increased the number of subscribers in the DDS domain B we found that the traffic transmitted between distributed nodes is proportional to the number of remote subscribers. When the number of subscribers increases, the stream sent by the publisher is also duplicated proportionally to the number of subscribers identified by the DDS Discovery service. For N subscriber nodes, therefore, the traffic exchanged is multiplied by N.

Although the Domain-Bridge allows matching distinct DDS domains ("A" and "B"), topics are duplicated by the Bridge-Federate, which is not mandatory. The configuration of the Bridge-Federate in the Domain-Bridge model added a new problem of flow duplication. This solution is considered partial since it routes the traffic between two different IP network domains, but does not optimize bandwidth utilization.

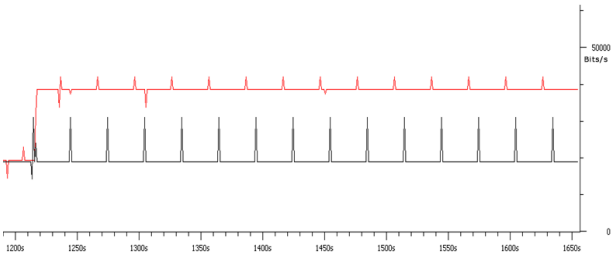


Figure 11: Traffic Exchanged Using the Domain-Bridge

Evaluation of the Topic-Bridge.

Rationale. In addition the Domain-Bridge described above, the Bridge-Federate model can be used as a Topic-Bridge to perform data transformation between distinct DDS domains, transforming topics in the DDS domain "A" (with domain_id "0") by publishing them to another topics having the same data content to the DDS domain "B" (with domain_id "1"). The Listing 3 depicts how the Topic-Bridge performs this operation.

Listing 3: Bridge-Federate Configuration as a Topic-Bridge

```
<?xml version="1.0"?>
 <domain_route name="DomainRoute" enabled="true">
 <participant_1>
   <domain_id>0</domain_id>
 </participant_1>

 <participant_2>
   <domain_id>1</domain_id>
 </participant_2>

<session name="Session" enabled="true">
  <auto_topic_route name="SquaresToCircles">
  <!-- Reading data from participant_1 -->

<input participant="1">
 <registered_type_name>ShapeType</registered_type_name>

 <!-- Reading topic Square -->
 <topic_name>Square</topic_name>
  </input>

 <output>
 <!-- Writing the same type -->
<registered_type_name>ShapeType</registered_type_name>

    <!-- With the same topic -->
    <topic_name>Square</topic_name>
  </output>

 </auto_topic_route>
</session>
</domain_route>
```

In particular, this figure shows that the Topic-Bridge creates only a single session to take data from

participant 1 (as an input participant), reads the registered topic type-names, and transforms them to participant 2.

To check whether the bandwidth utilization depends on the number of subscribers to these topics, this test consists of publishing topics (*e.g.*, squares in Figure 12) from *participant A* and subscribing to them by DDS subscribers (Participants B and C) in different network domains and different DDS domains. The purpose of this configuration is to perform topic transformation from DDS domain "A" to DDS domain "B" (as shown in Figure 12), thereby transforming topics published on domain "A" to another topics having the same data content on domain "B." Figure 12 also shows how the Topic-Bridge

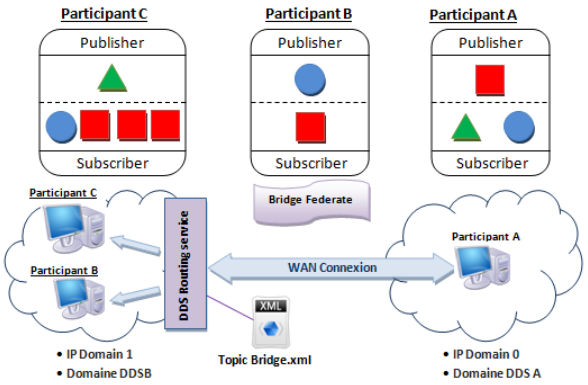
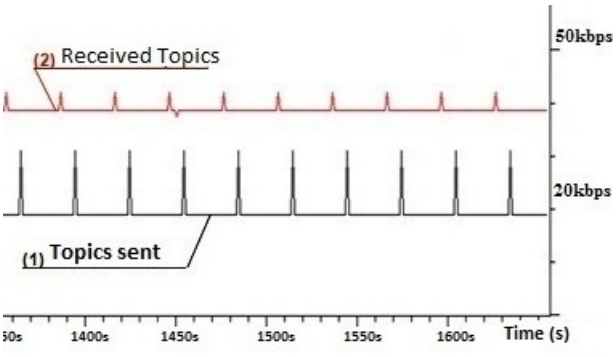


Figure 12: Experiments with the Topic-Bridge Model

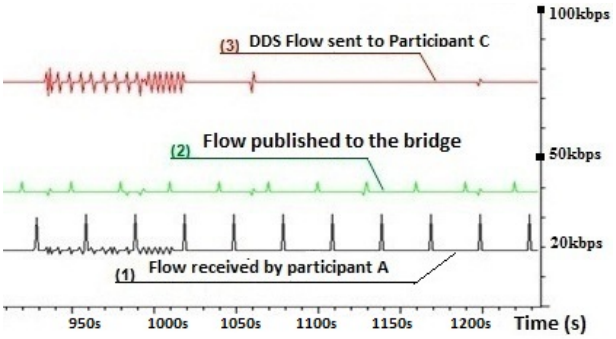
model can regenerate the original topics, even when the number of exchanged topics between different machines is high.

Analysis Figure 13a shows the traffic flow sent by the publisher in participant "A" and received by the remote subscribers (participant "B" and "C"): traffic (1) describes the flow sent by publisher at 20Kbps (distribution of a single topic instance (square)) and traffic (2) shows topic data received by the DDS-RS at 40Kbps, which corresponds to a subscriber registering for two topic instances (circle and triangle).

Figure 13b shows the traffic sent and received by the Topic-Bridge on participant "B". In this sce-



(a) Outgoing Bandwidth Captured on Participant A

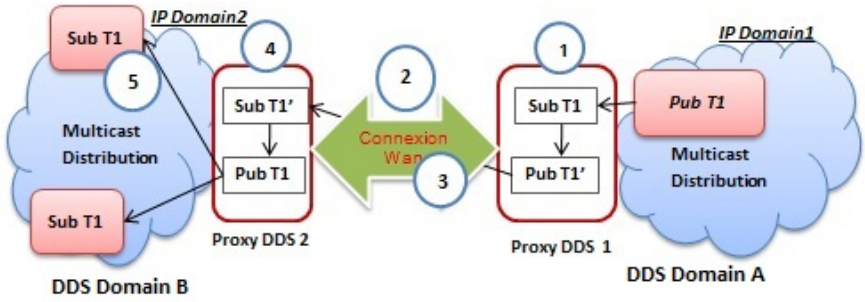


(b) Data Flow with the Topic-Bridge

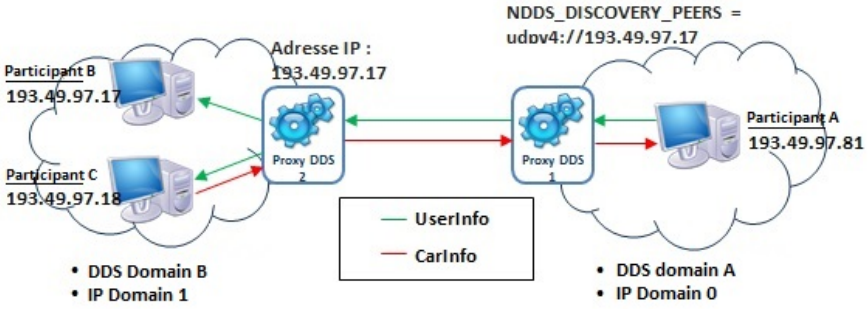
Figure 13: Traffic Exchanged Using the Topic-Bridge

nario, traffic (1) describes topics received by participant "A" which is the same traffic (1) on Figure 13a. Traffic (2) shows topic data published by participant "A" to the bridge. Traffic (3) shows topic data retransmitted by the bridge and received by participant "C" at 80Kbps, which corresponds to the distribution of two and four topics instances. From the expectation of Figure 13a and Figure 13b, the volume of the traffic published by participant "A" on the DDS domain A to subscribers participant "B" and participant "C" on the DDS domain B is independent of the number of subscribers and the number of publishers at run-time. Instead, it depends solely on the topic types being exchanged. The Topic-Bridge model decreased the bandwidth between remote DDS applications in different network domains and different DDS domains.

Despite the absence of data replication, however,



(a) Deployment of Two Proxy DDS Components



(b) Test Scenario Using Proxy DDS Components

Figure 14: Deployment and Test Scenario with Two Proxy DDS

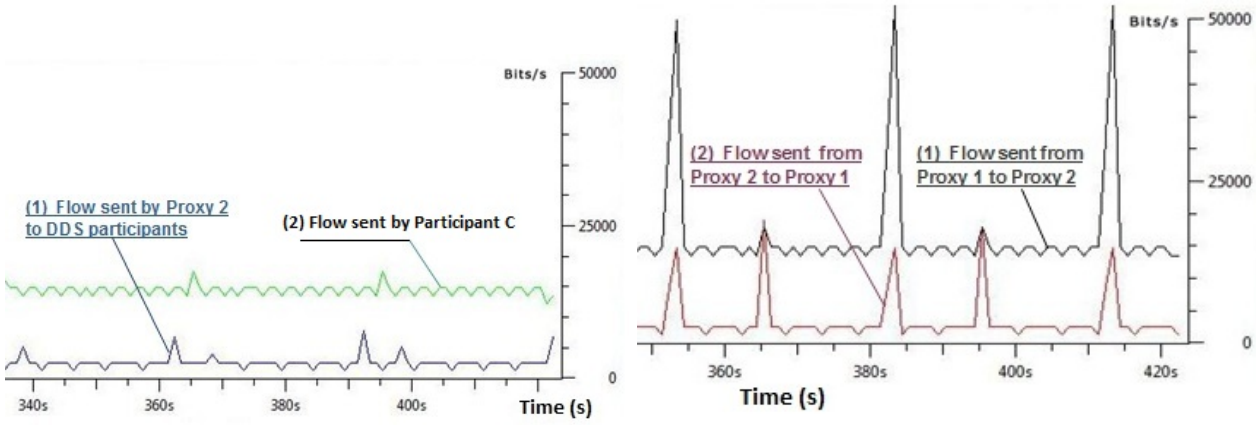
the bandwidth from participant "A" to participant "B" is twice big as when they are in the same DDS domain. This solution therefore does not offer the same performance in the case of a single domain DDS. In particular, the flow exchanged between participants should be optimized to overcome the limits on its flexibility, extensibility and scalability when the number of participant increase, and hence the number of flows exchanged increase accordingly.

4.2.2. Evaluation of the Proxy DDS

The goal of these experiments is to evaluate Proxy DDS against the Bridge-Federate model in terms of optimizing bandwidth and meeting application communication requirements in inter-DDS domains.

Implementation and Deployment of the DDS Proxy. We consider the publisher *Pub T1* in IP domain 1 and DDS domain "A" sending topic data T_1 to a subscribers in IP domain 2 in distinct DDS domain "B", and two proxies at the edge of each network, as shown in Figure 14a. The scenario to distribute DDS topics is as follows:

- Step 1: the publisher produces topic instances T_1 and send them to its local neighbors in DDS domain "A" using multicast service.
- Step 2: The Proxy DDS_1 subscribes to this topic T_1 and consumes all its samples (in DDS domain "A").
- Step 3: The Proxy DDS_1 stores the topics coming from "A" in a FIFO queue, redistribute T_1 topic by transforming it into another topic T'_1 (only the name changed). This transformation avoids the pub/sub loop to the same topic by Proxy DDS_1 (avoid the subscription to **T1**).



(a) Throughput Between Each Proxy and Its Local Neighbors

(b) Throughput Between Two Proxies

Figure 15: Evaluation of the Proxy DDS

- Step 4: The Proxy DDS_2 subscribes to topic T'_1 using unicast service once it is presented on its DDS domain "B," transforms T'_1 instances to T_1 instances, and sends all topic to all participants using multicast.
- Step 5: Subscribes in DDS domain "B" receive all samples of topic T_1 published by their proxy.

The deployment of these proxies should also ensure the coherence of the DDS QoS profiles among different DDS participants in the same DDS domain (between neighbors) and between proxies in different DDS domains over WAN. These QoS profiles are encoded on XML configuration file, *i.e.*, ensure each proxy is compliant with the requested/offered (RxO) contract described on each QoS-profile. In addition, the queuing mechanism enables the proxies to adapt their publish rate under bandwidth fluctuation and avoid packet loss that can occur in WANs.

For example, if the publish rate of Proxy DDS_1 is much higher than the subscription capacity of the subscriber proxy, the publisher proxy stores topics on its FIFO queue. It increases dynamically the capacity of the datawriter cache (uses the RESOURCE_LIMITS DDS QoS policy to avoid consumption excess and forward them later.

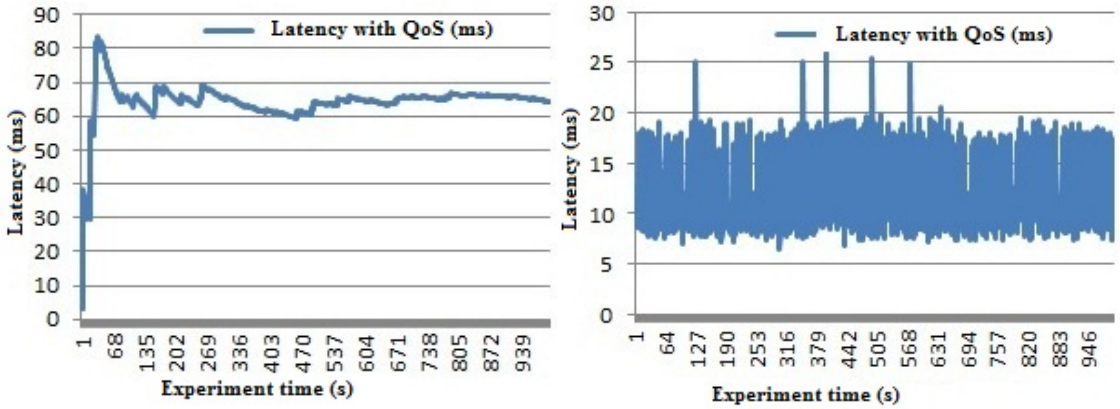
Evaluation of the Proxy DDS.

Rationale. We performed several experiments to evaluate the bandwidth utilization between each proxy DDS and its neighbors subscribers and between remote proxies. This scenario is depicted in Figure 14b. This figure shows the following:

- A publisher (participant A) in DDS domain "A" sends "UserInfo" topic and subscribes to "CarInfo" topic, and many participants that subscribe to the "UserInfo" topic and publish "CarInfo" topic, all in the same DDS domain "B"
- At the edge of each network many proxies deployed to subscribe to topics sent by their correspondent proxies and reflect them to their local DDS participants.

Analysis Figure 15a describes the traffic transmitted between the Proxy DDS_1 and its neighbors in the same DDS domain "A."

Curve (1) shows the throughput for the "UserInfo" topic sent by Proxy DDS_1 and received by the subscriber machine at 15Kbps. Curve (2) describes the throughput of the "CarInfo" topic sent from a subscriber to its neighbor Proxy DDS_1 at 2.5Kbps average bit rate. Figure 15b describes the band-



(a) Without QoS (b) With QoS

Figure 16: Impact of NetQSIP Signaling on Latency

width utilization between Proxies DDS_1 and DDS_2, the traffic data sent from Proxy DDS_2 to its local subscriber, and the flow transmitted from the neighbor subscriber to Proxy DDS_2. Curve (1) shows the traffic exchanged between Proxy DDS_1 and Proxy DDS_2, which corresponds to the distribution of the topics "UserInfo" at 15Kbps. Curve (2) describes the throughput which results from sending "CarInfo" topic data between the remote proxies at 2.5Kbps average.

These results show that the traffic exchanged between proxies does not depend on the number of subscribers, but depends on the topic types (topic size). In comparison with the Bridge-Federate, the Proxy DDS reduced the traffic and thereby optimizes bandwidth utilization.

4.3. Evaluation of the NetQSIP Framework

Below we present the results of experiments conducted to evaluate the performance of the NetQSIP framework described in Section 3.2.3. These results evaluate NetQSIP's latency performance. In particular, we identified two topic flows sent from the publishing application and used them to evaluate the impact of NetQSIP on the time delay by (1) generating real-time traffic containing DDS topic messages that use the DiffServ expedited forwarding per-hop-behavior (PHB) model [5] (whose characteristics of low delay, loss, and jitter are suitable for voice, video, and other real-time services) and (2) perturbing UDP best-effort traffic using the Jperf [58] traffic generator, which is a framework for writing and running automated performance and scalability tests.

4.3.1. Evaluation transmission delay

Rationale. To study the impact of the NetQSIP resource provisioning mechanisms on the moving average delay, we consider two variants of tests as follows:

1. DDS traffic is generated by DDS participant publishing SDP messages, which include the DDS QoS policies. These DDS QoS fields are translated by the proxy into control messages to perform QoS allocation. In this configuration, the NetQSIP resource configurator does not provide any QoS mechanism between the publisher and the subscriber. The default configuration therefore uses the Best-Effort (BE) service and the DDS traffic goes through the best-effort queues of the edge routers.
2. This variant of the experiments uses the same QoS mechanism configurations in all components, as described in Section 3.2.3. NetQSIP

configures the DDS application with the DiffServ Expected Forward (EF) network QoS class, configures the edge routers queues to support 40% Best-Effort (BE) traffic, 30% EF traffic and 20% DiffServ Assured Forward (AF) traffic, and 5% for network control packets.

Analysis. Figure 16 depicts the moving average delay between the publisher and subscriber.

The results in this figure confirm that the average delay experienced by the NetQSIP QoS management mechanism (Figure 16b) performs better results than the application without using NetQSIP QoS allocator (Figure 16a). The average delay remains 15ms when using NetQSIP QoS mechanisms, but without any QoS consideration in the network the average delay remains 4 times higher (∼60ms).

To further validate these results, we considered the dispersion and variability of the delay obtained from traces (Figure 17). The minimum value of the latency is ∼13ms and its maximum value is ∼16ms. These results confirm the potential of NetQSIP to perform network resource reservation. In addition, Figure 18

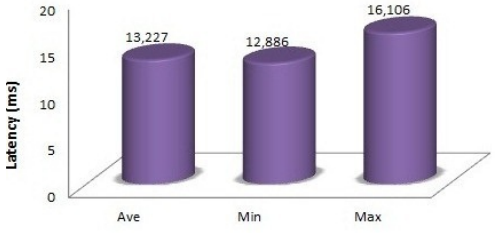


Figure 17: Statistical Distribution of the Moving Average Delay

depicts the results of experiments on the competing UDP background traffic injected from Jperf traffic generator.

This figure shows the packet delays (ms) and the packet lost experienced by the competing UDP flow. The delay experienced for the perturbing UDP flow (curve in Figure 18a) increases dramatically up to ∼500 ms due to the impact of the prioritization of the DDS traffic experienced by NetQSIP. The packet loss rate (Figure 18b) reaches 20% for the UDP best-effort traffic, which has DSCP value 0. UDP packets are dropped because the buffers of the router queues are flooded by the DDS traffic that should be processed in priority.

4.3.2. Evaluation Session Establishment Delay

Rationale. The parameter that describes the effectiveness of the signaling system is the *setup-release latency*, which is represented by the average (a similar parameter describes the effectiveness of the signaling system in telephony networks). DDS session establishment can only be achieved if the proxy sends the 200OK message to the publisher and the resource allocation is achieved. This scenario involves a set of messages that should be negotiated between the caller participant and the called during the session establishment, as described in Section 3.2.3.

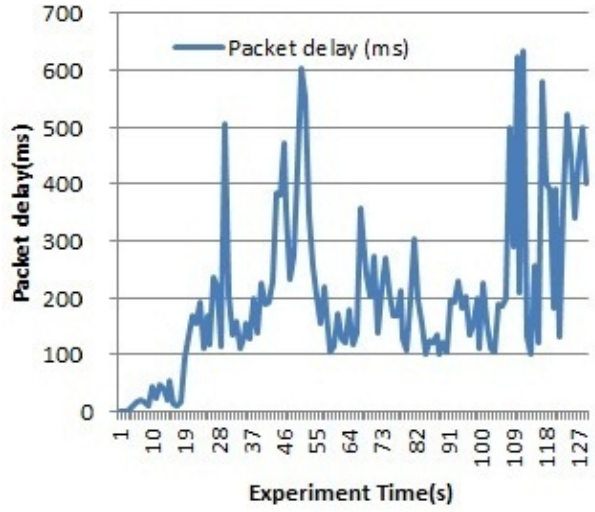
To evaluate session establishment delay, we analyzed the performance of the signaling system, *i.e.*, those components that introduce significant delay to setup/release latencies. This scenario includes all messages exchanged by DDS participants (*i.e.*, Start DDS session, Invite, 183 Progress, Prack, 200OK Prack, 200OK, and Ack) and the messages (*i.e.*, Decision, Report, Reserve, and Response) exchanged between the Proxy DDS, the resource allocator, resource configurator.

The setup procedure starts when a publisher sends an Invite message and is finished on receiving ACK by the subscriber. The 200OK message that reaches the remote subscriber, however, already informs it about successful connection establishment. We therefore define the setup latency as the time elapsed between sending Invite message and receiving 200OK message.

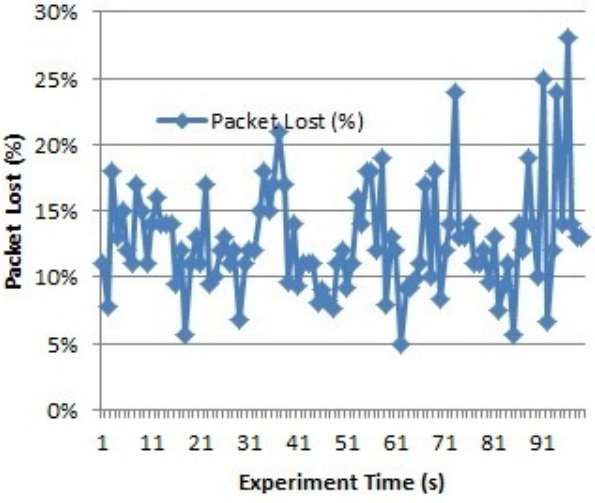
Analysis. Figure 19 shows both the overall setup latency and time delay for each individual SIP message.

The cumulative time latency shown in this figure quantifies the potential impact of the addition of QoS control mechanisms in session establishment. The results in Figure 19 also evaluate the potential of NetQSIP to support the end-to-end DDS QoS policies in WAN-based enterprise DRE systems.

The cumulative transfer delay of 200ms is sufficient to establish real-time communication between



(a) Packet Delay



(b) Packet Lost

Figure 18: Impact of DDS QoS Mechanisms on the concurrent UDP Flow

end-points. The setup latency for each transaction is around 30ms, which is the average of the acceptable setup latency for end-to-end connection establishment. The time delay achieved by the INVITE and the 200OK messages is 23.8ms and 21.42ms, respectively.

These results underscore another benefit of NetQSIP: additional network processing delays are not incurred since messages are not intercepted by the NetQSIP resource configurator at both edge networks. The 183 SESSION IN PROGRESS message, however, had a moving average delay of ∼46.5ms because this message incurs additional delay when the message should pass through the resource configurator. In addition to the data transfer time delay, the time delay for session establishment is added by the implementation of QoS, so the total time reaches values ∼205ms.

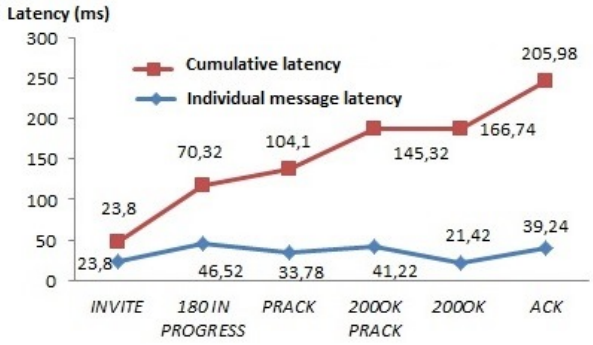


Figure 19: End-to-End Session Establishment Latency

4.3.3. Evaluating the QoS Setup Delay

Rationale. The goal of this experiment is to measure the temporal impact incurred by NetQSIP's resource configurator during session establishment. These experiments are based on the COPS-PEP configured in provisioning mode and the COPS-PDP, which are described in Figure 6. The experiment thus aims to evaluate the latency incurred when the QoS is configured in edge networks.

Latency has been measured many times with more than 100 messages exchanged between them. These messages are (1) the REQUEST message (REQ) per-

Message	Delay (ms)
Message REQ (PEP→PDP)	1.83
Message DEC (PDP→PEP)	35.56
Message RPT (PEP→PDP)	37.94
Total	75.33
Min	73.5
Max	78.22

Table 4: Transmission Delay and Processing Time of NetQSIP Messages (ms)

formed by the resource allocator to request QoS reservation to the resource configurator, (2) the DECISION message (DEC) sent from the QoS configurator to the resource allocator after checking its policy database, to inform it that the session is established with success, and (3) the REPORT message (RPT) transmitted by the resource allocator after the reception of the DEC message, setting the QoS.

Analysis. Table 4 shows the transmission delay and processing time of NetQSIP messages. The REQ message requires less than 2ms to reach the resource configurator because it does not involve any specific process on it. The DEC message requires 35.56ms to be received by the PEP. The resource configurator starts processing REQ message, refers to its internal policy database to select the appropriate elements for triggering DEC message, and then triggers all the decision process to the resource allocator.

The resource allocator receives the DEC message, analyzes the decision being received to implement the QoS policies required by the media sessions, and send the RPT report message to the configurator. This process incurs an average delay of ∼38ms. The overall moving average delay for the QoS negotiation on the control plane is thus ∼75ms, which is accepted for SIP-based communication as an additional delay to the session setup.

5. Concluding Remarks

Enterprise *distributed real-time and embedded* (DRE) systems often span *wide area networks* (WANs). *Quality-of-service* (QoS)-enabled publish/-subscribe (pub/sub) communications, such as the topic-based model supported by the OMG *Data Distribution Service* (DDS) standard, is a hallmark of enterprise DRE systems. Realizing WAN-based enterprise DRE systems is hard for a variety of reasons, including

- Lack of connectivity among isolated DDS domains that may be geographically distributed across the WAN,
- Lack of IP multicast at the network level that hinders discovery of publishers and subscribers at the WAN-scale, and
- Lack of capabilities to enable DDS QoS policies end-to-end over WANs.

To overcome these challenges, this paper presents the design and evaluation of Proxy DDS and NetQSIP, which are middleware that provides network QoS signaling for DDS-based applications in enterprise DRE systems running in WANs. NetQSIP helps configure the underlying WAN transparently on behalf of applications by integrating Proxy DDS components with SIP. This integration enables Proxy DDS components to connect isolated data-spaces, optimize the network resources, and automate the QoS management in QoS-enabled IP networks. The results from experiments we conducted quantify the impact of Proxy DDS and NetQSIP to deliver assured QoS to enterprise DRE systems implemented using DDS and running over WANs.

We learned following lessons from developing and evaluating our Proxy DDS and NetQSIP framework:

• **Autonomic/Self adapting mechanisms are needed to automate DDS QoS configurations.** This paper describes the design and implementation of NetQSIP, which is middleware framework that allows network QoS signaling for DDS-based DRE-applications. Experiments show the potential of NetQSIP to deliver guaranteed and certifiable QoS over the Internet. For certain type of DRE systems, however, NetQSIP's strategy for allocating network resources may be too limiting. In particular, IP-based cyber-physical systems present a challenging

environment for network and service management, which requires a new approach.

We are therefore extending NetQSIP to support adaptive content delivery using ontologies for heterogeneous mobile systems [13]. For example, to provide a QoS-based service selection, DDS applications will require tools to decide which network transport policy and which DiffServ service (AF, EF, etc.) fits its requirements best. The use of ontologies makes it possible to detect the *Service Level Agreement* (SLA) to perform matching between DDS application needs and the network service(s) offered at runtime. This tool will implemented as an adjacent layer to DDS middleware to detect the real behavior of the network service.

• **Reliable discovery protocol is needed to enhance the reliability between remote DDS participants.** This paper described the integration of Proxy DDS and SIP for request session establishment and installing DDS QoS policies at edge routers. The communication between the DDS participants is based on the unreliable UDP protocol for and installing DDS QoS policies at edge routers. The communication between the DDS end-points is based on the unreliable UDP protocol for compliance with existing SIP stacks. For reliable communication, applications perform the sending acknowledgment (ACK-NAK) hop-by-hop between proxies. This situation is not always preferred as it requires publishers to receive ACK from remote subscribers. Addressing this challenge effectively requires end-to-end reliable mechanisms for managing the acknowledgment between data writers and data readers on end-systems. Our future work will align this requirement with ongoing work to enhance discovery protocol in the DDSI specification and associated scalability issues on DDS-based communication.

• **Supporting additional dimensions of QoS require refinements.** The paper addresses the end-to-end timeliness issues in large-scale networks, which relies on information sharing between different participants among shared data-spaces. Since DDS is deployed in mission-critical and enterprise real-time systems for its ultra-low latency benefits, the framework should support high information assurance requirements without overloading the overall system. A minimum and efficient use of cryptography is therefore required to enhance the integrity of the information dissemination. In particular, security policies to control and restrict access to information to only the authorized recipients should be included to the framework to prevent denial of service attacks and ensure the non-repudiation of information.

Our future work will focus on developing security policies to allow authentication, authorization, access control, and secure transport. These capabilities can be done with the help of the *Security Assertion Markup Language* (SAML) to allow exchanging public/private keys as part of the DDS QoS policies, for example within a confidential DDS topics that may be transported using *Datagram Transport Layer Security* (DTLS) protocol. We are also exploring techniques for assuring the reliability of the event dissemination and tolerance to failures.

• **Mechanisms are needed to refactor the network at the control plane.** The results presented in this paper show the importance of integrating the QoS policies into the proxy to allow end-to-end resource allocation in the edge routers. NetQSIP allows QoS provisioning for DDS applications using the existing QoS routing protocols, but does not address how those protocols communicate with the forwarding plane to deliver DDS Topics end-to-end. In particular, it does not allow re-engineering the DDS traffic to test out new protocols in existing networks.

Our future work focuses on adding software-defined networking technology, which separates the network control plane from the data plane so the two can evolve independent of each other. Specifically, we plan to use a software-defined networking technology called OpenFlow [39], which is designed to abstract the network state, to orchestrate the network elements and the network topology, and perform functions including QoS support, traffic monitoring, security management, and flexible diagnostics under networking constraints (such as routing costs). We will use an OpenFlow-based controller to enable scalable DDS applications that can react faster and thus cope with higher update rates to ensure consistency, durability, and scalability while maintaining a logical view of the network.

References

[1] K. Almeroth. The evolution of multicast: From the mbone to inter-domain multicast to Internet2 deployment. *IEEE Network*, 14, 2000.

[2] K. An, S. Pradhan, F. Caglar, and A. Gokhale. A publish/subscribe middleware for dependable and real-time resource monitoring in the cloud. In *Proceedings of the Workshop on Secure and Dependable Middleware for Cloud Monitoring and Management*, SDMCMM '12, 2012.

[3] L. Andersson and T. Madsen. Provider Provisioned Virtual Private Network (VPN) Terminology. RFC 4026 (Informational), Mar. 2005.

[4] Antonio, C. and Luca, F. A DDS-compliant P2P infrastructure for reliable and QoS-enabled data dissemination. 2009.

[5] B. Davie and A. Charny and J.C.R. Bennet and K. Benson and J.Y. Le Boudec and W. Courtney and S. Davari and V. Firoiu and D. Stiliadis. An Expedited Forwarding PHB (Per-Hop Behavior), 2002.

[6] J. Balasubramanian, S. Tambe, B. Dasarathy, S. Gadgil, F. Porter, A. S. Gokhale, and D. C. Schmidt. NetQoPE: A Model-Driven Network QoS Provisioning Engine for Distributed Real-time and Embedded Systems. pages 113–122, 2008.

[7] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Service. *IETF RFC 2475*, 1998.

[8] P. Boonma and J. Suzuki. *TinyDDS: An Interoperable and Configurable Publish/Subscribe Middleware for Wireless Sensor Networks*, chapter 9, pages 206–231. IGI Global, June 2010.

[9] G. Camarillo and P. Kyzivat. Update to the Session Initiation Protocol (SIP) Preconditions Framework. *RFC 4032*, 2005.

[10] G. Camarillo, B. Marshall, and J. Rosenberg. Integration of Resource Management and SIP. *IETF Internet Draft*, 2002.

[11] G. Camarillo, W. Marshall, and J. Rosenberg. Integration of Resource Management and Session Initiation Protocol (SIP). *RFC 3312*, 2002.

[12] B. Cao, J. Yin, S. Deng, Y. Xu, Y. Xiao, and Z. Wu. A highly efficient cloud-based architecture for large-scale stb event processing: industry article. In *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems*, DEBS '12, 2012.

[13] R. Carlos, L. S. Rito, Álvarez Sabucedo Luis M., and C. Paulo. An ontology for managing network services quality. *Expert Syst. Appl.*, 39(9), July 2012.

[14] S.-Y. Chae, S. Ahn, K. Kang, J. Kim, S. Lee, and W.-t. Kim. Fast discovery scheme using dht-like overlay network for a large-scale dds. In *Control and Automation, and Energy System Engineering*, volume 256. 2011.

[15] Chen, Jaime and DíAz, Manuel and Rubio, Bartolomé and Troya, José M. PS-QUASAR: A publish/subscribe QoS aware middleware for Wireless Sensor and Actor Networks. *J. Syst. Softw.*, 86(6), June 2013.

[16] M. Cinque, C. Di Martino, and C. Esposito. On data dissemination for large-scale complex critical infrastructures. *Computer Networks*, 2011.

[17] CORBA-OMG. Common Object Request Broker Architecture (CORBA/IIOP). *Object Management Group, Inc.*

[18] B. Dasarathy, S. Gadgil, R. Vaidyanathan, A. Neidhardt, B. Coan, K. Parmeswaran, A. McIntosh, and F. Porter. Adaptive network QoS in layer-3/layer-2 networks as a middleware service for mission-critical applications. *JSS*, 80, 2007.

[19] B. Dasarathy, S. Gadgil, R. Vaidyanathan, K. Parmeswaran, B. Coan, M. Conarty, and V. Bhanot. Network QoS Assurance in a Multi-Layer Adaptive Resource Management Scheme for Mission-Critical Applications using

the CORBA Middleware Framework. *IEEE. RTAS*, 2005.

[20] S. Deering. Host extensions for IP multicasting. RFC 1112 (Standard), 1989.

[21] S. Deering and D. Cheriton. Host groups: A multicast extension to the Internet Protocol. *RFC 966*, 1985.

[22] Dianes, J. A. and Diaz, M. and Rubio, B. Using standards to integrate soft real-time components into dynamic distributed architectures. *Comput. Stand. Interfaces*, pages 238–262, 2012.

[23] S. R. E., L. J. P., R. Craig, S. D. C., K. Yamuna, and I. Pyarali. Flexible and adaptive qos control for distributed real-time and embedded middleware. In *Proceedings of the ACM/IFIP/USENIX 2003 International Conference on Middleware*, Middleware '03, pages 374–393, 2003.

[24] M. Femminella, R. Francescangeli, F. Giacinti, E. Maccherani, A. Parisi, and G. Reali. Scalability and performance evaluation of a jain slee-based platform for voip services. In *Teletraffic Congress, 2009. ITC 21 2009. 21st International*, pages 1–8, 2009.

[25] M. Femminella, R. Francescangeli, E. Maccherani, and L. Monacelli. Implementation and performance analysis of advanced it services based on open source jain slee. In *Local Computer Networks (LCN), 2011 IEEE 36th Conference on*, pages 1–8, 2011.

[26] M. Ge, S. V. Krishnamurthy, and M. Faloutsos. Application versus network layer multicasting in ad hoc networks: the alma routing protocol. *Ad Hoc Netw.*, 4(2), 2006.

[27] Gerardo Pardo-Castellote. OMG Data Distribution Service: Architectural Overview. *ICDCSW*, 2003.

[28] Gill, C. D. and Gossett, J. M. and Corman, D. and Loyall, J. P. and Schantz, R. E. and Atighetchi, M. and Schmidt, D. C. Integrated Adaptive QoS Management in Middleware: A Case Study. *Real-Time Syst.*, 29:101–130, 2005.

[29] P. Grace, D. Hughes, B. Porter, G. S. Blair, G. Coulson, and F. Taiani. Experiences with open overlays: a middleware approach to network heterogeneity. *SIGOPS Oper. Syst. Rev.*, 42, 2008.

[30] A. Hakiri, P. Berthou, A. Gokhale, D. Schmidt, and T. Gayraud. Supporting End-to-end Scalability and Real-time Event Dissemination in the OMG Data Distribution Service over Wide Area Networks. *Submitted to Elsevier Journal of Systems Software (JSS)*, May 2013.

[31] M. Handley, V. Jacobson, and C. Perkins. SDP: Session Description Protocol. RFC 4566 (Proposed Standard), July 2006.

[32] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg. SIP: Session Initiation Protocol. *RFC 2543*, 1999.

[33] M. Hosseini, D. Ahmed, S. Shirmohammadi, and N. D. Georganas. A survey of application-layer multicast protocols. *Communications Surveys Tutorials, IEEE*, 9, 2007.

[34] G. Hunt. DDS Use Cases: Effective Application of DDS Patterns and QoS. In *OMG's Workshop on Distributed Object Computing for Real-time and Embedded Systems*, Washington, D.C., July 2006. Object Management Group.

[35] X. Jin, K.-L. Cheng, and S.-H. G. Chan. Island multicast: combining ip multicast with overlay data distribution. *IEEE Transactions on Multimedia*, 11(5), 2009.

[36] W. Kang, K. Kapitanova, and S. H. Son. Rdds: A real-time data distribution service for cyber-physical systems. *Industrial Informatics, IEEE Transactions on*, 8, 2012.

[37] J. M. Lopez-Vega, J. Povedano-Molina, G. Pardo-Castellote, and J. M. Lopez-Soler. A content-aware bridging service for publish/-subscribe environments. *J. Syst. Softw.*, 86, 2013.

[38] Lu, Xinjie and Yang, Tian and Liao, Zaifei and Li, Xin and Wang, Yongyan and Liu, Wei and Wang, Hongan. A Novel QoS-Enable Real-Time Publish-Subscribe Service. *ISPA '08*, 2008.

[39] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, and J. Rexford. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, pages 69–74, 2008.

[40] Mingozzi et al. EuQoS: End-to-End Quality of Service over Heterogeneous Networks. *IFIP, Comput. Commun.*, 2009.

[41] K. Muthukrishnan and A. Malis. A Core MPLS IP VPN Architecture. *RFC 2917*, 2000.

[42] OASIS. Web Services Brokered Notification Version 1.3. *http://www.oasis-open.org/*, 2006.

[43] Object Management Group. The real-time publish-subscribe wire protocol dds interoperability wire protocol (ddsi), 2009.

[44] OMG-DDS. Data Distribution Service for Real-Time Systems Specification. *Object Management Group, Inc*, 2007.

[45] OMG Specification. IDL Language Mapping Specifications. OMG specification, 2013.

[46] P. Owezarski, P. Berthou, Y. Labit, and D. Gauchard. LaasNetExp: a generic polymorphic platform for network emulation and experiments. *4th International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities*, Mar. 2008.

[47] Y. Park, D. Chung, D. Min, and E. Choi. Middleware integration of dds and esb for interconnection between real-time embedded and enterprise systems. In G. Lee, D. Howard, and D. lzak, editors, *Convergence and Hybrid Information Technology*, volume 206. Springer Berlin Heidelberg, 2011.

[48] P. R. Pietzuch and J. Bacon. Peer-to-peer overlay broker networks in an event-based middleware. In *Proceedings of the 2nd international workshop on Distributed event-based systems*, DEBS '03, 2003.

[49] J. Povedano-Molina, J. M. Lopez-Vega, J. Sánchez-Monedero, and J. M. Lopez-Soler. Instant Messaging Based Interface for Data Distribution Service. *XIII Jornadas de Tiempo Real*, 2010.

[50] E. Rosen and Y. Rekhter. BGP/MPLS IP Virtual Private Networks (VPNs), 2006.

[51] J. Rosenberg and H. Schulzrinne. An Offer/Answer Model with Session Description Protocol (SDP). *RFC 3264*, 2002.

[52] J. Rosenberg and H. Schulzrinne. Reliability of Provisional Responses in Session Initiation Protocol (SIP). *RFC 3262*, 2002.

[53] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261 (Proposed Standard), 2002.

[54] A. I. T. Rowstron, A. M. Kermarrec, M. Castro, and P. Druschel. SCRIBE: The Design of a Large-Scale Event Notification Infrastructure. pages 30–43, 2001.

[55] RTI Innovation. RTI Routing Service for DDS, 2010.

[56] S. Salsano. COPS Usage for Diffserv Resource Allocation (COPS-DRA). *IETF Draft*, 2001.

[57] Sebastian Marius Rosu and George Dragoi. *VPN Solutions and Network Monitoring to Support Virtual Teams Work in Virtual Enterprises*, volume 8. 2011.

[58] T. Brethour and K. Gibbs. Jperf version 1.0 The Iperf Front End, 2003.

[59] L. Veltri, S. Salsano, and D. Papalilo. Sip extensions for qos support in diffserv networks. *IETF Draft, draft-veltri-sip-qsip-00*, October 2000.

[60] Y. Vigfusson, H. Abu-Libdeh, M. Balakrishnan, K. Birman, R. Burgess, G. Chockler, H. Li, and Y. Tock. Dr. multicast: Rx for data center communication scalability. In *Proceedings of the 5th European conference on Computer systems*, EuroSys '10, 2010.

[61] N. Wang, D. C. Schmidt, A. Gokhale, C. Rodrigues, B. Natarajan, J. P. Loyall, R. E. Schantz, and C. D. Gill. QoS-enabled Middleware. In Q. Mahmoud, editor, *Middleware for Communications*, pages 131–162. Wiley and Sons, New York, 2004.

[62] Y.-H. Wang, S.-H. Yang, A. Grigg, and J. Johnson. A dds based framework for remote integration over the internet. *7th Annual Conference on Systems Engineering Research (CSER 2009)*, April 2009.

[63] C. Zhang, Sadjadi, S. Masoud, S. Weixiang, R. Raju, and D. Yi;. A user-centric network communication broker for multimedia collaborative computing. *IEEE, CollaborateCom*, 2006.

[64] L. Zhou, L. H. Ngoh, X. Shao, T. Chai, T. K. Lee, and J. Teo. Ims service plane enabled heterogeneous networks for multimedia applications. In *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*, 2010.