iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 9 **Issue:** I **Month of publication:** January 2021

DOI: https://doi.org/10.22214/ijraset.2021.29243

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue I Jan 2021- Available at www.ijraset.com

Criteria Set for Evaluation of different DDS Distributions

Aditya Baunthiyal[1], Rakesh[2], Manjeet Gupta[3], Amit Kr. Jain[4]
[1, 2, 3, 4]*Central Research Laboratory, Bharat Electronics Limited, Ghaziabad (India)*

Abstract: ***Data Distribution Service (DDS) is the open standard for messaging using a publisher-subscriber model from Object Management Group(OMG). DDS fosters the development of loosely coupled, modular and open architecture systems. By supporting well-defined interfaces between components and subsystems, DDS eradicates stove-pipe, closed and proprietary architectures. This eliminates complexity to reduce integration, maintenance and upgrade costs; promotes competition at the subsystem and middleware levels and eases reuse. Multiple implementations of DDS are available, ranging from high-end commercial on the shelf products to open-source community-supported projects. Every implementation claims to fit in the standard and provides the best possible parameters for data communication but very few studies on performance analysis, usability and features exists. The current paper bridges this gap and facilitates middleware selection by proposing a criteria set to evaluate and compare different DDS Distributions(RTI Connext, Vortex Open Splice, OpenDDS , CoreDx and Eprosima Fast RTPS).***

Keyterms: DDS, performance analysis, QOS, OpenDDS, RTPS, CoreDx DDS, Vortex Open Splice, Eprosima Fast RTPS, DCPS, CORBA

I. INTRODUCTION

Traditionally, a socket based network library is used in distributed software for communication which is based either on request-response or client-server model [1] This approach is inefficient for mission-critical real time distributed systems, industrial automation etc. where spatial and temporal uncertainty increases exponentially by increasing number of additional computing nodes. In such systems publish-subscribe communication pattern seems more practical and this require QoS-enabled data-centric middleware platforms capable of providing distributed applications with the needed information (often high-volume data) in a timely manner (often with high data-rate). OMG has come up with Data Distribution Service (DDS) specification which satisfies aforementioned requirements of any middleware [1]. It describes a Data-Centric Publish-Subscribe (DCPS) model for distributed application communication and integration. All the DDS distributions implements DCPS model and complies with the specification which makes them equally efficient. System development requires selection between different implementations of OMG –DDS specification. Existing literature and market survey concludes that many researchers have evaluated DDS distributions. Previous works in this area for example [2] and [3] compares only the popular distributions such as OpenSplice, RTI DDS and OpenDDS from latency point of view. Major drawback of [2] being it was tested for small message sizes up to 200 bytes. Research [3] analyzes DDS under different publication rates and number of topics but other critical features and performance parameters were not considered. The present work in this paper not only evaluates different DDS implementations from performance point of view but also considers other critical features such as architecture(federated, non- federated), Interoperability, Persistence, routing, monitoring tool, license, community support, latency , throughput, Low-Bandwidth support, language and Operating system.

In this paper Section II gives the necessary background and definitions about DDS in general. Section III enlists the criteria set for evaluation of DDS implementations. It further focuses on understanding the DDS implementations based on the enlisted criteria and gives a detailed analysis of latency and throughput collected by experiment conducted in our lab. Finally, conclusive remarks in section IV end the paper.

II. DDS BACKGROUND

The DDS specification defines two separate interfaces. A Data-Centric Publish-Subscribe (DCPS) interface which provides a global data space in which Type-specific data publishers send data that interested subscribers can receive [1]. This is considered a low level interface and very similar to past MOMA (Message Oriented Middleware Architecture) implementations that became prevalent in the 1990's. A Data Local Reconstruction Layer (DLRL) which allows for distribution of an object model. This is an optional layer built on top of DCPS. DLRL is functionally very attractive as it provides a higher level interface and hides much of the pub/sub details behind a local framework. Most of the modern DDS implementations use DCPS [1]. DCPS overview is given in Figure:1.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue I Jan 2021- Available at www.ijraset.com

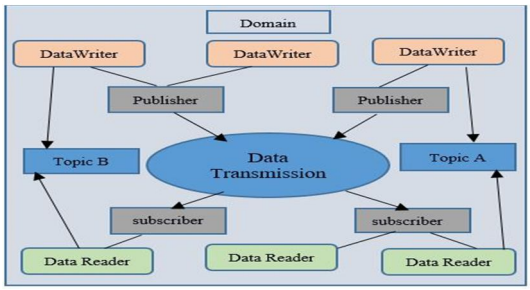


Figure: 1 DCPS Overview

Domain is the fundamental partitioning unit within DCPS.
All DDS entities belonging to one domain can only interact with other entities in that same domain. Topic is the fundamental means of interaction between publishing and subscribing applications. Each topic has a unique name and specific data type associated with it. The data writer is used by the publishing application code to pass values to the DDS. Each data writer is bound to a particular topic. Publisher is responsible for taking the published data and disseminating it to all relevant subscribers in the domain. The subscriber receives the data from the publisher and passes it to any relevant data reader connected to it. The data reader takes data from the subscriber, demarshals it into the appropriate type for that topic and delivers the sample to the application. The DDS specification also defines a number of Quality of Service (QoS) policies that are used by applications to specify their QoS requirements to the service. Participants specify what behavior they require from the service and the service decides how to achieve these behaviors. These policies can be applied to the various DCPS entities (topic, data writer, data reader, publisher, subscriber, domain participant). RTPS (Real Time Publish Subscribe Protocol) is a protocol for best effort and reliable pub-sub communications over unreliable transports such as UDP in both unicast and multicast.

III. EVALUATION OF DDS DISTRIBUTIONS

DDS is a low-latency communication architecture that gives high-performance in real time applications. This is because the DDS and RTPS specifications are themselves written to satisfy the requirements of real-time and near real-time systems [4]. These specifications contain instructions to keep data copies to a minimum within the DDS middleware. They also specify a compact data encoding on the wire, light-weight notification mechanisms, and the ability for the application to specify resources limits, allowing DDS to pre-allocate memory and reduce the number of memory allocations during run-time. All of these characteristics result in DDS implementations that are in general efficient, low-overhead, and high-performance. Evaluation criteria used in current paper to compare different implementations are based on features such as architecture (federated, non- federated), Interoperability, Persistence, routing, monitoring tool, licensing, community support, latency, throughput, Low-Bandwidth support, language and Operating system. All the above features are summarized in the table 1 and detailed analysis of different DDS Implementations is given below: -

A. Architecture & Interoperability

There are two types of architecture in DDS community viz. federated and non-federated. RTI Connext, CoreDx DDS and Eprosima Fast Rtps adopts a non-federated architecture (decentralized) where applications are self-contained and do not require any separate services for middleware management. Latency is usually better in this approach and also the performance bottleneck faced in middleware service daemons is solved. The biggest advantage in this approach is avoiding the single point of failure. The only supported data delivery and discovery protocol is the standard RTPS protocol [4]. Other implementations such as OpenDDS and Vortex OpenSplice supports both federated and non-federated architecture. Federated model allows a clear separation of the applications, that run in a separate user process, from the DCPS configuration and communication related details. The DCPS daemon is a potential communication bottleneck at very high data rates and a single point of failure within each physical host. To overcome such limitations, OpenSplice and OpenDDS supports also the decentralized model, although it is less optimized and not considered the default setting. Moreover, the OpenSplice DCPS implementation supports both an optimized proprietary data delivery protocol, called RTNetworking, suggested as the default one for local LAN deployments, and DDSI protocol for interoperability with other DDS solutions [5]. OpenDDS implements a standalone service called the DCPS Information Repository (DCPSInfoRepo) to achieve the centralized discovery method and peer-to peer discovery with RTPS.

B. Persistence profile

Almost all implementations support persistence with different levels of flexibility. In RTI Connext it is achieved through a separate service called persistence service which can be configured with the help of XML [6]. In OpenDDS persistence is only limited to centralized discovery(DCPSInfoRepo) and there is no persistence for RTPS discovery. In OpenSplice persistence is achieved through Daemon services in federated architecture called durability service [7]. Fast-RTPS comes with built-in persistence plugin which provides persistence on a local file using SQlite3 API. CoreDxDDS also supports persistence [8]. The persistence profile is highly automated in RTI and OpenSplice with minimal configuration changes. Everything is handled externally and there is no change in the applications internal logic.

C. Routing

RTI Connext DDS also provides routing service, an out-of-the-box solution for integrating disparate and geographically dispersed systems. It scales RTI Connext DDS applications across domains, LANs and WANs, including firewall and NAT traversal [9]. OpenSplice also provides routing with the help of Vortex link which provides transparent discovery and routing between data readers and data writers regardless of location. Vortex Link supports a number of different deployment and connectivity scenarios, including individual device to cloud, system to cloud and also connecting your different LAN's to turn them into a single system [10]. In OpenDDS DCPSInfoRepo runs over TCP/IP and RTPS discovery uses UDP/IP multicast. It can be configured to send unicast announcements in order to work on networks (WAN, NAT) that don't route multicast traffic but the process is manual and not automated. Fast RTPS provides eProsima Integration Service for creating parameterized communication bridges between different systems, services and protocols. It is also able to perform transformations over the messages such as customized routing and mapping between input and output attributes or data modification [11]. CoreDxDDS provides routing with the help of multiplexor. Multiplexor can be used to bridge between two DDS Domains. The CoreDX DDS Multiplexor can selectively forward topic data and transform topic QoS policies. This configuration is particularly useful when connecting and exchanging data between systems with different DDS deployment strategies [12].

D. Monitoring Tools Support

It is important to have development tools that are straightforward and easy to use, but still have the power to analyze, diagnose, debug, administer and elegantly report the communications details of your system. For solving this purpose RTI Connext, vortex OpenSplice and CoreDxDDS comes with their proprietary monitoring tool. OpenDDS also provides a monitoring tool but it works only with central discovery with low efficiency. Fast-RTPS does not have any monitoring tool. RTI's Admin Console offers you a centralized tool for monitoring, debugging, and administering your distributed system. It has following key features: -

Table :1 Comparison Table

Distribution	Latency	Throughput	Interoperability	Architecture (Federated/Non-federated)	Persistence Support	Routing Support	Monitoring Tool	Low Bandwidth support	Commercial / Open Source
OpenDDS	363 us for 32 bytes 1114 us for 8192 bytes	780 megabits/sec on 1 Gbps Ethernet	DCPSInfoRepo for central and RTPS transport For peer-to-peer	Both	Yes	No	Yes (only with central discovery)	No	Open Source with commercial support
CoreDX DDS	227 us for 32 bytes 920 us for 8192 bytes	930 megabits/sec on 1 Gbps Ethernet	Real-Time Publish Subscribe (RTPS) protocol	Non-Federated	Yes	yes	Yes (doesn't allow creation of entities on the fly.)	No	commercial

Vortex OpenSplice	200 us for 32 bytes 930 us for 8192 bytes	940 megabits/sec on 1 Gbps Ethernet	DDSI-RTPS used only for interoperability while RT Networking is default	Both	Yes (with RT networking service)	yes	Yes (OSPL Tuner) QOS Match graph not available	yes	commercial
RTI Connext DDS	225 us for 32 bytes 917 us for 8192 bytes	over 940 megabits/sec on 1 Gbps Ethernet	RTPS transport	Non-Federated	Yes(with Routing service)	Yes	Yes (Admin console) doesn't allow creation of entities on the fly.	yes	commercial
eProsima Fast RTPS	280 us for 32 bytes 922 us for 8192 bytes	930MBits/sec	RTPS transport	Non-Federated	Yes (with built in plugin)	yes	No	yes	Open Source with commercial support

It checks to ensure that your distributed system's Quality of Service (QoS) settings are compatible. This analysis happens automatically and in real-time. As QoS changes occur in your system, Admin Console is notified and updates the analysis to reflect the most recent settings. It collects entity counts and summarizes them in one easy-to-read table. You will see the number of hosts, processes, and RTI services participating in your distributed system as well as the Topics, Data Writers, and Data Readers. This table can be used to compare a running system's current state with a previous state to ensure that everything is running that is expected. For RTI services that provide detailed performance statistics, such as Routing Service, Admin Console will display information such as data throughput and latency for each route. Admin Console provides a Distributed Log view to display log messages from any RTI services or processes that use RTI Distributed Logger. You can also customize the verbosity of the log messages through the Log. DDS discovery information can be exported from Admin Console into an XML file. The exported data can then be imported and displayed. This feature could be used to collaborate with project team members. Admin console lacks in creation of publishers / writers on the fly to experiment and validate how this data should be treated by the middleware.

The Vortex OpenSplice Tuner Tool can be used during all phases of software development. It allows injection of test input-data by creating publishers and writers 'on the fly' as well as validating the responses by creating subscribers and readers for any produced topics. During the test phase, the total system can be monitored by inspection of data (by making 'snapshots' of writer and reader history caches) and behaviour of readers and writers (statistics, like how long data has resided in the reader's cache before it was read) as well as monitoring of the data distribution behaviour (memory-usage, transport-latencies). The Vortex OpenSplice Tuner Tool is differentiated from other vendor's DDS based tooling by its dynamic capabilities to not only connect to any remote Vortex OpenSplice DDS based system at runtime, but also its capabilities to create, discover and (QoS) tune any DDS entities 'on-the-fly'. We don't have to create our own testing simulator thus saving development time. It also provides facilities to observe entities in a Vortex OpenSplice system and browse over their (mutual) relationships using different views. It doesn't provide a match graph of DDS entities and QOS mismatch thus hindering integration process.

CoreDx DDS spy Tool provides a number of useful features to help system development, integration, and test activities. The CoreDX DDS Spy provides a clear, intuitive display of the current state of all Entities on the DDS Network. The user can easily see all Domain Participants, Topics, DataReader, and DataWriter. This information can be grouped either by Domain Participants, or by Topics. In both cases, matched and mismatched Readers and Writers are clearly visible and colour coded. This information provides a clear visibility into which Readers and Writers on the system are matched, and if they are not matched, the reason for the miss-matches.

Selecting on one of the DDS Entities will populate another area of the CoreDX DDS Spy Window with detailed information about that Entity, including its Quality of Service (QoS) configuration. This detail provides important information into the configuration of each Entity and can help explain communication behaviour in the DDS Network. Another area of the CoreDX DDS Spy Window displays the high-level log of DDS events on the network, including Entity creation, Entity deletion (or expiration), Entity matching, and Entity non-matching. This chronological log helps developers determine when and why data communications occur (or do not occur). Also available, is a Wireshark-like display of every DDS packet that is captured by CoreDX DDS Spy. This can be useful for developers who have a deeper understanding of the RTPS wire protocol to see specific network traffic produced by DDS applications on the network. Only disadvantage is that it does not allows creation of publishers / writers and subscribers / readers on the fly to experiment.

E. DDS in Low Bandwidth Environment

DDS performance in constrained network environments such as Tactical radio links and satellite links is very poor. It takes very long discovery time and gives very low effective throughput because of the chatty discovery protocol, large size protocol header (56 bytes), non-compressed data and non-optimized QOS [13]. To meet the above challenges RTI DDS and Eprosima Fast-RTPS provides a limited bandwidth plugin which reduces the protocol header size, compresses data and greatly improves performance parameters. OpenDDS and CoreDx provides no support and OpenSplice uses ZLIB compression and static discovery for optimizations over low bandwidth networks.

F. Licensing

RTI Connext, OpenSplice and CoreDx DDS are commercial and OpenDDS and Fast RTPS are open Source with limited commercial support on demand. When deciding between open source DDS and a commercial DDS it is important that you completely understand your requirements, and what features and services you need before selecting which DDS implementation to use. The following are some of the key questions to help define your requirements:

1) Is the language and O.S that you want to use supported?
2) Is integration with scripting languages required?
3) Do you need integration with the cloud? Or with a relational database?
4) Is any monitoring tool available for debugging and diagnosis?
5) Will you need security in the future? Or certification?
6) Is the integration process automated?
7) Is the release robust? Does it go through rigorous testing? Has it been successfully used numerous times in fielded applications?
8) Is I.P mobility required?
9) Is Low bandwidth communication supported?
10) Which architecture(federated/Non-federated) is required?

Looking at above questions we suggest that if you are developing a proof of concept project, language is not a barrier and you only need standard basic features then Open source should be the choice for you. But if you need all language support, database and cloud integration, security, 24*7 support, strong community and costing is not a problem etc. then go for Commercial DDS.

G. Operating System & Language Support

RTI Connext, CoreDX DDS, Eprosima Fast-RTPS supports Linux, Mac OS, Windows, Windows CE, Embedded, Solaris 10, Lynx, VxWorks and android. whereas OpenDDS and Vortex OpenSplice doesn’t supports android. Every implementation supports languages like C, C++, java, C#. OpenDDS and Eprosima Fast-RTPS also supports all languages except C#.

H. Latency & Throughput analysis

To compare the performance (latency and throughput) of different implementations we created a framework called DDS benchmarking Framework. The main concept behind this framework is to put all the DDS implementations under the same testing environment and measure their latency and throughput parameters and present them in the same format so that they can be compared to find out which performs better.

Benchmarking environment consists of RHEL 7.1, Test duration: 200 seconds per test (data point). Test Machines: Intel Core i5 CPU Cache: 6MB , Number of cores: 4, Speed: 3.20 GHz ,Memory: 8GB.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue I Jan 2021- Available at www.ijraset.com

Test Setup for latency

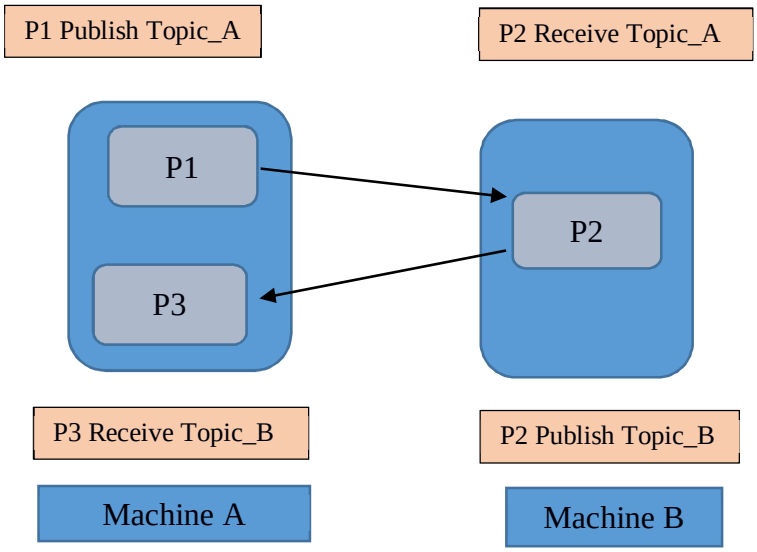


Figure: 2 Latency Test Setup

Process P1: This process runs on machine A and publishes Topic A with a frequency of 1 second with write time stamp. It started with a payload of 32 bytes and repeated up to a maximum payload of 8000 bytes. Every payload was tested for 200 seconds.
Process P2: This process runs on machine B. It receives Topic A and publishes Topic B with same datatype.
Process P3: This process runs on machine A. It receives Topic B and logs the write time and receive time in excel file.
Latency test setup is shown in figure:2. Latency was measured, in microseconds, by having the process _P2 on machine B echo messages back to the Process_P3 on machine A. This allowed round-trip latency to be measured on the sending machine, avoiding time synchronization issues. The round-trip latency was divided in half to get the one-way latency that is shown. The latency numbers include the system and network transport latencies plus the time to copy the entire data sample since the timestamps are from the start of sending to the end of receiving. The test was repeated up to a maximum payload of 8192 bytes for every DDS with using C++ API. This setup can be used for any other DDS implementation which is not mentioned in the paper.

Latency = (Receive_time – write_time)/2
Latency Experiment results

Minimum latency experiment results are shown in Figure 3. It can be seen from the results that for Small payload sizes, from 32 bytes to 1024 bytes Vortex OpenSplice with RT networking outperforms all the other implementations due to batching optimization and also there is very less variation between minimum and median values demonstrating that data delivery is very predictable. Minimum latency ranges from 200 to 930 microseconds.
For payload sizes larger than 1024 bytes RTI Connext is better because it uses peer-to-peer messaging — without a centralized or per-node Enterprise Service Bus (ESB), message broker, server or daemon processes — it does not impose any inherent limit on aggregate messaging capacity.
CoreDx and Eprosima Fast-RTPS also shows similar trends as RTI Connext because of RTPS protocol but RTI Connext is slightly better with minimum latency ranging from 225 microseconds at 32 bytes to 917 microseconds at 8192 bytes. OpenDDS latency performance was average with minimum latency values between 363 and 1114 microseconds. Figure:4 shows the median latency comparisons of the all the implementations.
The data shows that for large payload sizes RTI Connext outperforms other distributions and for small data sizes Vortex OpenSplice is better [14] [15] [16] [17] [18] [19] [20].

Test setup for throughput

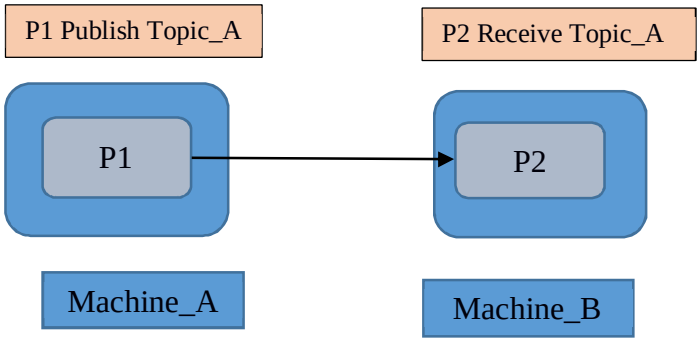


Figure: 5 Throughput setup

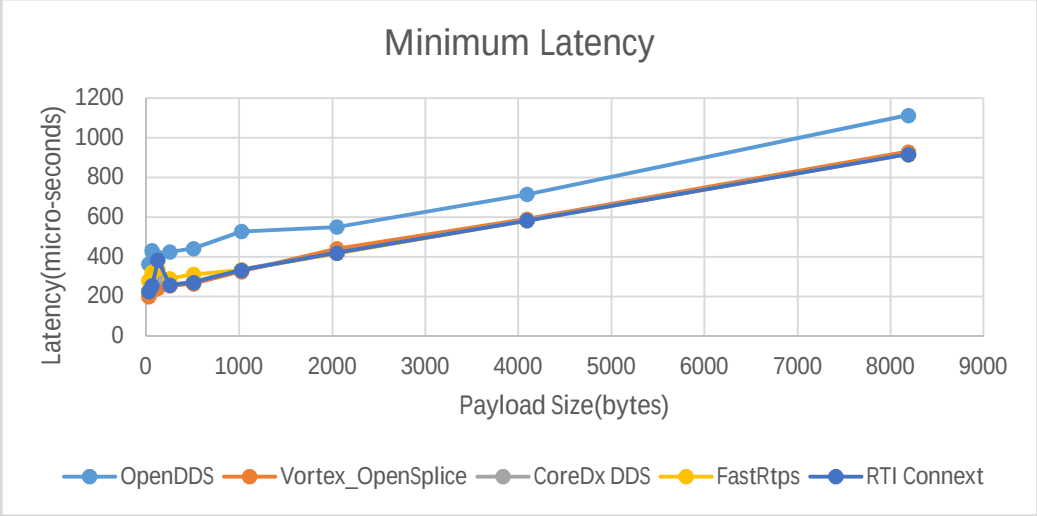


Figure: 3 Minimum Latency

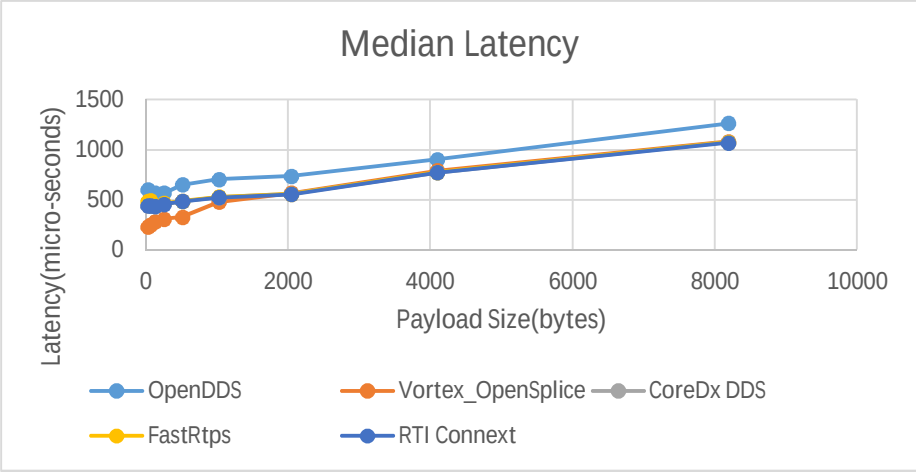


Figure 4: Median Latency

1) *Process- P1:* This process runs on machine A and publishes Topic A with a rate of 5000-6000 samples/sec. It started with a payload of 32 bytes and repeated up to a maximum payload of 8000 bytes. Every payload was tested for 200 seconds.
2) *Process_P2:* This process runs on machine B. It receives Topic A.

Throughput test setup is shown in figure:5. Figure:6 shows one-to-one (point-to-point) publish/subscribe throughput of RTI DDS, Fast-RTPS, CoreDx, OpenSplice and OpenDDS in terms of network bandwidth (megabits per second). It was measured between a single producing (Data writer) and consuming (Data reader) thread, over Gigabit Ethernet and a single DDS topic. Almost all implementations were able to utilize the available bandwidth. The data shows Connext DDS is able to fully utilize all of this available bandwidth when sending messages larger than 256 bytes with C++ applications. For Vortex OpenSplice with RT Networking service and FastRtps throughput values larger than 600Mbits/sec are obtained with messages from 1024 bytes. The maximum throughput value obtained remains stable around 950MBits/sec when the message size increases, meaning that we can take advantage of about a 95% of the available bandwidth. For CoreDx throughput values ranges from 660Mbits/s at 32 bytes to 810Mits/sec at 1024 bytes beyond that CoreDX was able to fully utilize all the available bandwidth. In Figure:7 we can see that OpenDDS higher throughput values are achieved only with data sizes more than 25000 bytes and maximum achievable throughput was below 800Mbps [14] [15] [16] [17] [18] [19] [20] .

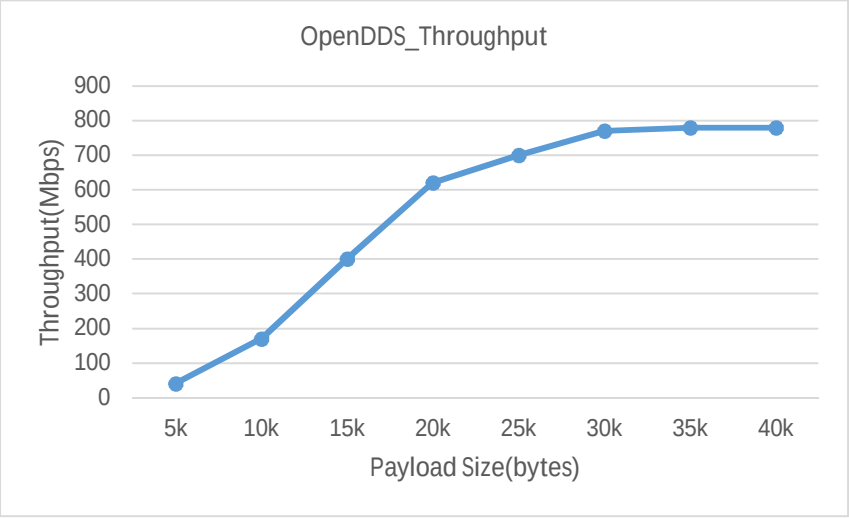


Figure: 6 OpenDDS Throughput

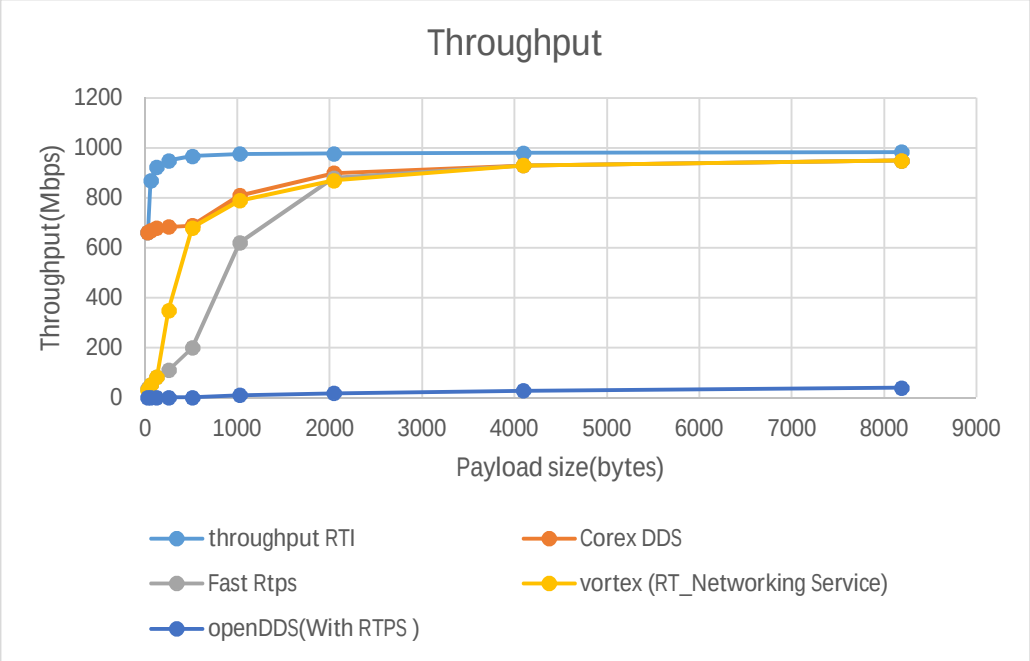


Figure:7 Throughput Comparison

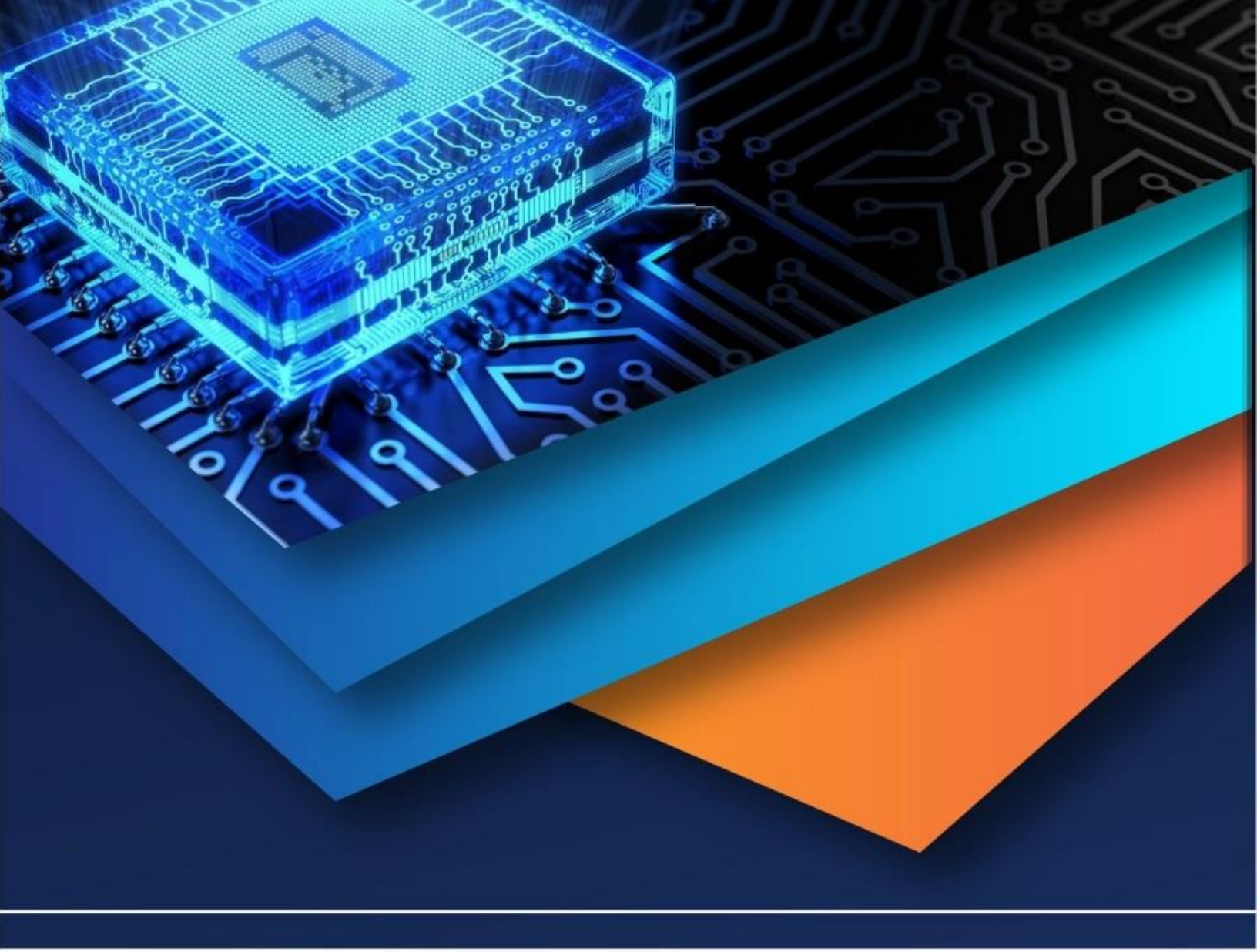
IV. CONCLUSIONS

The work presented in this paper shall provide the users a guide to select the best DDS implementation according to their application development and Deployment requirements. This paper, provides a criteria set which evaluates and compares different DDS implementations. DDS implementations with federated(centralized) architecture pose a communication bottleneck at high data rates and a single point of failure within each physical host. This problem is solved by implementations with non-federated or de-centralized architecture. All implementations provide support for routing and persistence with some manual configurations overhead which are least in RTI Connext. For small data sizes Message latency is better in Vortex OpenSplice and for data sizes larger than 1024 bytes RTI performs better. Throughput for almost all distributions is same for large data sizes and all implementations were able to utilize the available bandwidth except OpenDDS. Commercial implementations of DDS have many features like routing service, advance monitoring tool, persistence service, XML based dynamic DDS entities creation and deletion, record, replay services. In free distribution OpenDDS provides most of the OMG compliance features required for publish- subscribe communication. Further, the paper plans in extending work to explore data security perspective in DDS middleware.

REFERENCES

[1] OMG, "Data Distribution Service for Real-Time Systems Specification", March 2004
[2] McCormick, L. Madden, "Open Architecture Publish-Subscribe Benchmarking", OMG Real-time and Embedded Systems Workshop, 2005,URL:http://www.omg.org/news/meetings/workshops/RT 2005/03-3 McCormick-Madden.pdf
[3] K. Krinkin, A. Filatov, A. Filatov, O. Kurishev and A. Lyanguzov, "Data Distribution Services Performance Evaluation Framework," 2018 22nd Conference of Open Innovations Association (FRUCT), Jyvaskyla, 2018, pp. 94-100.
[4] The Real-time Publish-Subscribe Wire Protocol DDS Interoperability Wire Protocol (DDSI), [Online] Available: https://www.omg.org/spec/DDSI/
[5] http://download.prismtech.com/docs/Vortex/html/ospl/DeploymentGuide/ddsi2-networking-service.html
[6] RTI Connext DDS Core Libraries and Utilities, User Manual Version 5.1.0, December 2013, [Online] Available:
[7] https://istkb.adlinktech.com/article/vortex-opensplice-documentation/
[8] https://fastrtps.docs.eprosima.com/en/v1.6.0/persistence.html
[9] Routing Service, [Online] Available: https://www.rti.com/products/routing-service
[10] https://www.adlinktech.com/en/vortex-link-data-distribution-service
[11] https://github.com/eProsima/Integration-Service
[12] Communication middleware and DDS, December, 2011[Online]Available:http://www.twinoakscomputing.com/coredx/documentation
[13] DDS in Low-Bandwidth Environments, Workshop on Real-time, Embedded and Enterprise-Scale Time-Critical Systems, April 17-19, 2012, Concorde La Fayette, Paris, France
[14] Evaluating the Performance of Publish/Subscribe Platforms for Information Management in Distributed Real-time and Embedded Systems [Online] Available: https://www.researchgate.net /publication /238695989_ Evaluating_the_Performance_of_PublishSubscribe_Platforms_for_Information_Management_in_Distributed_Real-time_and_Embedded_Systems
[15] Network Performance and Benchmark [Online] Available: https://www.rti.com/products/benchmarks
[16] The DDS Tutorial release [Online] Available: https://www/download.prismtech.com/doc.
[17] Vortex OpenSplice DDS Testing Results: [Online] Available: https://www.adlinktech.com/en/vortex-opensplice-performance.aspx
[18] OpenDDS Performance Testing Results, [Online] Available: https://objectcomputing.com/ resources/ publications/mnb/interpreting-opendds-performance-testing-results.
[19] Network Performance and Benchmark [Online] Available:http://www.twinoakscomputing.com/coredx/performance_network.
[20] Network Performance and , [Online] Available: https://www.eprosima.com/index.php/resources-all/performance/40-eprosima-fast-rtps-performance.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue I Jan 2021- Available at www.ijraset.com



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY