Embedded Self Organizing Systems (Vol 9. No 3. 2022) (pp.62-69)

Embedded Selforganizing Systems

Special Issue Topic: "International Symposium on Computer Science and Educational Technology"

Low Latency Reliable Data Sharing Mechanism for UAV Swarm Missions

Fatih Kilic
Department of Computer Engineering
Technische Universität Chemnitz
Chemnitz, Germany
fatih.kilic@informatik.tu-chemnitz.de

Wolfram Hardt
Department of Computer Engineering
Technische Universität Chemnitz
Chemnitz, Germany
wolfram.hardt@informatik.tu-chemnitz.de

***Abstract*— The use of Unmanned Aerial Vehicle (UAV) swarms is increasing in many commercial applications as well as military applications (such as reconnaissance missions, search and rescue missions). Autonomous UAV swarm systems rely on multi-node interhost communication, which is used in coordination for complex tasks. Reliability and low latency in data transfer play an important role in the maintenance of UAV coordination for these tasks. In these applications, the control of UAVs is performed by autonomous software and any failure in data reception may have catastrophic consequences. On the other hand, there are lots of factors that affect communication link performance such as path loss, interference, etc. in communication technology (WIFI, 5G, etc.), transport layer protocol, network topology, and so on. Therefore, the necessity of reliable and low latency data sharing mechanisms among UAVs comes into prominence gradually. This paper examines available middleware solutions, transport layer protocols, and data serialization formats. Based on evaluation results, this research proposes a middleware concept for mobile wireless networks like UAV swarm systems.**

***Index Terms*— middleware, transport protocol, data serialization, UAV swarm**

I. INTRODUCTION

Cyber-physical systems (CPSs) like autonomous vehicles and UAVs can be capable of making decisions and operating independently by generally combining sensor networks with embedded computing to monitor and control the physical environment. This requires reliable and low-latency real-time data exchange. Data Distribution Service (DDS), which is also a middleware that enables reliable, high-performance, real-time data exchange using a publish-subscribe pattern among these cyber-physical systems [1].

DDS addresses the needs of applications like aerospace and defense, air-traffic control, autonomous vehicles, medical devices, robotics, power generation, simulation and testing, smart grid management, transportation systems, and other applications that require real-time data exchange [1]. To give an example for the use case of DDS in autonomous vehicles, cooperative perception is one of the most important DDS use cases where data exchange is very critical. It is also known as cooperative sensing or collective perception which enables vehicles and infrastructure nodes to detect objects (e.g. non-connected vehicles, pedestrians, obstacles) beyond their local sensing capabilities. Its goal is to enable the wireless exchange of sensor information between vehicles and infrastructure nodes in order to overcome the limitations. For example, in situations where the sensor's field of vision is blocked (by other vehicles or buildings) and cooperative perception improves the perception capabilities of the vehicles [2] [3].

In Fig. 1, the white car is unaware of the pedestrians on the crosswalk and this information is transferred by the red car to the white car, similarly, the red car is unaware of the obstacle on the way in Fig. 2 and the obstacle information is provided by the blue car to the red car in order to prevent an accident in both situations.

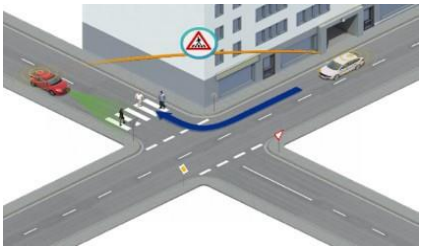

Fig. 1: Collective Perception [2]

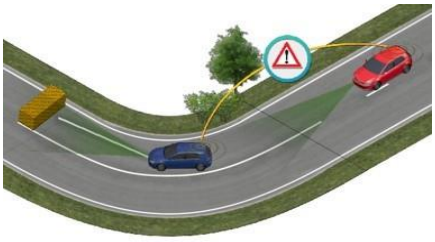

Fig. 2: Collective Perception [3]

Cooperative surveillance and collaboration systems are examples of a DDS use case in the UAV application area.

In Fig. 3, each UAV and Unmanned Ground Vehicle (UGV) have a different range of coverage areas and tasks to establish. They exchange information in real-time in order to effectively complete a cooperative surveillance mission.

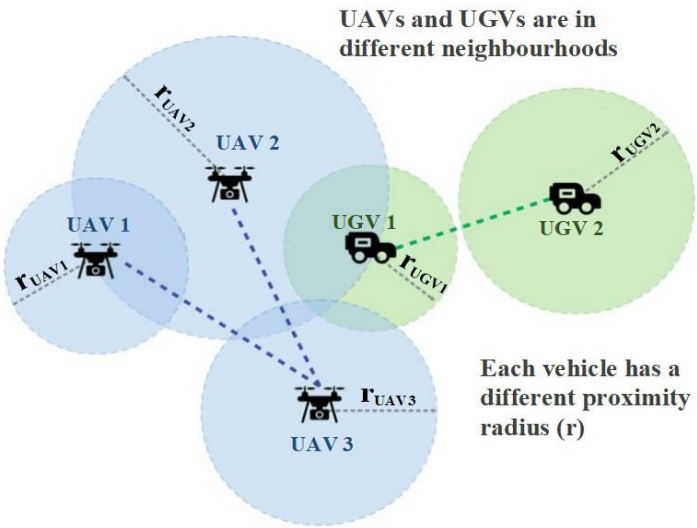

Fig. 3: Cooperative Surveillance using UAV Swarms [4]

In Fig. 4, each UAV has different sensors and is responsible for different tasks such as human detection, perceiving breath signals using bio-radar, and carrying emergency relief supplies in order to complete a complex mission by exchanging information.

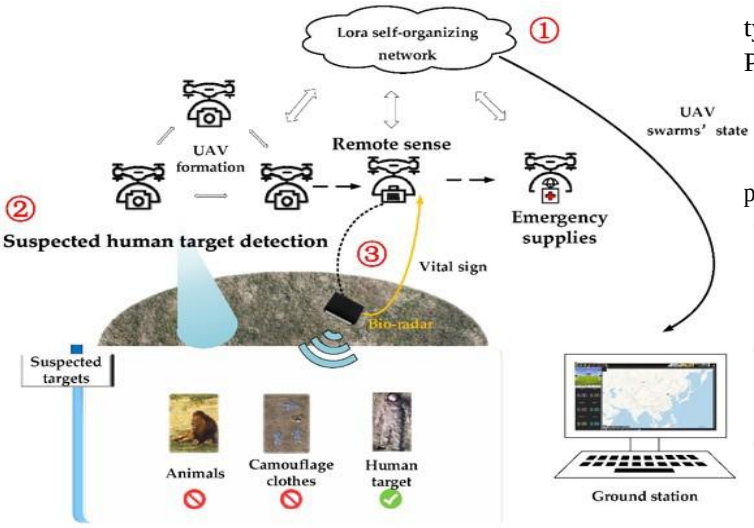

Fig. 4: UAV Swarms' Collaboration System [5]

II. MOTIVATION

Wireless communication environment has various problems like path loss, shadowing, signal fading, and interference. These pose a significant threat to achievable throughput, latency, and reliability of data transmission for autonomous UAVs and vehicles. There has been a significant improvement in communication technologies such as 5G and IEEE 802.11p in the last decade. These technologies especially contributed to latency and throughput which aim to achieve highly reliable communication among autonomous systems. However, today's middleware solutions mostly use TCP, UDP, or optimized UDP that integrates a QoS mechanism on top of UDP; as a transport layer protocol.

TCP is known for its reliability and UDP for its simplicity and low latency. The drawbacks of TCP and UDP have a significant effect on the communication for data exchange in autonomous systems. While TCP has high reliability, it overloads the network with packets when a loss packet is encountered. This causes latency and is not suitable for CPSs where the latency is significant. While UDP has low latency, it lacks reliability which may cause major problems in real-time systems. Some of the middleware solutions use Real Time Publish Subscribe (RTPS) protocol or their own QoS mechanism in order to ensure data reliability and low latency. However, these solutions, unfortunately, do not have adaptation between application-level requirements and communication channel constraints which is important for CPSs, which briefly means that the transport layer protocol is aware of latencies that occur at runtime. This helps to avoid the latency in the first place or accurately predict them if it cannot be avoided. In order to overcome this problem, this paper compares state-of-the-art transport layer protocols and available middleware solutions, and then proposes the use of Predictably Reliable Real-Time (PRRT) transport layer protocol which is highly reliable, has low latency, and is specially designed for CPSs in order to propose a reliable low latency data sharing mechanism [6]. Additionally, the performance of different types of data serialization formats is evaluated using the PRRT protocol.

III. STATE OF THE ART

In this section, the state-of-the-art research includes three main points;

- Available open-source middleware solutions such as eCAL, LCM, etc. as well as eProsima Fast DDS and Eclipse Cyclone DDS which implement open-source DDS \& RTPS.
- Comparison of transport layer protocols with respect to various metrics like reliability, error control mechanism, etc.
- Data serialization formats and their latency evaluation using PRRT protocol

A. Middlewares

DDS is a networking middleware that implements a publish-subscribe pattern for sending and receiving data, events, and commands among the nodes. Nodes generating information are known as publishers. Publishers create "topics" (e.g., temperature, location, pressure) and publish "samples". DDS delivers these samples to subscribers that declare an interest in that topic

(e.g., temperature) [1]. Some open-source DDS implementations like eProsima Fast DDS use Real-time Publish-Subscribe Protocol (RTPS). RTPS was specifically developed to support the unique requirements of DDSs and is designed to be able to run over multicast and connectionless best-effort transports such as UDP/IP [7].

As can be seen in Table 1, popular middlewares use RTPS protocol over UDP as a transport layer protocol, while others use TCP\&UDP, and almost all utilize Inter-process Communications (IPC) to share data between services on a single machine. There are also many popular middlewares used in IoT such as ZeroMQ, MQTT, MQTT-SN, RabbitMQ, ActiveMQ, and gRPC, etc. Data distribution in most of these middleware solutions is based on a message broker, message queue mechanism, or TCP which is beneficial for different use cases but not suitable for CPSs [8].

In [9], it is stated that eProsima FastRTPS for DDS delivers high performance in the round-trip time (RTT) of packages sent to the server. According to the performance evaluations in [10] [11], it is striking fast and performs better than alternatives such as ZeroMQ and other DDS middleware solutions. Therefore, eProsima Fast DDS is chosen as a state-of-the-art solution to compare its performance with the PRRT protocol (explained in the next subsection).

B. Transport Layer Protocols

Many successful protocols have emerged such as RTP, QUIC, SRT, etc. for real-time applications.

RTP (Real-time Transport Protocol): "provides end-to-end network transport functions suitable for applications trans- mitting real-time data such as audio, video or simulation data, over multicast or unicast network services" [12, p.1].

QUIC: is a multiplexed transport over UDP [13].

SRT (Secure Reliable Transport Protocol): is a transport protocol that was developed for low latency live video and audio streaming, as well as data transfer [14].

In Table 2, the comparison of transport layer protocols is given. Each of these protocols provides well-tuned functionality for specific purposes or application domains. Among these protocols, PRRT has two major application domains: live multimedia applications and control applications [6].

PRRT as a transport protocol adapts application-level requirements with communication channel constraints. It provides low latency and reliability for live multimedia as well as control applications. It uses adaptive hybrid error coding architecture that "enables the protocol to adaptively follow the dynamic capacity of the packet-erasure channels generated by a wide range of Internet protocol infrastructures. Combined with packet loss notifications via negative acknowledgments, it provides capacity-approaching coding efficiency in point-to-point as well as one-to-many transmission scenarios" [15, p.1].

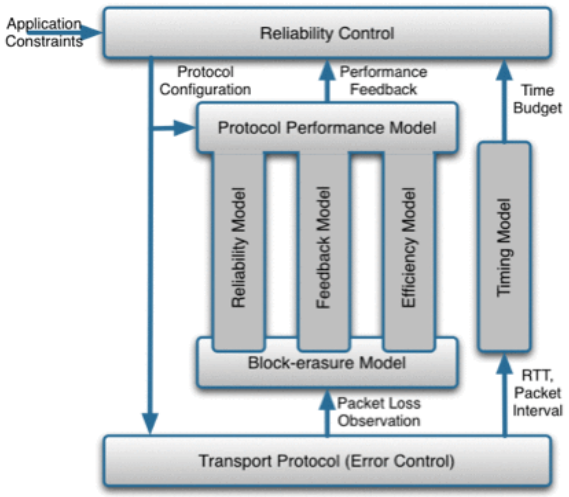

Fig. 5: Predictable Reliability [16]

PRRT only requires addressing and multiplexing to be done by the lower layer (e.g., UDP/IP, Ethernet). Therefore, it is not relying on communication processes (e.g. WIFI, LTE, Ethernet) used for the application. It is aware of the varying parameters in the channels established between communication processes and copes with them to ensure the reliability and latency constraints of the application [6].

"This holistic awareness is leveraged in decisions about the configuration of hybrid error control, loss-avoiding congestion control, as well as cross-layer packet pacing-leading to predictably low latencies and hence reliable timing" [6,p.6].

Due to the benefits of the PRRT protocol explained earlier and advantages given in Table 2 such as latency awareness and latency-predictability, the PRRT transport layer protocol is chosen to compare its performance with the state-of-the-art middleware solution "eProsima Fast DDS".

C. Data Serialization Formats

Data serialization is the process of encoding data objects into a byte stream in order to store, transfer and distribute this byte stream on physical devices 17]. There are different data serialization types and many formats available such as text-based "XML" and binary "Protobuf". The selection of data serialization type and format is determined by the intended application considering various factors such as data complexity, need for human readability, speed, and storage space constraints. The amount of data and the serialization or deserialization speed also directly affect the processing time and memory utilization [18].

A comprehensive comparison of data serialization formats is given in Table 3. Considering the disadvantages of data serialization formats like; N3, it is not chosen due to its time consumption during parsing, or Apache Thrift, it is not chosen due to lack of support for streaming large amounts of data [19]. CSV, JSON, YAML, BSON, MessagePack, and Protobuf are chosen according to the evaluations in [17]. FlatBuffers has not been tested due to the issues occurred in implementation. It will be evaluated and given with further development results.

TABLE I: Comparison of Middlewares

Name	Operation mode	Transport between processes on a single machine	Network transport	Security	Language support
eProsima Fast DDS [20]	publish-subscribe	IPC	TCP, RTPS (over UDP)	authentication, access control encryption)	Python, C++
Eclipse Cyclone DDS [21]	publish-subscribe	IPC	RTPS (over UDP)	authentication, access control encryption)	C
eCAL [22]	publish-subscribe, server-client	IPC	TCP, UDP	N.A.	Python, C, C++)
LCM [23]	publish-subscribe	IPC	UDP	N.A.	Python, C, C++ and more.
Aeron [24]	publish-subscribe	IPC	Aeron (over UDP)	encryption	Java, C++
UAVCAN [25]	publish-subscribe, server-client	CAN Bus	N.A.	N.A.	Python, C, C++
MAVLink [26]	publish-subscribe, server-client	N.A.	TCP, UDP)	authentication	Python, C, C++ and more.

TABLE II: Transport Layer Protocols [6]

	PRRT	TCP	UDP	RTP	QUIC	SRT
Reliability	< 100% (configurable)	100%	best effort	configurable via profile	100%	< 100%
Error Control	HARQ	ARQ	N.A.	configurable via profile	ARQ & XORFEC	ARQ
Acknowledgment Scheme	SACK	CACK & SACK	N.A.	configurable via profile	SACK& NACK	SACK& NACK
Congestion Control	BBR based	various	N.A.	see [27]	various	custom (live & file mode)
Segmentation	no	yes	no	yes	yes	yes
Latency Awareness	yes	no	no	no	no	no
Latency Predictability	high	low	low	low	low	low

TABLE III: Data Serialization Formats

Name	Type	Applications	Advantages	Disadvantages
XML	text based	exchange of information through web services [28]	large user base [29], does not require library support [6]	overly verbose, not efficient due to the size overhead & complex encoding mechanisms [30]
JSON	text based	common standard for many applications on the web [18]	lightweight and a more efficient alternative to XML [29], does not require library support [30]	lack of namespace support and input validation [29], power consumption
YAML	text based	web applications [18]	higher readability than JSON [18]	relatively high complexity implies low parsing speed [18]
CSV	text based	large dataset transmission [31]	compact and efficient way to transfer large datasets [31]	only supports serialization of a single result set [31]
N3	text based	Serialization of Resource Description Framework [32]	more readable than regular RDF/XML notation [33]	supports RDF rules, consumes more time to parse [32]
EN	text based	sensors with limited computation and communication capabilities [34]	one of the least payload sizes, comparatively has the lowest latency and minimal resource usage [19]	uses a predefined template created by the user, processing large data structures can require considerable resources. EN does not support nesting [34]

Continued on next page

Name	Type	Applications	Advantages	Disadvantages
Message-Pack	binary	IoT applications [30]	schemaless, implementation concept is simpler than Protobuf and FlatBuffers [30]	require library support [30]
Protobuf	binary	IoT applications where sensor nodes that deal mostly with numeric sensor readings [30]	efficient encoding [35], high energy efficiency [35], extremely lightweight [29], platform-neutral and language-neutral [30]	requires library support and is based on schema [30]
FlatBuffers	binary	developed for performance critical applications [30]	better memory efficiency and speed compared to Protobuf [30]	requires library support and is based on schema [30]
BSON	binary	MongoDB&NoSQL databases [36], IoT applications [37]	similar to JSON but faster and not human readable [37]	has a limit to a maximum size of document no more than 16Mb [37]
Apache Thrift	binary	web applications [30]	extremely lightweight [29], fast to serialize and deserialize [29]	lack of support for streaming large amounts of data [29]
XDR	binary	exchange of information through web services [28]	efficient encoding [35], high energy efficiency [35]	lack of schema, has limited code generator and library support in modern programming languages [35]

IV. CONCEPT

Based on the advantages of the PRRT protocol, the middleware shown in Fig. 6 is being developed and tested by using the PRRT transport protocol. The middleware implements four main modules at the moment.

- *Preliminary Tasks Module* is responsible for general tasks such as node discovery which allows a device to obtain the available topic's "data" and its publisher's "IP address". Additionally, each device keeps the track of its subscriber list for required topics. The development of this module is still in progress.
- *Data Serialization Module* is used to serialize\slash deserialize the data before\slash after the data transfer. In this module, data serialization formats such as JSON, BSON, YAML, MessagePack, and Protobuf are used to encode/decode data.
- *Transport Layer Module* allows us to send multicast and unicast data using PRRT protocol.
- *Inter-process Communication (IPC) Module* allows low latency data read/write mechanism between processes using Shared Memory. The main reason for integrating IPC into the proposed middleware is to allow internal services access to the shared data on the companion board. In our UAV swarm use case scenario, the sensor data shared by neighbor UAVs is stored in shared memory as soon as it is received by the PRRT socket. This enables the software modules such as decision mechanism module to use the data to perform better in situations like UAV swarm formation or collision avoidance.

In the Result section, the analysis of data serialization formats is given with respect to serialization, deserialization and total transmission time. The hardware and the method to calculate these measurements are also explained.

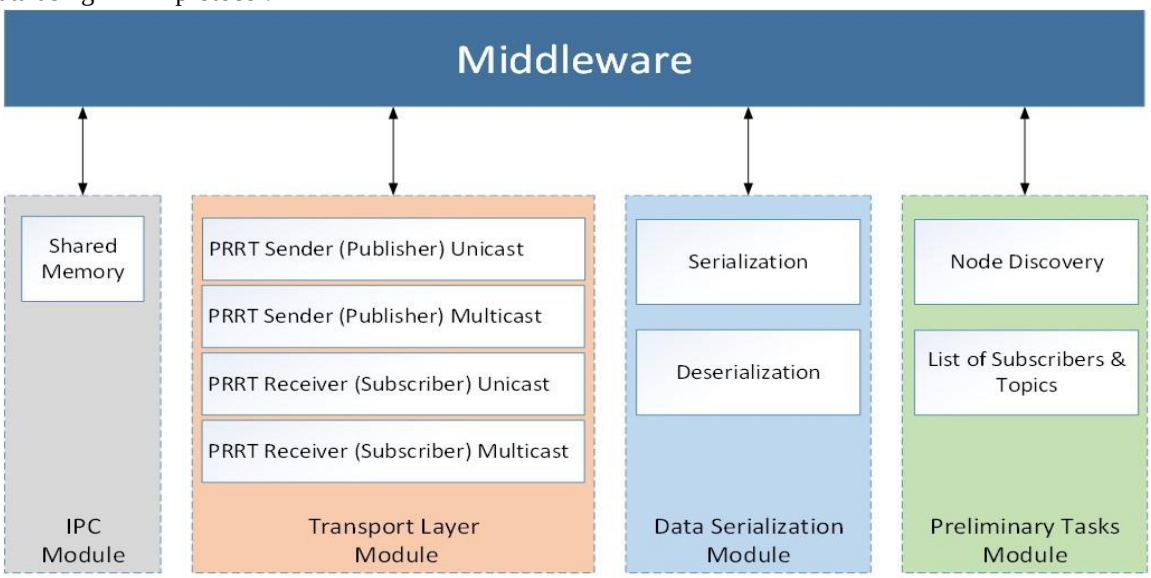

Fig. 6: The middleware concept for UAV Swarm Missions using PRRT Protocol

V. RESULTS

The ODROID-XU4 companion board, EW-7811Un WLAN adapter, and Belkin N300 wireless router are used for implementation and to obtain results in our tests. The ODROID-XU4 boards do not have a direct connection to each other, they are directly connected to the router. The time synchronization between devices is enabled by Network Time Protocol (NTP) in order to have accurate results when timestamps are used. NTP is the standard for synchronizing time between two devices over a network [38].

Every serialization format takes varying times to perform the serialization of data. In order to calculate this, a timestamp is recorded before performing serialization and another timestamp is recorded right after the serialization of data is completed. Hence, the difference between these two timestamps gives us the time taken for serialization. The same method is used to calculate deserialization time. In addition to this, python implementation of serialization formats is used in latency measurements.

The serialization formats were tested with different amounts of data (1KB, 5KB, and 1MB) and compared with respect to their serialization and deserialization time. It can be clearly seen that the MessagePack serialization format has the lowest serialization and deserialization times for the different amounts of data.

Additionally, in order to ensure that the receiver is aware of the transmission status when the data transfer is completed, a UTF-8 encoded data containing the string “END” packet is generated. This packet is transmitted by the sender to the receiver after the transmission of the serialized data is completed. This indicates the end of the data transmission and is recognized by the receiver. The receiver initiates the deserialization of the received data after receiving the "END" packet.

The “END” packet is also used to calculate the total transmission time which is the time taken to transfer serialized data from sender to receiver. The total transmission time includes network latency. To calculate the total transmission time, two timestamps are used. The first timestamp is recorded before the start of serialized data transmission at the sender side and another timestamp is recorded right after the receiver receives UTF-8 encoded string containing the "END" packet. The time difference between these timestamps gives us the total transmission time.

Table 4 gives the serialization and deserialization time in milliseconds and Table 5 shows the comparison of data serialization formats with respect to the total time taken to transfer serialized data from sender to receiver in different amounts of data using PRRT protocol. The total time of BSON and MessagePack formats for 1KB of data is the lowest whereas the total time of JSON and MessagePack formats for 5KB of data is the lowest. However, MessagePack still has the best performance for 1MB data, it is quite fast.

TABLE IV: Serialization and deserialization time (ms)

	Serialization Time			**Deserialization Time**		
	1KB data	**5KB data**	**1MB data**	**1KB data**	**5KB data**	**1MB data**
CSV	0.706	1.983	377	0.565	1.183	100
JSON	0.644	1.752	258	0.621	1.181	153
YAML	15.638	80.512	14803	31.615	148.963	12063
BSON	1.391	6.011	1074	0.534	2.17	414
MsgPack	0.427	0.546	18	0.199	0.411	25
Protobuf	1.416	1.395	19	0.24	1.265	55

TABLE V: Transfer time of serialized data (ms)

	Transfer Time		
	1KB data	**5KB data**	**1MB data**
CSV	16.022	43.126	3021
JSON	8.666	12.549	3417
YAML	6.588	51.535	21989
BSON	4.288	32.995	3296
MsgPack	4.923	13.024	1998
Protobuf	6.579	33.762	4625

In conclusion, different amounts of data transmission were tested using eProsima Fast DDS middleware and PRRT protocol. Additionally, single-value raw data which is a 1KB single string, is used in latency measurements. Different amounts of data transmission mean that single-value raw data is contained e.g. 1 time, 5 times, and 1000 times in 1KB, 5KB, and 1MB data, respectively.

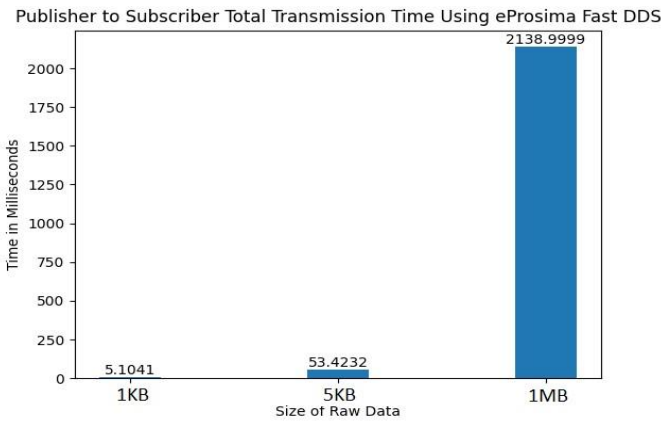

Fig. 7: Comparison of the total amount of time to transfer 1KB, 5KB and 1MB raw data using eProsima Fast DDS

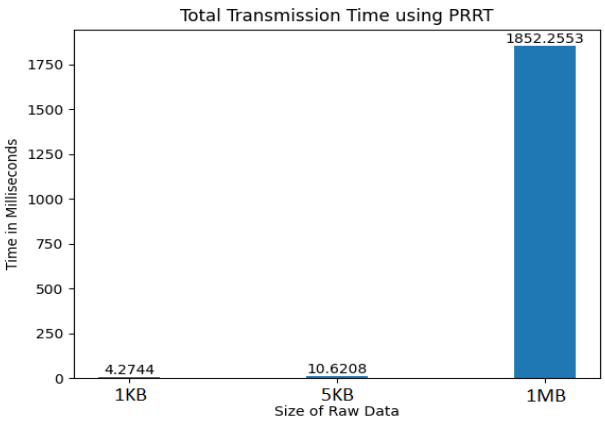

Fig. 8: Comparison of the total amount of time to transfer 1KB, 5KB and 10KB raw data using PRRT

VI. CONCLUSION

In this study, the low latency reliable data-sharing mechanism concept for UAV Swarm Missions is proposed by analyzing the state-of-the-art middleware solutions, transport layer protocols, and data serialization formats. The results of this study show that PRRT is performing slightly better than eProsima Fast DDS with respect to latency in peer-to-peer data transfer. In addition to this, MessagePack has the lowest serialization/deserialization time, and also it adds very low latency.

The results of this research support the idea of using PRRT protocol for data transfer among UAV swarm nodes since it gives promising results compared to eProsima Fast DDS middleware. Therefore, the following points will be investigated as a next step: comparison of multicast data transfer using PRRT and eProsima Fast DDS, integration of Shared Memory (IPC Module), and integration of Network Discovery and Pub/Sub Pattern features.

REFERENCES

[1] ‘‘What is DDS?’’ dds-foundation.org. https://www.dds-foundation.org/what-is-dds-3 (Accessed: Nov. 23, 2022).

[2] Garlichs, ‘‘Simulative untersuchungen zur kollektiven wahrnehmung,’’ ibr.cs.tu-bs.de. https://www.ibr.cs.tu-bs.de/projects/collective- perception/ (Accessed: Nov. 23, 2022).

[3] H.-J. Gunther, B. Mennenga, O. Trauer, R. Riebl, and L. Wolf, ‘‘Realizing collective perception in a vehicle,’’ in 2016 *IEEE Vehicular Networking Conference (VN C)*. Columbus, OH, USA: IEEE, December 2016. [Online]. Available: https://doi.org/10.1109/vnc.2016.7835930

[4] D. H. Stolfi, M. R. Brust, G. Danoy, and P. Bouvry, ‘‘UAV-UGV-UMV Multi-Swarms for Cooperative Surveillance,’’ *Frontiers in Robotics and AI*, vol. 8, February 2021. [Online]. Available: https://doi.org/10.3389/frobt.2021.616950

[5] Y. Cao, F. Qi, Y. Jing, M. Zhu, T. Lei, Z. Li, J. Xia, J. Wang, and G. Lu, ‘‘Mission Chain Driven Unmanned Aerial Vehicle Swarms Cooperation for the Search and Rescue of Outdoor Injured Human Targets,’’ *Drones*, vol. 6, no. 6, p. 138, May 2022. [Online]. Available: https://doi.org/10.3390/drones6060138

[6] A. Schmidt, ‘‘Cross-layer latency-aware and -predictable data communication, doi:10.22028/d291-30851,’’ Ph.D. dissertation, Fakultät für Mathematik und Informatik., Universität des Saarlandes, Saarbrücken, 2019.

[7] ‘‘About the DDS Interoperability Wire Protocol specification version 2.5.’’ omg.org. https://www.omg.org/spec/DDSI-RTPS/2.5/About-DDSI-RTPS/ (Accessed: Nov. 23, 2022).

[8] T. Treat, ‘‘Dissecting message queues, Brave New Geek,’’ bravenewgeek.com. https://bravenewgeek.com/dissecting-message- queues/ (Accessed: Nov. 23, 2022).

[9] S. Profanter, A. Tekat, K. Dorofeev, M. Rickert, and A. Knoll, ‘‘OPC UAversus ROS, DDS, and MQTT: Performance Evaluation of Industry 4.0 Protocols,’’ in *2019 IEEE International Conference on Industrial Technology (ICIT)*. Melbourne, VIC, Australia: IEEE, February 2019. [Online]. Available: https://doi.org/10.1109/ICIT.2019.8755050

[10] “Fast DDS Performance,’’ eprosima.com. https://www.eprosima.com/index.php/resources-all/performance (Accessed: Nov. 23, 2022).

[11] “Fast DDS vs Cyclone DDS,’’ eprosima.com. https://www.eprosima.com/index.php/resources- all/performance/fast-dds-vs-cyclone-dds-performance (Accessed: Nov. 23, 2022).

[12] H. Schulzrinne, S. L. Casner, R. Frederick, and V. Jacobson, “RTP: A Transport Protocol for Real-Time Applications,’’ RFC 3550, Tech. Rep. 3550, Jul. 2003. [Online]. Available: https://www.rfc- editor.org/info/rfc3550

[13] “QUIC, a multiplexed transport over UDP,’’ chromium.org. https://www.chromium.org/quic/ (Accessed: Nov. 23, 2022).

[14] “Secure Reliable Transport, (SRT) Protocol,’’ github.com. https://github.com/Haivision/srt (Accessed: Nov. 23, 2022).

[15] “LARN / PRRT,’’ git.nt.unisaarland.de. https://git.nt.unisaarland.de/LARN/PRRT (Accessed: Nov. 23, 2022).

[16] “PRRT: Predictably Reliable Real-Time Transport | Telecommunications Lab.’’ nt.uni-saarland.de. http://www.nt.uni-saarland.de/project/prrt-predictably-reliable-real-time-transport/ (Accessed: Nov. 23, 2022).

[17] “Data serialization,’’ devopedia.org. https://devopedia.org/data-serialization (Accessed: Nov. 23, 2022).

[18] M. Eriksson and V. Hallberg, ‘‘Comparison between json and yaml for data serialization,’’ Bachelor’s Thesis, KTH Royal Institute of Technology, 2011.

[19] X. S. Altti Ilari Maarala and J. Riekki, “Semantic reasoning for context-aware internet of things applications,’’ *IEEE Internet of Things Journal*, vol. 4, no. 2, p. 461–473, 2016. [Online]. Available: https://doi.org/10.48550/arXiv.1604.08340

[20] “eProsima FastDDS Documentation,’’ buildmedia.readthedocs.org. https://buildmedia.readthedocs.org/media/pdf/eprosima-fast-rtps/latest/eprosima-fast-rtps.pdf (Accessed: Nov. 23, 2022).

[21] “Eclipse Cyclone DDS™,’’ projects.eclipse.org. https://projects.eclipse.org/projects/iot.cyclonedds (Accessed: Nov. 23, 2022).

[22] “eCAL - enhanced Communication Abstraction Layer,’’ github.com. https://github.com/eclipse-ecal/ecal (Accessed: Nov. 23, 2022).

[23] “Lightweight Communications and Marshalling (LCM),’’ github.com. https://lcm-proj.github.io/ (Accessed: Nov. 23, 2022).

[24] “real-logic/aeron: Efficient reliable UDP unicast, UDP multicast, and IPC message transport,’’ github.com. https://github.com/real- logic/aeron (Accessed: Nov. 23, 2022).

[25] “Uncomplicated Application-layer Vehicular Computing And Networking,’’ uavcan.org. https://uavcan.org (Accessed: Nov. 23, 2022).

[26] “Introduction · MAVLink Developer Guide,’’ mavlink.io. https://mavlink.io/en/ (Accessed: Nov. 23, 2022).

[27] C. Perkins and V. Singh, “Multimedia Congestion Control: Circuit Breakers for Unicast RTP Sessions,’’ Internet Engineering Task Force, Tech. Rep., March 2017, proposed Standard. [Online]. Available: https://www.rfc-editor.org/info/rfc8083

[28] D. C. Castillo, J. Rosales, and G. A. T. Blanco, ‘‘Optimizing binary serialization with an independent data definition format,’’ *International Journal of Computer Applications*, vol. 180, no. 28, pp. 15–18, March 2018.

[29] A. Sumaray and S. K. Makki, “A comparison of data serialization formats for optimal efficiency on a mobile platform,’’ in *Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication*, ser. ICUIMC ’12. New York, NY, USA: Association for Computing Machinery, February 2012. [Online]. Available: https://doi.org/10.1145/2184751.2184810

[30] A. K. Biswal and O. Almallah, ‘‘Analytical assessment of binary data serialization techniques in iot context,’’ Master’s thesis, Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria, December 2019.

[31] “CSV format,’’ docs.servicestack.net. https://docs.servicestack.net/csv- format (Accessed: Nov. 23, 2022).

[32] X. Su, H. Zhang, J. Riekki, A. Keränen, J. K. Nurminen, and L. Du, ‘‘Connecting IOT sensors to knowledge-based systems by transforming SenML to RDF,’’ *Procedia Computer Science*, vol. 32, p. 215–222, June 2014. [Online]. Available: https://doi.org/10.1016/j.procs.2014.05.417

[33] B. R. Kaithi, ‘‘Knowledge graph reasoning over unseen rdf data,’’ Master’s thesis, Wright State University, 2019.

[34[X. Su, J. Riekki, and J. Haverinen, “Entity notation: Enabling knowledge representations for resource-constrained sensors,” Personal and Ubiquitous Computing, vol. 16, p. 819–834, 2012. [Online]. Available: https://doi.org/10.1007/s00779-011-0453-6

[35] D. Friesel and O. Spinczyk, ‘‘Data Serialization Formats for the Inter- net of Things,’’ *Electronic Communications of the EASST*, vol. 80, 2021. [Online]. Available: http://dx.doi.org/10.14279/tuj.eceasst.80.1134

[36] J. C. Viotti and M. Kinderkhedia, “A Survey of JSON-compatible Binary Serialization Specifications,’’ *CoRR*, vol. abs/2201.02089, January 2022. [Online]. Available: https://arxiv.org/abs/2201.02089v2

[37] ‘‘MongoDB Limits and Thresholds,’’ mongodb.com. https://www.mongodb.com/docs/manual/reference/limits/ (Accessed: Nov. 23, 2022).

[38] “How to Install and Configure NTP on Linux,’’ timetoolsltd.com. https://timetoolsltd.com/ntp/how-to-install-and-configure- ntp-on-linux/ (Accessed: Nov. 23, 2022).