Analysis of swarm application layer protocols (SALP) used in event-driven communication

Antouan Anguelov
Faculty of Computer Systems and Technologies (FCST)
Technical University of Sofia
Sofia, Bulgaria
antouan@tu-sofia.bg

Roumen Trifonov
Faculty of Computer Systems and Technologies (FCST)
Technical University of Sofia
Sofia, Bulgaria
trifonov@tu-sofia.bg

Ognian Nakov
Faculty of Computer Systems and Technologies (FCST)
Technical University of Sofia
Sofia, Bulgaria
nakov@tu-sofia.bg

Communications and communication protocols play an important role in mobile robotic systems (MRS). Addressing real-time mobile robot applications, MRS should communicate with minimal effort and resources, dealing with long delays and any losses incurred. In a swarm scenario implementing a stack of communication protocols becomes a key factor for both the performance and the regular operation. The paper aims to review the most widely used application layer protocols for swarm robot scenarios. These protocols are the short publish-subscribe message protocols available for application layer communication (ALC). In the paper, we present specific design guidelines for the pipeline architecture implementing MAVLink, DDS-XRCE, and rosserial application protocols (SALP) in an event-driven real-time communication channel. The shown pipelines architecture is a conceptual model for hybrid and cluster-based distributed fault-tolerant networks using a zero-copy shared memory approach with over inter-process-communication (IPC).

Keywords— mobile robot systems, robot operating system, swarm, application layer protocol, event-driven communication, fog computing, distributed fault-tolerant network

I. INTRODUCTION AND PROBLEM STATEMENT

A system of peer-to-peer (PtP) networked agents (mobile robots) creates a decentralized and distributed group of less powerful nodes, synchronized over communication protocols called a swarm. In such swarm scenarios, communication between mobile robots is critical. If it is damaged or disrupted at some point, this will require the mobile robots to form an ad hoc network, using each other as redirecting nodes to create a new communication channel [2]. The data messages sent and received in that infrastructure additionally need to meet certain criteria such as the ability to be recovered. Thus, messages between the mobile robots require specific routing protocols at application level that can operate without central control or broker (server) following dynamic changes in topology due to the mobility of the environment in which robots operate.

A. Messages Categirozation

Transferring messages between the mobile agents requires the use of protocols without a central control station or server following the topology of the swarm. Data messages transmitted between the various MRS applications [1] are:

- Unicast when one mobile robot sends data to another concrete node or the central station.
- Multicast when a central server or a mobile robot sends data to the group of robots with coordination or additional control and sensor information.

Two main groups of communication messages are used in the MRS communication channel:

- Coordination as short massages contains meta data related to the coordination and control of the mobile robots. In some cases, coordination-oriented communication may require both unicast (one - another) and multicast (one - many) connection types.
- Data-oriented are heavy massages such pictures or videos.

B. Network Topologies [2]

- The centralized architectures consume mainly unicast messages because the central/coordinator communicates to the nodes with one-to-one communication.
- The decentralized and distributed designs expect robustness and the ability to handle failures by allows dynamic network configurations by utilizing multicast communication channels - the node needs to talk to all others. That is why distributed redundancy network of applications would need a lightweight protocol to achieve fast response times.
- Hierarchical or hybrid networks with multi-hops where a robot could act as a temporary broker/coordinator for a group of robots. It combines decentralized control and provides robustness with hierarchical control to achieve global synchronization and coordination.

By participating and subscribing to the appropriate topic of interest using a publish-subscribe message protocol, MRS can achieve the expected event-driven communication in a hybrid/hierarchical cluster-based and distributed fault-tolerant network. There will be multiple concurrent publishers and subscribers for each separate single topic [4], and a single robot may publish or subscribe to the various topics of interest. MRS is playing a server role (broker) in a given moment, thus establishing a high availability cluster as a so-called load balance structure.

II. SHORT REVIEW OF APPLICATION LAYER PROTOCOLS (SALP) FOR SWARM SCENARIOS

At the application layer, there are a few publish-subscribe pattern protocols used effectively in MRS swarm scenarios. Their key futures are summarized in the Table I.

A. MAVLink protocol

MAVLink [3] is a portable messaging protocol designed for the drone ecosystem widely used in wireless environments for communication between the Ground Control Stations (GCS) and all the nodes in autonomous systems. Being a very lightweight messaging protocol for communicating between onboard aerial vehicles (UAVs) components or through wireless with a GCS, MAVLink represents a hybrid between the publish-subscribe and the point-to-multipoint (PtM) design patterns. The data streams are published as topics while configuration sub-protocols such as the mission protocol or parameter protocol are point-to-point (PtP) with the retransmission option. The messages are design and constructed using XML files. Each XML file defines the message set supported by a particular MAVLink system, also referred to as a "dialect".

B. The data-distribution service (DDS) for extremely resource-constrained environments (XRCE) protocol

DDS-XRCE protocol is a client-server protocol used to transmit messages between resource-constrained devices (clients) and an XRCE agent acts as server shown in Fig. 1.

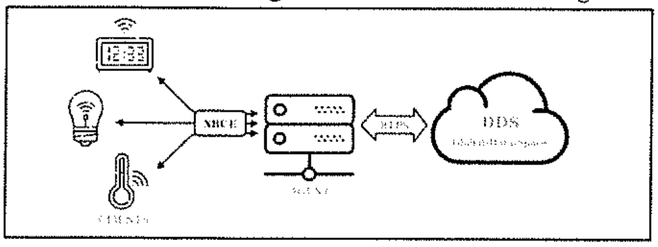

Fig. 1. The DDS-XRCE conceptual model. (https://micro.ros.org/docs/concepts/middleware/Micro_XRCE-DDS/)

Known as real-time publish-subscribe (RTPS) protocol, developed to support DDS applications is a publication-subscription communication middleware over best-effort transports such as UDP/IP. Furthermore, DDS provides support for TCP, serial, or fast shared memory (SHM) transports. The data distribution service (DDS) is a data-centric communication protocol used for distributed software application communications. The protocol allows resource collaborative robots embedded processors with sleep/wake cycles access to the DDS. It describes the communications application programming interfaces (APIs) and communication semantics that enable communication between data providers and consumers [4]. XRCE-DDS protocol supports routing, quality of service (QoS), and discovery options regarding the popular and widely adopted IoT industry publish-subscribe message queuing telemetry transport (MQTT) protocol.

Often XRCE is used for inter-process communication (IPC) as true zero-copy (shared memory), which allows data exchange between publishers to subscribers without a single copy. An example is the XRCE middleware "AceOrix" by Eclipse.org [5] plays a role as SHM transport. "AceOrix" has its origins in the automotive industry in driver assistance or automated systems for driving and robotics. "AceOrix" uses a true zero-copy, shared memory approach that allows transferring data from publishers to subscribers without a single copy, ensuring data transmissions with constant latency, regardless of the size of the payload. XRCE protocol becomes a powerful instrument if a large amount of data between different processes should be shared - among entities running on the same host.

C. Rosserial Protocol

Based on a client-server architecture approach, it is a protocol [6] for wrapping standard robot operating system (ROS) serialized messages and multiplexing multiple topics and services over a character device such as a serial port or network socket. rosserial allows platforms based on microcontrollers to communicate with a regular computer communicating with the ROS network on its behalf. Rosserial-clients serialize data into the serial link, and then, the serialized data is collected by the rosserial-server and forwarded to the conventional ROS network. Rosserial, in those cases, acts as a bridge between hardware communication protocols and a ROS network. Disadvantages of the rosserial are no routing capabilities with no reliability during network drops or problems.

TABLE I. SALP COMPARISON OVERVIEW (MAVLINK V2, ROSSERIAL, DDS-XRCE APPLICATION PROTOCOLS):

Key futures	**Application Protocols**		
	MAVLink	***Rosserial***	***DDS-XRCE***
Topology	PtM/PtP	PtP/PtM	PtP/PtM
Comm. type	Offboard	Onboard / offboard	Onboard / offboard
Architecture pattern	client-server	client-server	client-server
OS Platform	Any	Any	Any
Transport	Wieless/UDP/ TCP/serial	TCP/serial	UDP/TCP/ serial or SHM
Long range	Yes	No	No
QoS/Reliability/ Integrity	Yes	No	Yes
Protocol Dialects	Yes	No	No
Main domain	Drones/GCS	ROS	ROS2/multi
Programming Languages	Any	Any	Any
Security	Yes	No	Yes - DTLS/TLS
Max.# of nodes	255	25	n/c
Max payload size	255 bytes	512 bytes	64 KB
Discovery support	Yes	No	Yes
Routing	No	No	Yes

III. ARCHITECTURE FOR PIPELINE DISTRIBUTED FAULT-TOLERANT NETWORK BASED ON SALP

MRS would be maintaining simultaneous links of real-time communication between different applications onboard or offboard with the sensor, positioning, or other types of traffic. The research paper recommends architecture showing the practical implementations of SALP as a pipeline process running on different nodes or MRS. The architecture block diagram (Figure 2) shows event-driven communication with the message protocols MAVLink, rosserial and DDS-XRCE over the middleware "AceOrix" [5] acting as IPC SHM transport.

A. Building an event communication model

The very first step for designing a fault-tolerance communication system is to build an event communication model (the logic) of the stream processor by creating a decision model [7] with a rule engine system (RES) using the environment called "Drools" [8]. Drools is an open-source Business Rules Management System (BRMS) software. It provides a core Business Rules Engine (BRE), a web authoring and rules management application that supports Decision Model and Notation (DMN) models. Parameters such as type of input/output data, events, nodes, and communication protocols, in this step, are defined in advance.

B. Lightweight Complex Event Processor (CEP)

The next step is to organize each process running (as a task) in a separate thread. It is likely to manage the various pipelines in just one instance. The task connects via the instance pipelines. A real-time multi-threading process is defined. Handling the event is done by the so-called Lightweight Complex Event Processor (CEP). An example for implementing CEP is the application "Eventerpretor" [9]. Eventerpretor is an open-source node-based CEP written in Kotlin, intended to process discrete events which arrive sporadically or at low sample rates. Each processing task uses its own thread, and all tasks are connected via the pipelines. It is possible to run multiple pipelines in one instance. All tasks relate to each other by passing data into queues so that the nodes are not active until new data arrives. Usually, CEP becomes a part of real-time embedded processors operational system (OS). "TaskManagerIO" [10] as part of "IoAbstraction" [11] is easy to implement in low-power CPUs running Arduino on any other embedded real-time operating system (RTOS) such as "FreeRTOS". "TaskManagerIO" for "FreeRTOS" is backed by a simple queue that supports, immediate queuing, scheduled tasks, and events. It is safe to add tasks from another thread, and safe to trigger events from interrupts.

C. Design Guidelines for the pipeline architecture implementing SALP over inter-process-communication (IPC)

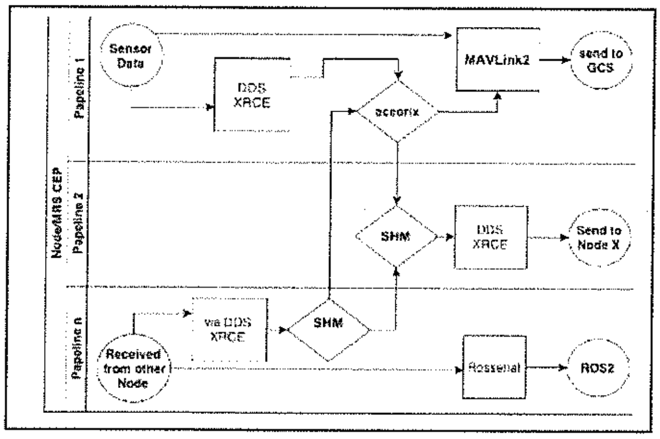

Fig. 2. Block diagram for distributed fault-tolerant hybrid network based on SALP

The design guidelines in Figure 2 target sample scenario with three separate pipelines for the existing one GCS and n-number of MRS:

- Pipeline 1. MRS sensor data is transmitted to the ground station (GSC) by the MAVLink protocol. DDS-XRCE protocol is processing the data to pipeline 2, using the inter-process-communication (IPC) in real-time. In our scenario, the middleware "AceOrix" [5] is used as SHM layer.
- Pipeline 2 run as a second separate process and is responsible for sending the same sensor data (from Pipeline 1) to the all other MRS in the network through IPC SHM and DDS-XRCE protocol by a hybrid short-range modular wireless transceiver (HSMWT) [12].
- Pipeline "n" is responsible for the coordination and position data messages received from all other MRS, passing it to the connected ROS2 application components. In case of communication failures, the pipeline will re-transmit messages to the GCS (Pipeline 1) and other nodes in the network (Pipeline 2) by using same IPC SHM and DDS-XRCE processes in pipeline 1 and 2.

IV. CONCLUSION

The research paper focused on recommending an architecture with SALP, provoking further experiments showing possible implementations in real-time scenarios as contribution. The analyzed protocols are the best candidates for event-driven communication proposed as system architecture in our research paper. Related solutions for the implementation of SALP can be found in scenarios where both drones or MRS need to be controlled in real-time [13]. Another practical implementation of the architectural model proposed (Fig. 2) could be the design of a new protocol to meet various QoS requirements such as throughput, jitter, lifetime and packet delivery ratio in order to improve the network of nodes in the wireless multimedia sensor networks (WMSN) [14].

However, several limitations merit comment:

- The synchronization between protocols in real-time is key component witch need further research;
- The robot-to-robot (R2R) communication is in the developing stage that is why there are no new communication standards implemented for the MRS application layer protocols;
- All big manufacturers are focused on cloud and cloud-based multi-robot systems;
- Fog computing and fog-based communication are just rising;
- True zero-copy shared memory approach shown in our design guidelines recently become available with new ROS2 showing its potential to replace rosserial;
- There are no quantitative or qualitative results with the shown SALP mechanism at this stage of the research.

Nevertheless, despite these limitations above, we believe this study opens new research opportunities although no quantitative or qualitative results from experiments or simulations with the SALP mechanism are presented in the paper.

We believe the shown insights in the paper will likely generate new novel methods of communication between MRS. Contrariwise, we will keep this discussion open for experts in messaging protocols. Using protocols such as DDS-XRCE, rosserial, and MAVlink in ROS2 future releases will further develop our method. There is, therefore, a great need for further research in this area.

ACKNOWLEDGMENT

We would like to acknowledge our colleagues from the faculty for the valuable insights, comments, and contribution towards the proposed protocols.

REFERENCES

[1] Saumitra M. Das, Y. Charlie Hu, C. S. George Lee, and Yung-Hsiang Lu, Mobility-Aware Ad Hoc Routing Protocols for Networking mobile robot teams, Journal of Communication and Networks, September 2007. DOI: 10.1109/JCN.2007.6182857

[2] Antouan Anguelov, Roumen Trifonov, Ognian Nakov, Protocol stack for hybrid short-range modular wireless transceiver, International Scientific Conference Computer Science'2020.

[3] Mavlink, The MAVLink Developer Guide, https://mavlink.io

[4] Eprosima Ltd., https://fast-dds.docs.eprosima.com/

[5] Inter-process-communication (IPC) middleware for various operating systems – "AccOrix", Eclipse.org, https://github.com/eclipse-iceoryx/iceoryx

[6] Robot operating system (ROS), https://micro.ros.org/docs/concepts/middleware/rosserial/

[7] Getting started with decision services - Decision Model and Notation (DMN), Redhat.com, https://access.redhat.com/documentation/en-us/red_hat_decision_manager/7.8/html-single/getting_started_with_decision_services/index?extIdCarryOver=true&sc_cid=701f2000001Css0AAC

[8] Business Rules Management System (BRMS) "Drools", https://www.drools.org

[9] Eventerpretor - an open-source node-based, lightweight Complex Event Processor (CEP), https://github.com/andreas-seiderer/Eventerpretor

[10] TaskManagerIO - library for Arduino and mbed, https://github.com/davetcc/TaskManagerIO/

[11] IoAbstraction - event-driven Arduino and mbed applications, https://www.thecoderscorner.com/products/arduino-libraries/io-abstraction/

[12] Antouan Anguelov, Roumen Trifonov, Ognian Nakov, Emerging and secured mobile ad-hoc wireless network (MANET) for swarm applications, BCI'19: Proceedings of the 9th Balkan Conference on Informatics, 2019. https://doi.org/10.1145/3351556.3351557.

[13] Ming Li, Kejie Lu, Hua Zhu, Min Chen, Shiwen Mao, Balakrishnan Prabhakaran, Robot swarm communication networks: Architectures, protocols, and applications, Communications and Networking in China, ChinaCom 2008, DOI: 10.1109/CHINACOM.2008.4684993

[14] Fernaz NarinNur, Nazmun Nessa, Narayan Ranjan Chakraborty, A Survey on Routing Protocols in Wireless Multimedia Sensor Networks, July 2013, International Journal of Computer Applications, DOI: 10.5120/12789-0006