Article

Not peer-reviewed version

Digital Battle: A Three-Layer Distributed Simulation Architecture for Heterogeneous Robot System Collaboration

Jialong Gao , Quan Liu , Hao Chen , Hanqiang Deng , Lun Zhang , Lei Sun , Jian Huang *

Posted Date: 20 February 2024

doi: 10.20944/preprints202402.1124.v1

Keywords: heterogeneous robot systems; distributed simulation; communications middle-ware; collaborative mechanism

Article

Digital Battle: A Three-Layer Distributed Simulation Architecture for Heterogeneous Robot System Collaboration

Jialong Gao, Quan Liu, Hao Chen, Hanqiang Deng, Lun Zhang, Lei Sun and Jian Huang *

College of Intelligence Science and Technology, National University of Defense Technology, Changsha 410073, China

* Correspondence: huang_jian@nudt.edu.cn

Abstract: This paper proposed a three-layer distributed simulation network architecture: 1. Distributed virtual simulation network, 2. Perception and control sub-network, and 3. Collaborative communication service network. This simulation architecture runs on a distributed platform. The platform can provide a unique virtual scenario and several simulation services to verify the individual robot system's basic perception, control, and planning algorithms and the distributed collaborative algorithms of heterogeneous multi-robot systems. Furthermore, we designed a simulation experiment scenario for classical heterogeneous robot systems such as unmanned aerial vehicles (UAVs) and unmanned ground vehicles (UGVs). At last, We gave a rough estimation model of the bandwidth.

Keywords: heterogeneous robot systems; distributed simulation; communications middle-ware; collaborative mechanism

1. Introduction

Unmanned and autonomous systems have played an essential role in various fields – whether industrial, civil or military – and experts from these fields have kept considering how to research the various possibilities of Task collaboration within heterogeneous robot systems.

However, before the effective deployment of multi-robot systems to perform complex tasks can be achieved, the research community still faces numerous fundamental technical challenges. These challenges, as listed in [39], encompass areas such as big data, Internet of Things, task complexity, autonomous machine learning, scalability and heterogeneity trade-offs, coalition formation and task allocation, human-in-the-loop, transfer learning, unified frameworks, communication constraints, and connectivity uncertainty. In addition, there are complex application challenges outlined in [38], including adaptive heterogeneous architecture and modeling methods for robot swarm systems, distributed perception and cognition of high-dimensional situations, intelligent decision-making and planning of robot swarm systems that can be guided, trusted, and evolved, as well as autonomous collaborative control of robot swarm systems.

In recent years, significant advancements have been made in the fields of robot operating systems, unmanned system task planning, formation control, ad-hoc network communication, and intelligent algorithms. The collaborative sensing of robot clusters primarily encompasses collaborative positioning technology[40,41], collaborative target recognition, multi-source sensor collaborative optimization, and collaborative target tracking[43]. Intelligent decision-making and planning in cluster systems involve traditional methods such as the Hungarian algorithm, auction algorithm, particle swarm algorithm, and corridor-based methods[42], as well as collaborative task networks constructed using machine learning models[46]. The cooperative control of robot clusters has also seen extensive theoretical research on fundamental modes like consistency control, formation control[45], and formation tracking control[44].

When robots perform complex tasks, whether operating independently or collaborating in clusters, they require a simulation architecture for integrated testing and validation. Distributed simulation

 doi:10.20944/preprints202402.1124.v1

technology, owing to its characteristics, holds the potential to serve as the simulation verification scheme that most closely resembles the actual working environment of heterogeneous multi-robot systems engaged in collaborative task algorithms. Past developments in distributed system technology have encompassed research on synchronization techniques based on message dialogue [47], the implementation of flexible access technologies founded on active services [48], and the provision of reliable operational support guarantees [49]. These advancements have positioned distributed simulation as a viable approach to accurately simulate and verify collaborative task algorithms in robot clusters, facilitating effective evaluation and testing in environments closer to real-world conditions. Through distributed simulation, it becomes possible to validate the collaborative behavior and performance of robots in complex environments, while also assessing the robustness and reliability of robot systems. This contributes significant support and assurance for the rapid advancement and application of robot technology.

Based on the theoretical work of predecessors, we have proposed an open system simulation architecture. In this paper, we will first list some relevant technical achievements that have inspired our work and describe the progress they have made (see Section 2). Next, we will focus on introducing the architecture of the three-layer network simulation verification platform we proposed for implementing collaborative algorithms of heterogeneous robot systems (see Section 3). In the experimental section (see Section 4), we will analyze network characteristics and provide reference suggestions for building similar architectures. Finally, in Section 6, we will offer some shallow insights into the study of collaborative networks.

2. Related Work

Over the past 25 years, robot simulation platforms have evolved almost simultaneously with robotics (see Table 1). Initially, research and development enterprises of robot hardware platforms provided simulation modules that could be integrated into Simulink, enabling users to facilitate secondary development. Meanwhile, the complexity of robot structures increased, leading to a wider range of tasks. As robot diversity expanded, the modeling complexity also grew higher. Consequently, a single module could no longer meet user demands, and independent simulation software began to emerge in collaborations between enterprises and universities [1,2,7,24].

Subsequently, as feature engineering and other multisensor fusion technologies advanced, simulation platforms had to provide not only basic dynamic simulation but also simulate digital signals from multiple sensors, such as visual cameras (even infrared) , LiDAR in [7,12,25] and Sonar in [17], allowing for a comprehensive and realistic representation of the robot's perception capabilities in the virtual environment. Fast forward to the past five years, with the progress of artificial intelligence technology, simulation platforms have found greater utility in providing training samples. This utilization leverages efficient training mechanisms offered by emerging learning algorithms, thus enabling robots to perform better on many nonlinear problems [19,20].

These systems provide platform support for research, learning, testing, and validation of control, perception, navigation, and adversarial tasks. Driven by application requirements, the development of robot simulation technology has been continuously progressing. It has evolved from simulating single objects to multiple heterogeneous objects, from simple static environments to complex environments, and from cooperative collaboration to adversarial games. In addition, the scale of software has been growing increasingly large.

The previous approach of simulating and accessing multiple controllers on a single computing terminal is no longer sufficient to meet the demands. A distributed simulation architecture is needed to address the challenges of scalability. Therefore, our task is to propose a scalable distributed multi-robot collaborative simulation architecture to meet the growing demands for functionality and complexity.

 doi:10.20944/preprints202402.1124.v1

Table 1. A representative series of open source simulation platforms for robot system

First published Year	Institution	Software	Tool	Render	Dynamics	Sensors	Vehicles
1998	Swiss Federal Institute of Technology in Lausanne[2]	webots	OpenGL		ODE	IMU, RGBD	Robots
2001	Simon Fraser University[15]	Stage	FLTK	OpenGL	–	IMU, RGBD	Robots
2007	University of California and University of Pittsburgh[3]	USARSim	Unreal Engine 2.0		ODE	IMU, RGB	Robots
2010	Carnegie Mellon University[16]	OpenRave	OSG		IKFast	IMU, RGB	Robotic arm
2011	Free University of Brussels[21]	ARGoS	Qt-OpenGL		ODE Chipmunk	IMU, RGB	Robots
2012	Universitat Jaume I[17,22]	UWSim	OSG	osgOcean	Bullet	IMU, RGB, Sonar	Underwater robots
2013	Technical University of Darmstadt	Hector	Gazebo(ROS)	OpenGL	ODE	IMU, RGB	Drones
2016	University of Zurich (ETH Zurich)[8]	RotorS	Gazebo(ROS)	OpenGL	ODE	IMU, RGBD	Drones
2016	OpenAI[24]	OpenAI-Gym	Gym		Mujoco[26]	IMU	Multi-joint robot
2017	Inter, Toyota etc.[5]	CARLA	Unreal Engine		PhysX	IMU, RGBD	Ground vehicles
2017	MicroSoft[7]	Airsim	Unreal Engine		PhysX fastsim	IMU, RGBD, Segment,LiDAR	Drones, Ground vehicles
2019	MIT[10,11]	FlightGoggle	Unity		-	IMU, RGBD, Segment	Drones, Ground vehicles
2020	University of Texas and Stanford University[27]	robosuite	mujoco-py	OpenGL	Mujoco	IMU, RGBD	Robots
2020	National University of Defense Science and Technology, Beihang University etc.[13,14]	XTDrone	Gazebo(ROS)	OpenGL	ODE	IMU, RGBD	Drones, Robots
2021	University of Zurich and ETH Zurich[6]	Flightmare	Unity		-	IMU, RGBD, Segment	Drones
2021	Stanford University[23]	IGibson	Unity		-	IMU, RGBD, Segment	Drones
2021	Beihang University and Central South University[4]	Rflysim	Unreal Engine		CopterSim	IMU, RGBD	UAV
2021	Nvidia[1]	Issac Gym	Issac Gym		PhysX	IMU	Bipedal robot, robotic arm, etc.
2022	University of Hong Kong[12]	MARSIM	ROS	OpenGL	-	IMU,LiDAR(HD)	Quadrotor
2023	Institute of Automation Chinese Academy of Sciences[25]	NeuronsGym	Unity3D		-	IMU,LiDAR	Mecanum wheeled robot
2023	Cosys-Lab (FTI) of University of AntwerpChinese Academy[18]	COSYS-Airsim	Unreal Engine		-	IMU,LiDAR,RGB	Drones, Ground vehiclest

Furthermore, the distributed nature of the simulation architecture possesses an inherent advantage: it more accurately reflects the communication characteristics present in real-world working conditions, regardless of whether the multi-robot system employs centralized or distributed collaborative algorithms. This advantage underscores the significance of utilizing a distributed network structure for deploying multi-robot systems.

3. Architecture for Digital Battle

Drawing on the design experience of control and perception modules from related works, and taking into account the coordination of multi-robot systems at the information level as well as the coupling relationships in kinematics, dynamics, and other aspects, we formally propose a three-tiered distributed simulation architecture tailored for heterogeneous multi-robot collaboration.

Taking into consideration the potential complexity associated with the collaborative operation of heterogeneous robot systems, we have adopted three fundamental principles in our design approach:

i The simulation process and outcomes must adhere to the laws of physics, ensuring their consistency across distributed simulation nodes;

ii The capabilities of each robot's motion platform are determined by the structural design, accessory selection, and control implementation;

iii The collaborative algorithms employed in heterogeneous robot systems should be validated through authentic distributed verification methods.

According to these three designing principles, we propose an architecture to organize models and algorithms by three key layers(Figure 1):

 doi:10.20944/preprints202402.1124.v1

(1). Distributed virtual simulation network layer.

It is mainly aimed at maintaining a unified virtual world for every distributed participant. As most Massively Online Role-Playing Games have employed, a popular technical solution is a client-server(C/S) network, where the clients keep a local copy of the virtual world, and the server keeps the authority of the simulation with the responsibility for four services, viz. physical interaction, scenario management, synchronization service and session management .

(2). Perception and control sub-network layer.

We have retained many of the advantages of Airsim's solution[7]. This sub-network connects the virtual avatar with its bonded "brain"(i.e., the one posses the control). The virtual avatar could be a motion platform with actuator and sensor units, while the "brain" could be a mature hardware system, a set of algorithms, or even a person, i.e., the sub-network could work either in the Artificial Intelligence Mode or the Human-in-the Loop Mode.

(3). Collaborative communication service network layer.

As has been proven to be a success in real-time distributed embedded systems, the idea of Data Distributed Service(DDS)[29] is also adopted here to provide a basic information transmission mechanism. It is believed to support most of the possible collaboration between the upper compute node of heterogeneous robot systems.

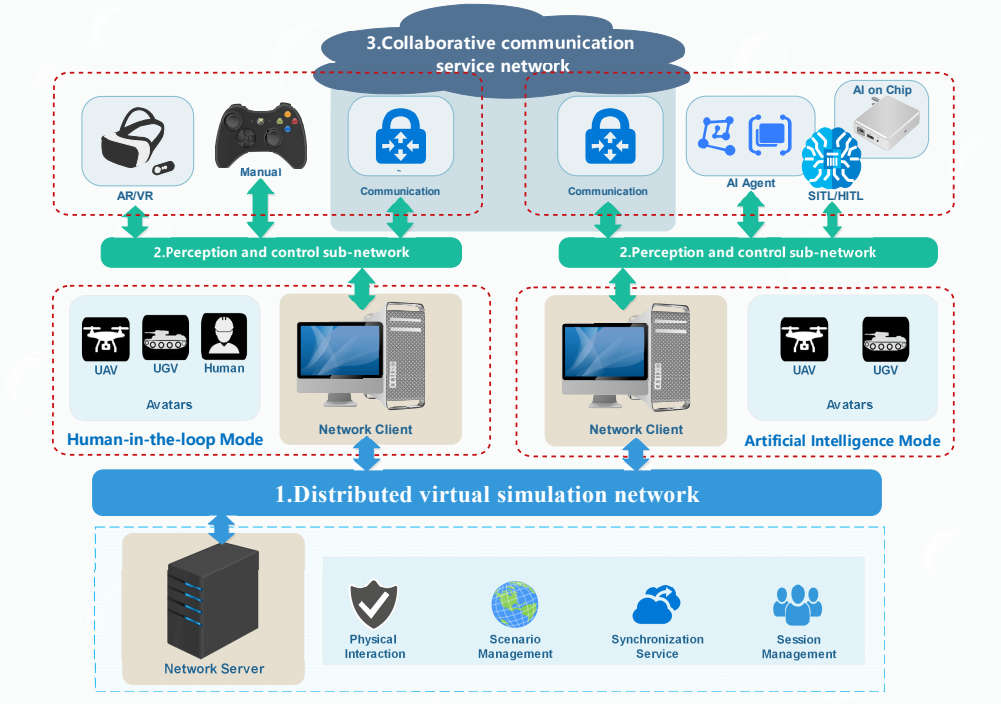

Figure 1. The architecture of the distributed simulation platform with three key layers of network

As a distributed system, the startup process of simulation is a bit cumbersome. There are two primary constraints on the startup order. 1. The Distributed virtual simulation network server should launch before other clients participate. 2. The Perception and control sub-network could only establish the connection between the avatar and the "brain" after the virtual character(i.e., the avatar) has been created in the virtual world. In addition, the Collaborative communication service network could work whenever a "brain" is online.

Next, more details about these three key layers' services will be described.

 doi:10.20944/preprints202402.1124.v1

3.1. Distributed virtual simulation network

To assure all the participants feel like staying in the same virtual world, the Distributed virtual simulation network provides a series of easy-to-use C/S services. Among these services, Session Management only deals with the long-term connection mechanism for virtual clients and servers, while Scenario Management, Physical Interaction and Synchronization Service work together to coordinate processing related to simulation content on distributed platforms.

3.1.1. Session Management

From the perspective of distributed simulation, session management is the functional basis for server-client architecture connection[28]. The essential functions provided mainly include providing port available when creating simulation services on the server side. The client discovers and joins the server side in the network, providing role selection and publicity of virtual robot system objects for specific simulation tasks.

The entire distributed simulation session is divided into three stages: 1. **At beginning**. After creating and joining a simulation session, the server controls all participating clients to enter the their local simulation map synchronously, generate corresponding virtual objects at the starting position set in scenario management and obtain control permissions over them. 2. **During process**. During the simulation, session management provides continuous connection support for data synchronization between the server and the client. This synchronized data contains individual dynamic objects in the scene, as well as robot system entities controlled by the client. 3. **At end**. After the distributed simulation task ends, the clients exit the server-managed session and end all data services. All these features are customization and upgrades on the mechanics of the game engine's multiplayer sessions.

3.1.2. Scenario Management

Like most simulation systems and virtual games, the scenario usually defines a global coordinate system in virtual space(either in a local three-dimensional Cartesian coordinate system or spherical coordinates of the Earth), initializes various entity objects in that coordinate system, and manage every simulate-involved objects, e.g., landscape, buildings, roads, non-player-characters (NPCs), robotics under control and even the wind field.

As in Figure 2, Scenario Management is mainly a matter of storage and calculating. The Difference is that the server updates a global scenario while clients only act on local space.

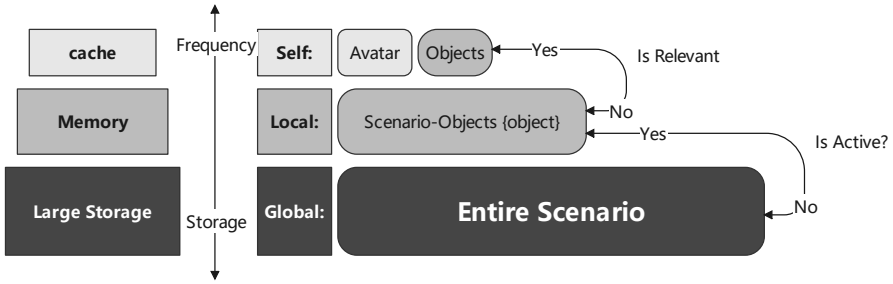

Figure 2. Scenario management plans storage based on local dependencies and activity.

(1).On the server.

Based on the fundamental understanding that the server possesses superior parallel computing and memory read/write capabilities, it is responsible for maintaining global scene information. When receiving a request from a client to update object properties in the scene, the server first checks for potential conflicts, then determines the multicast node list based on the activity level of each client involved in the update, and finally distributes the update results to the respective clients.

 doi:10.20944/preprints202402.1124.v1

(2).On the client.

The client-side scene management loads local object data according to the specific requirements of each client. There are three main categories for handling object data: 1. Objects closely related to the controlled objects are stored in a cache for efficient access and support dynamic calculations at higher frequencies; 2. Objects that are not directly affected by the client, but require awareness, are kept in memory and synchronized with the server's replication commands; 3. Unrelated objects are set to an inactive state and remain stored on the hard disk as a resource.

3.1.3. Physical Interaction

Inspired by Mujoco's state-of-the-art performance in Contact Dynamics[26], we believed it's worthwhile to provide an interaction mechanism to support similar calculation on distributed platform. Commonly, interaction calculation has three key steps:

Step 1.

Since a certain interaction must be triggered by some specific condition, we might as well define the conditional function $\mathbf{C}$ as in formula.(1). For arbitrarily selected objects obj_i and obj_j with generalized state parameters X_i and X_j, the positiveness of function $\mathbf{C}$ can indicate whether the interaction conditions are met. Mathematically, function $\mathbf{C}$ is a binary relationship.

$$c_{i,j} = \mathbf{C}(X_i, X_j) \tag{1}$$

A graph $\mathbf{G} = (V, E)$ can describe the interaction relations within objects. The vertex set $V \triangleq \{obj_i\}$ and the edge set $E = \{e_{i,j}\}$ construct a adjacency matrix, each element $e_{i,j}$,

$$e_{i,j} = \begin{cases} 1, & c_{i,j} >= 0 \\ 0, & else \end{cases} \tag{2}$$

(For those discrete event-driven interactions that can be linearly separated into pairs, skip directly to step 3)

Step 2.

For those relatively complicated cases. Instead of listing various kinds of interaction models in detail, we use standard Hamiltonian systems as a theoretic representative of multi rigid-body dynamics. Suppose there exists a connected sub-graph $G_l = (V_l, E_l)$ (i,e, $V_l \subseteq V$, $E_l \subseteq E$, and the derived distance matrix D_l of G_l, s.t., $\forall d_{i,j} \in D_l$, there is $0 < d_{i,j} \leq l$). This sub-graph G_l illustrates the object set V_l as isolated from other objects by distance l. The edges in E_l and the corresponding constraints determine a tangent bundle (i.e., configuration space) spanned by the objects in V_l, e.g., along with the constraint,

$$\dot{}(\mathbf{p}, \mathbf{q}, t) = \mathbf{0} \tag{3}$$

The constrained forces subject to D'Alembert-Lagrange principle, define the virtual displacements ω and constrained forces $\mathbf{Q}^C$, there would be $\omega^T \mathbf{Q}^C = 0$, and also with the assumption that multi rigid-body problem subjects to the Hamilton's principle of least action.

$$\delta \mathbf{S}(\mathbf{p}, \mathbf{q}, t) \triangleq \delta \int_t^{t+T} (\mathbf{p} \cdot \mathbf{q} - H(\mathbf{p}, \mathbf{q}, t)) dt = 0 \tag{4}$$

The motion equation as cotangent bundle in formula.(5), where $\mathbf{F}^u$ refers to the combined force of the input driving force and the possible dissipative force[33,34,36].

 doi:10.20944/preprints202402.1124.v1

$$\dot{\mathbf{q}} = \frac{\partial H}{\partial \mathbf{p}}, \dot{\mathbf{p}} = -\frac{\partial H}{\partial \mathbf{q}} + \mathbf{Q}^C(\mathbf{p}, \mathbf{q}, \mathbf{t}) + \mathbf{F}^u \tag{5}$$

If the expression in the formula.(5) is separable, the explicit differential expression could be constructed easily[31], but for those implicit issues, the construction method depends on the area of the problem and is beyond the scope of this article.

Step 3.

Suppose we already have an explicit expression pattern formula.(6) for every object involved in the interaction sub-graph $G_l = (V_l, E_l)$. The evolution of the isolated interacted system can be figured out and updated for every distributed client.

$$X_i(t+T) = \mathrm{F}(X_i(t), \mathbf{Q}^C, \mathbf{F}^u, T) \tag{6}$$

3.1.4. Synchronization Service

Synchronization services is the key to maintaining limited consistency for each node in a distributed virtual network. Figure 3 shows two paths for synchronization, namely emergent frame and state update.

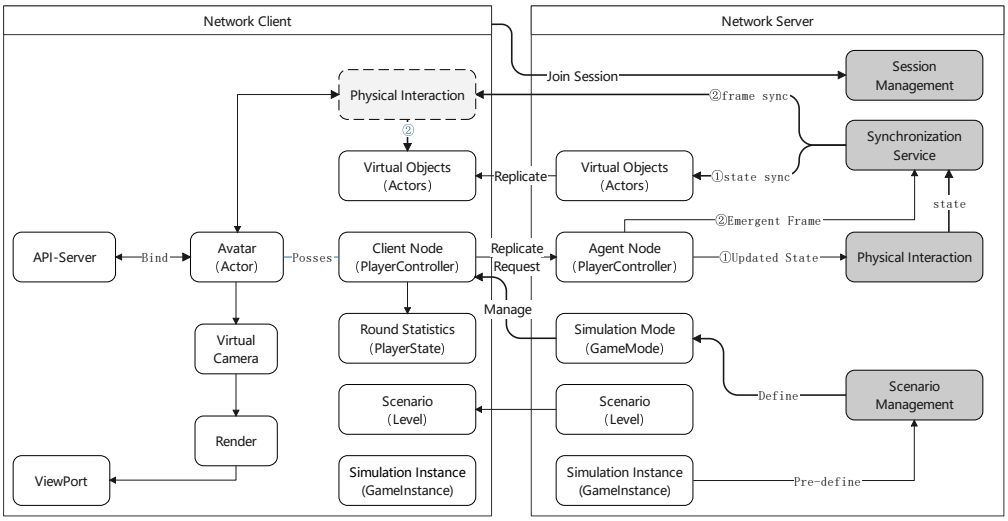

Figure 3. Architecture of the Distributed virtual simulation network.

(1).Emergent frame.

In objects with coupling relationships(e.g., constraints on the state), if there are more than two related clients, emergent frame is the event-driven way to reconstruct the motion model on the related clients. This specific method is based on the remote procedure call (RPC) mechanism, and the server triggers the response of the relevant client to resconstuct the motion model(i.e., the step2 in Physical Interaction).

(2).State update.

It is a cycle-adaptive multicast method. The client will detect whether the properties of the local object change every time interval and send a synchronization request to the server if it changes. The server accepts the request and confirms that it is valid, and will package the object of the application attribute change in the next state synchronization frame and synchronize it to the relevant client according to the multicast list. The selection of time intervals is positively correlated with the number of occurrences in other client local zones, and does not exceed the upper limit allowed by the network. Assuming that all properties with amount D are recognized as needing to be synchronized to other

 doi:10.20944/preprints202402.1124.v1

$n-1$ clients within the synchronization allowable time T_s, the supremum of State update network traffic as stream S_v can be inferred to:

$$\sup S_v = \frac{(n-1)D(n)}{T_s} \tag{7}$$

3.2. *Perception and control sub-network*

The perception and control network is a one-to-one network abstracted from Airsim[7]. Sensor data, actuator signals, perception data and desired state can either use the Mavlink protocol to connect the algorithm hardware and the virtual client through the serial port or use remote procedure call (RPC) locally to realize the interoperability between the algorithm software and the virtual client or access the user through the peripheral of human-computer interaction.

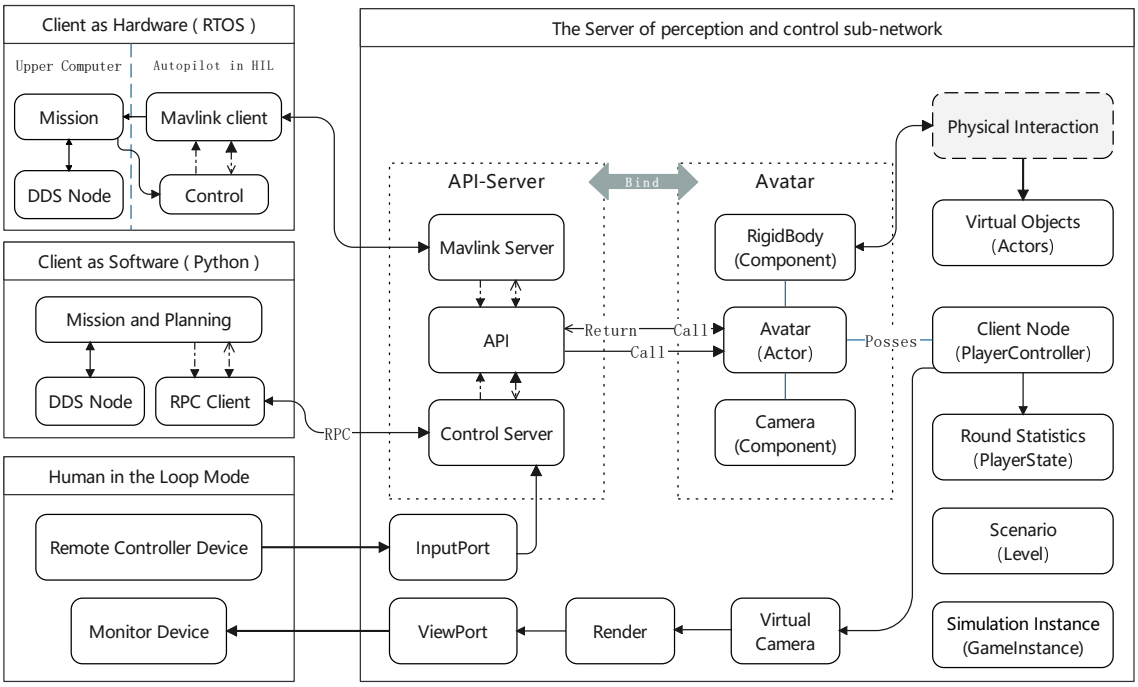

Figure 4. Architecture of the Perception and control sub-network

Unlike the other two-layer networks, the perception and control subnetwork is an independent connection mechanism, which only requires the baud rate of the local hardware interface or the read/write rate of interprocess communication to meet the bandwidth requirements of state data and control instructions.

3.2.1. Artificial Intelligence Mode

The simulation verification subject for this platform's design is the robot system's planning and control algorithm. For clients in artificial intelligence mode, algorithm access methods are mainly divided into hardware-in-the-loop and software-in-the-loop.

Since the update frequency of sensor data and control data is fixed, it may as well be set up m_i the size of the ith data and f_i the corresponding update frequency. Thus, the average traffic on the perception and control sub-network,

$$S_c = \sum_{i=1}^{n} f_i \cdot m_i \tag{8}$$

 doi:10.20944/preprints202402.1124.v1

(1).Hardware-in-the-loop.

Thanks to the contribution of the open-source community, we can directly integrate Mavlink support into the hardware-in-loop and enable autonomous driving hardware to drive the corresponding robot system in the virtual environment. The hardware-in-the-loop is usually transmitted through the serial communication interface at a negotiated baud rate, and the actual effective bandwidth appropriate to the formula.(8).

(2).Software-in-the-loop.

The simulation of software in the loop, based on the inter-process communication mode, connects the input and output of the artificial intelligence algorithm with the request and service form of remote process call, and the artificial intelligence algorithm is responsible for environmental perception, task decision-making, action planning, and specific control node function implementation and organizational framework and is not limited by the three-layer distributed simulation system architecture of this paper, as long as it meets the process communication rate constraints of the local perception-control loop.

3.2.2. Human in the Loop Mode

The human loop mode provides two possibilities for use. One is to directly send the input of the remote control handle of the human as a command signal to the API end of the robot system, which is a form of human-computer interaction with direct control; Second, the client operator exists in the digital space as an avatar, which is aimed at verifying the collaborative mode of human-machine symbiosis.

The direct control mode is mainly connected to the operation control in the serial port mode of the controller simulator, and the human-computer symbiosis mode is realized through OpenXR access to VR or AR human-computer interaction devices.

3.3. Collaborative communication service network

Different from Distributed virtual simulation network's master-slave structure and Perception and control sub-network's one-to-one structure, the collaborative communication service network is a decentralized and completely flat distributed structure. For this design purpose, we use FastDDS as the communication middleware for this layer of the network.

However, from the perspective of simulation construction, it is worth noting that the collaborative communication service network and the Distributed virtual simulation network share the physical bandwidth of the network, but generally speaking, the application's use of bandwidth is closely integrated with the Collaborative problem itself, and we can only give an estimate of the bandwidth usage margin.

Assuming that the actual maximum bandwidth available to the network is W and Ditributed virtual simulation network utilize S_v bandwidth in formula.(7), thus we have,

$$Margin = W - S_v \tag{9}$$

3.3.1. Basic communication mechanism

As in Figure 5, DDS enables all intelligent algorithms of robot systems in a virtual scene to connect to the same publish/subscribe network. In the collaborative data space of the network, each topic can be regarded as a communication channel, and the nodes related to the topic in the network will subscribe to the topic corresponding to the channel and publish collaborative information to it as needed. The specific publishing process is as follows: first, encode the collaborative information according to the standardized JSON format, then serialize it into a string, and finally publish it to

 doi:10.20944/preprints202402.1124.v1

the topic. It is possible to transform a data-centric DDS network into a collaborative task-centred collaborative network through the above mechanism.

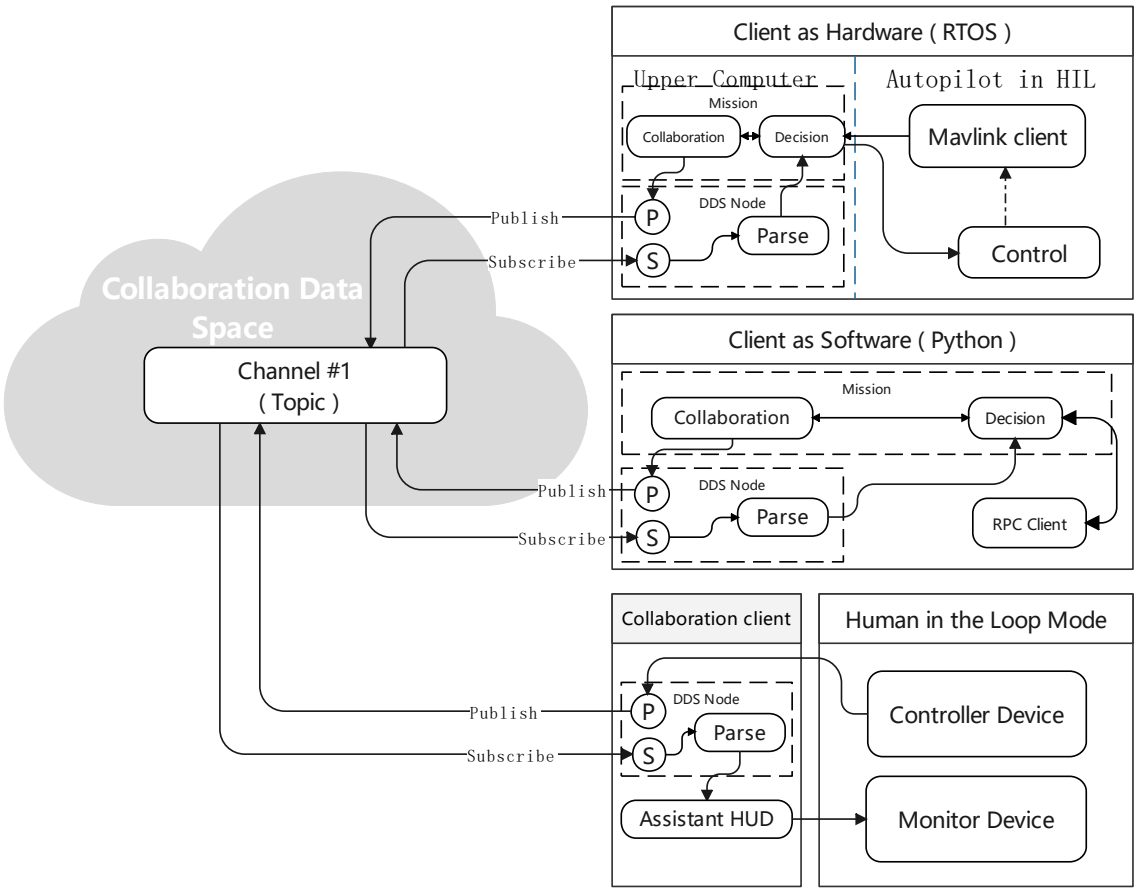

Figure 5. Architecture of the Collaborative communication service network

3.3.2. Distributed collaboration process

This article provides an interop protocol process based on the above communication mechanism. Think of a collaborative task simply as a diagram of activity(Figure 6) between two entities that request and respond together(Another work in [37]). The Public Topic refers to the channel that all intelligent systems will subscribe to, mainly for requests and responses for collaborative tasks. When the requester and responder of a task match, create a dedicated task channel for continuous collaborative communication requirements.

 doi:10.20944/preprints202402.1124.v1

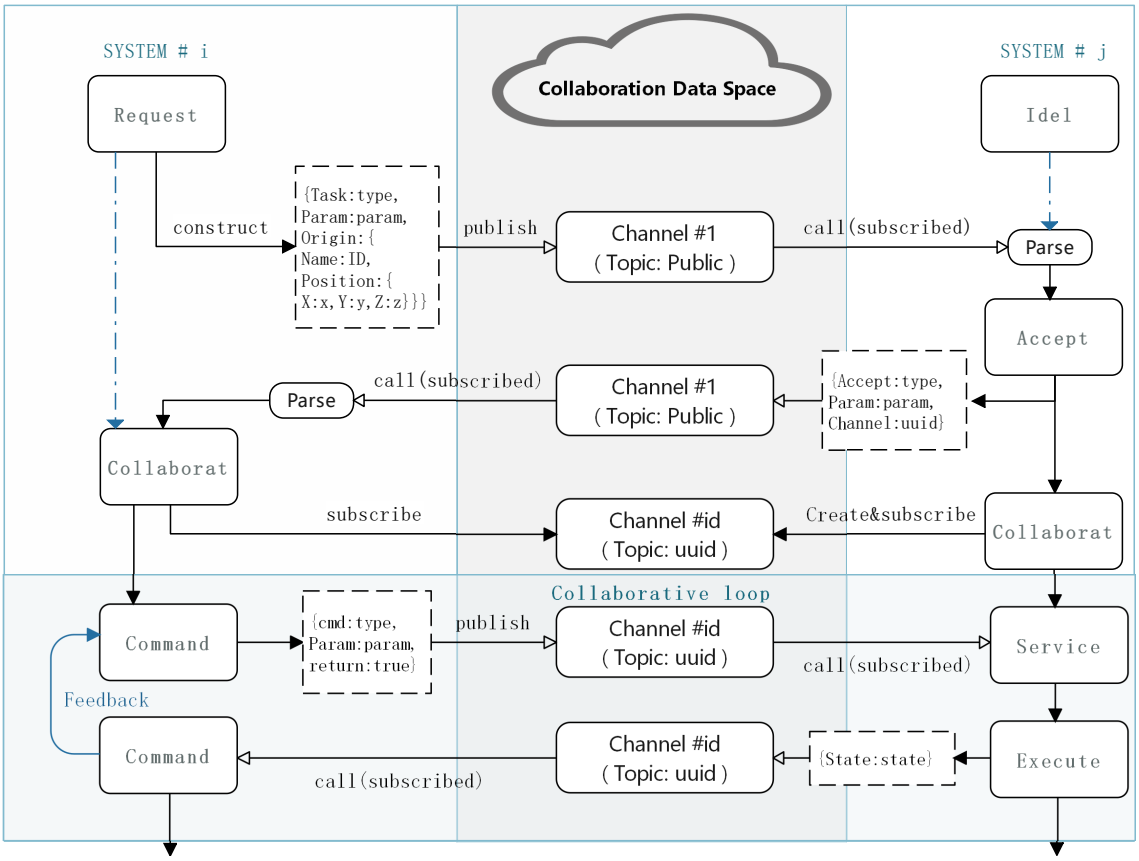

Figure 6. A diagram of activity for Distributed collaboration process.

In some special cases, it may not be necessary to form a dedicated topic channel, and when the coordination requirements are matched and confirmed, the task parameters will be sufficient to guide the cooperative system *SYSTEMj* to complete the corresponding operation.

4. Example and Experiments

Based on the three-layer distributed simulation architecture, the performance test process is roughly divided into four steps:

1. Design a virtual scene, define the coordinate system in the scene, place buildings and roads, and set the initial generation position of the robot system;
2. Connect the computing platforms to the network, and specify a server and multiple clients ammong them, as well as set the corresponding avatar(UGV or UAV).
3. Start the simulation service and record the network stream frames at the same time with an improved Network-Profiler;
4. Analyze the distributed data service and compare the theoretical estimation.

Note: Any entity or logical object will be instantiated as a subclass of the Actor when the experiment is implemented. Therefore, actors will be used as a general term when analyzing and describing results.

4.1. Collaborative tasks

The cyclic execution process of two typical types of collaborative tasks is as follows:

 doi:10.20944/preprints202402.1124.v1

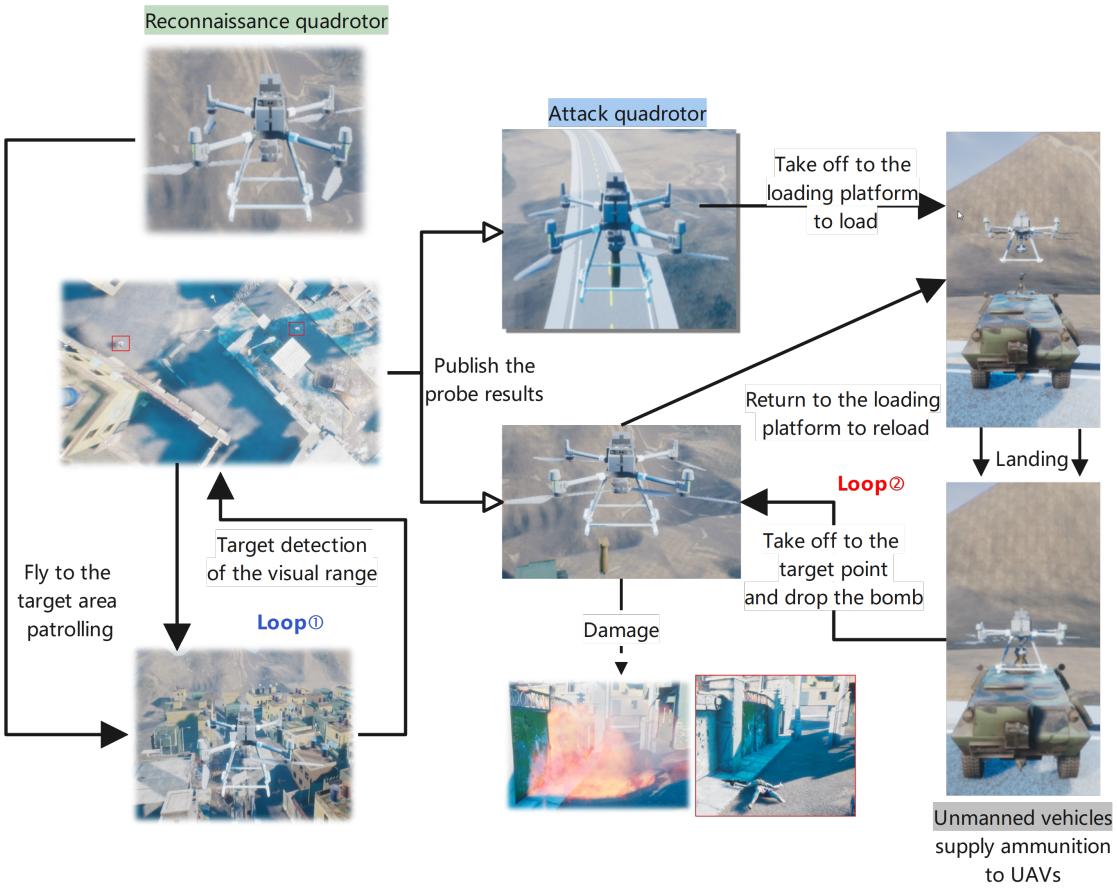

Figure 7. The cyclic execution process of two typical types of collaborative tasks. Loop ①: The cycle of "target detection - bomb strike" between the reconnaissance quadcopter and the attack quadcopter; Loop ②: The cycle of "bomb attack - ammunition supply" between attack quadcopter and armored unmanned vehicles.

(1).Target Detection - Bomb Strike Loop:

Within this loop, the reconnaissance quadcopter is responsible for detecting potential targets, and once a target is confirmed, the attack quadcopter executes the bomb strike action. The reconnaissance quadcopter employs its sensor system to identify and locate the target, and subsequently communicates the relevant information to the attack quadcopter. Upon receiving the information, the attack quadcopter promptly responds, launching a strike on the target and carrying out the bomb attack.

(2).Bomb Attack - Ammunition Supply Loop:

In this loop, after executing a bomb strike, the attack quadcopter requires replenishment of ammunition in order to continue its mission. The armored unmanned vehicle serves as the provider of ammunition, transporting the required ammunition to a designated location based on the needs of the attack quadcopter. Once the ammunition resupply is completed, the attack quadcopter can proceed with the next round of bomb strikes.

Through the implementation of such cyclic execution processes, effective coordination between the reconnaissance quadcopter and the attack quadcopter, as well as between the attack quadcopter and the armored unmanned vehicle, is achieved, facilitating efficient and synchronized combat operations.

4.2. Sampling software

In this subsection, we will explore a valuable addition to the software: a new feature module that enables users to output network traffic and performance information in batch to JSON files. Although

 doi:10.20944/preprints202402.1124.v1

this enhancement does not change the usage method, it greatly simplifies subsequent data processing, making analysis and optimization more efficient.

Figure 8. The Network Profiler Interface for Data Analysis and transforming the file format

(1).Introduction to Relevant Functions:

Network Profiler is an independent tool designed to display network traffic and performance information captured by the engine during game runtime. This tool is highly effective in identifying areas of high bandwidth consumption in multiplayer games, allowing users to view the percentage of bandwidth occupied by various actors, PRCs, and attributes, thereby assisting in optimizing network performance.

(2).Recording and Analyzing Sessions:

Before using Network Profiler, users need to record the relevant data for analysis. Recording can be achieved by enabling the engine's process tracking feature, typically by compiling the engine into a debug build or using an editor build for non-debug configurations. Additionally, users can control the recording of network data by adding the command line parameter "networkprofiler=true" at engine startup or using other command line instructions.

The recorded data will be saved to the specified path, allowing users to open the file in the standalone tool for further analysis. It is important to note that temporary files will be renamed according to a specific scheme when the analysis session ends, facilitating accurate tracking and data processing.

5. Dicussion

5.1. Actor-synchronous feature

This part of the experiment mainly examines different actors' actual replication frequency and replication time during synchronization. One UAV and two UGVs were set up for each of the three clients, and non-player characters were also set up in the scene that could track moving actors. The drone just get the hovering instruction while UGV are driven along the road by joystick.

The results of the data visualization are shown in the figure(Figure 9). The plot records the distribution box plot of the replication frequency of each actor that triggered the overall data change of

 doi:10.20944/preprints202402.1124.v1

the simulation at the beginning, process, and end of the simulation. Since the drone is only controlled to hover, the state change is moderated, and the update frequency is slow(merely 10 Hz); The UGVs are constantly moving, so the update frequency is high(always more than 75Hz); Non-player characters are stationary or patrolling for an period because they cannot see moving vehicles, so the median is very close to the upper quartile(about 50Hz). In addition, Object name sorting is performed from smallest to largest amount of data synchronized throughout the simulation.

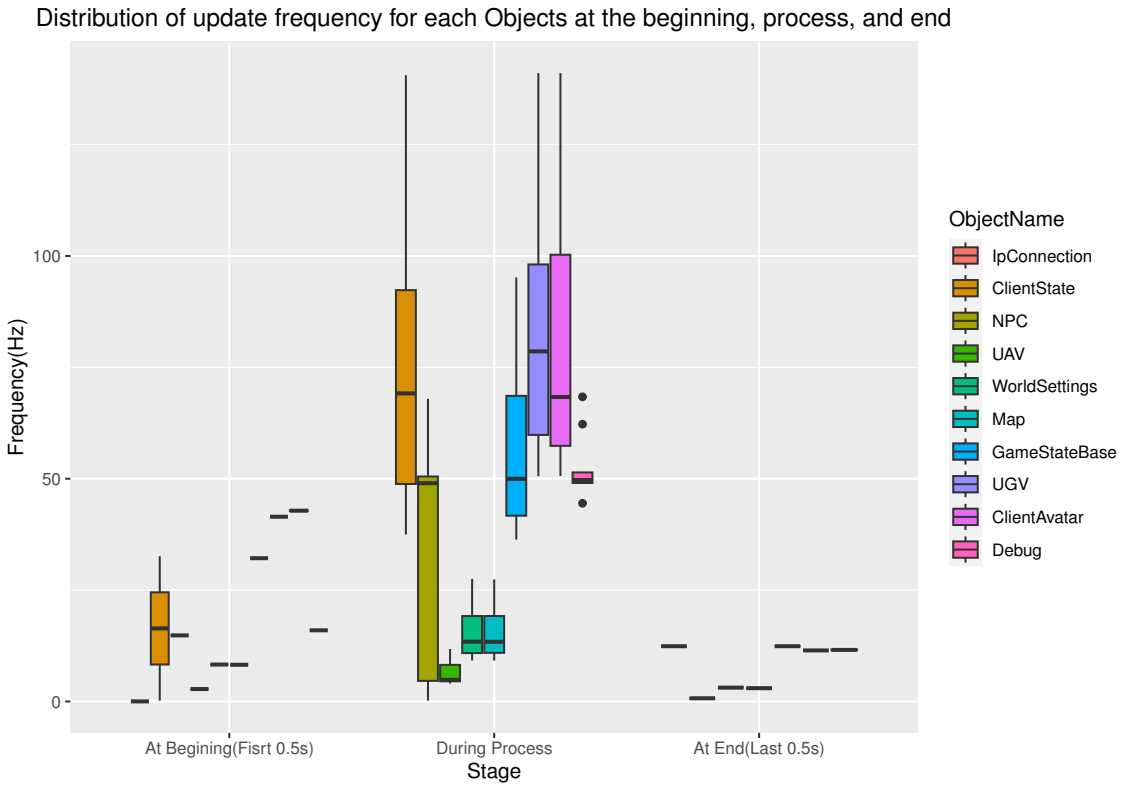

Figure 9. The replication(update) frequency for each Actor varies at the beginning, process, and end. Among all synchronized actors, their update frequencies are within the range of 0 to 150Hz.

For the result data of the same experiment, we also analyzed the replication time distribution when different actor states are synchronously managed to replicate to remote clients, and the visualization results are shown in the Ridge-line plot(Figure 10). Actors representing motion entities have a shorter replication time overhead because their state descriptions are relatively single and are usually state vectors of rigid bodies. They can usually complete the replication of states within 0.03 milliseconds, which can meet the time interval allowed for frame synchronization. The peaks in distribution roughly indicating several independently updated states or events for that actor.

 doi:10.20944/preprints202402.1124.v1

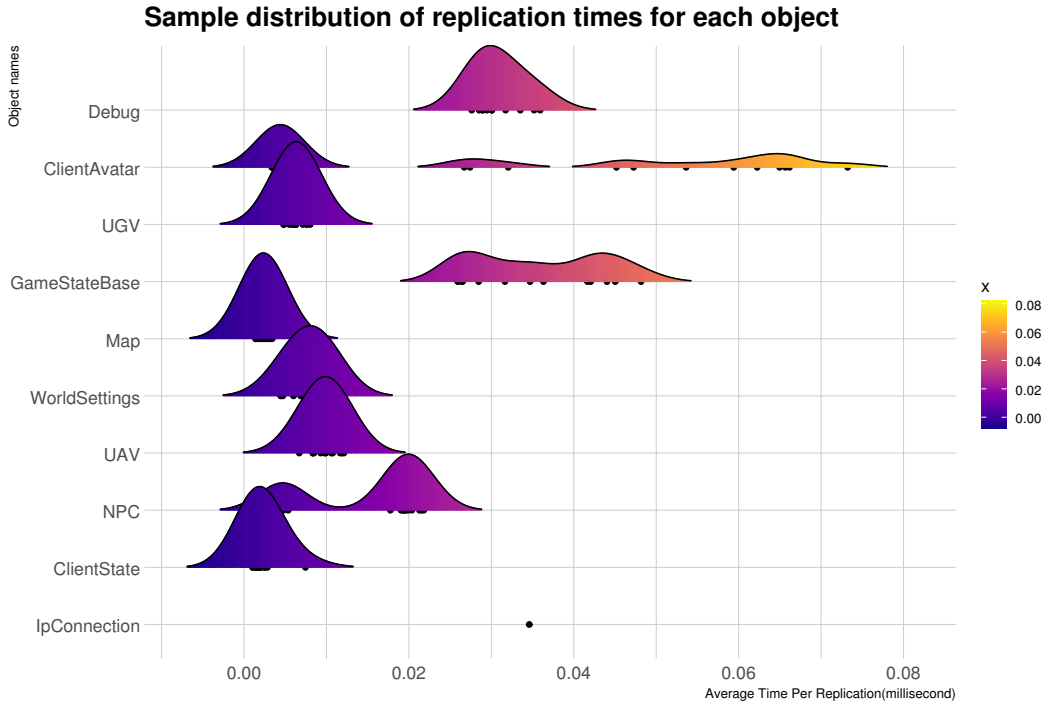

Figure 10. Replication time ridge plot of different actor states synchronized. The replication time varies from 0 to 0.08 milliseconds

5.2. Time domain analysis

In this part of the experiment, the goal is to test the data bandwidth and total throughput of as many clients as possible based on existing lab resources and to examine the correlation between data frames and state data. We built a server (Intel i9 with 32GB Memory) and invited 12 students to participate. They provided their laptops as clients(Computer models and configurations are diverse) and wisdom to design collaborative task, and we finally formed 15 distributed client nodes (some students single-machine multi-open clients) to complete the distributed simulation experimental process. The duration of the simulation is less than 240 seconds.

Figure 11(A) provides an intuitive picture of the bandwidth occupied by the data frame, where the packet is initially used to complete the configuration work required for initialization on the distributed network. The data frame monolith is significant, but the synchronization frequency is relatively low. In the intermediate process, the data frame is mainly used to synchronize the dynamic properties of the actor between the various clients, so its monolith is small, but it is updated relatively frequently; At the end of the simulation, the number of entities is reduced, and the bandwidth consumption of synchronized data frame rate is also reduced. The above situation corresponds to the stage characteristics shown in Figure 12. In addition, the median bandwidth occupied by data frames is 8KB/s, and the maximum bandwidth usage is about 25KB/s. The results of Figure 11(B) reveal a positive correlation between the size of the data frame and the size of the attribute data in the frame. Therefore, we can use a rough estimate of the amount of data that needs to be synchronized. And the results of (B) can be verified bidirectionally by Figure 11(C), and the bandwidth occupied by attribute synchronization has been fluctuating below 50% but occasionally exceeding 87.5% throughout the simulation time, which can confirm the coefficient of 2.13 obtained by linear fitting $y = 0.0096 + 2.13x$ in Figure 11(B).

 doi:10.20944/preprints202402.1124.v1

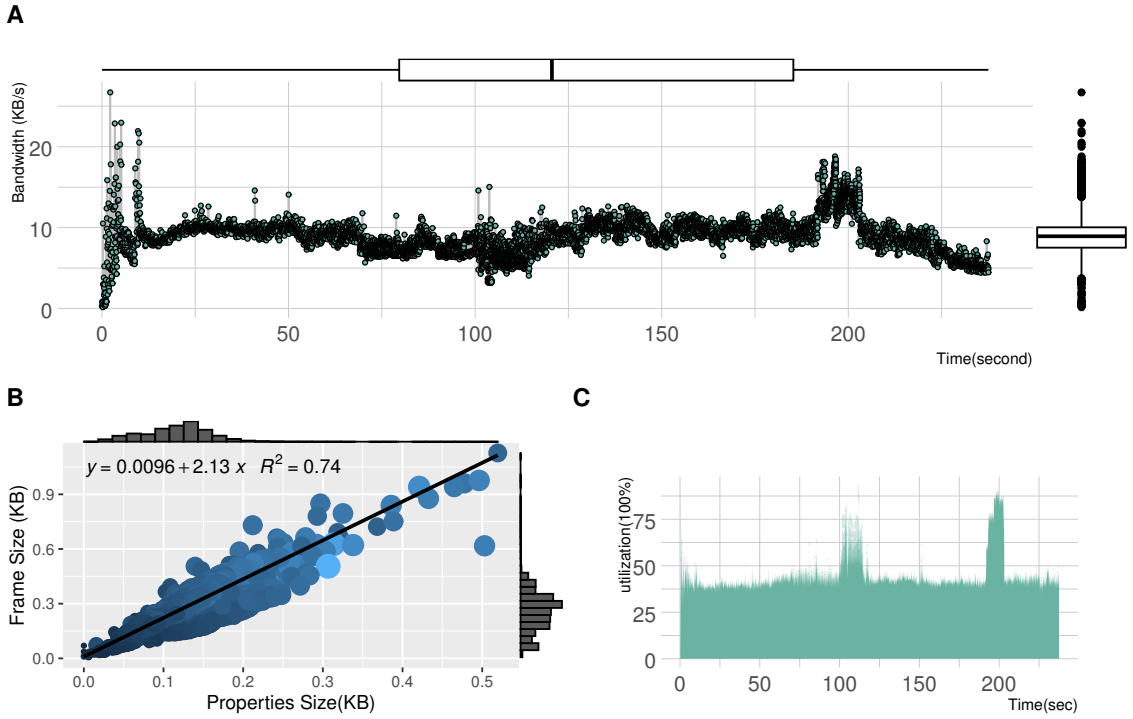

Figure 11. (A).Bandwidth distribution over the time domain; (B).The size correlation between the frame and the properties data; (C).Time distribution of attribute data on bandwidth usage

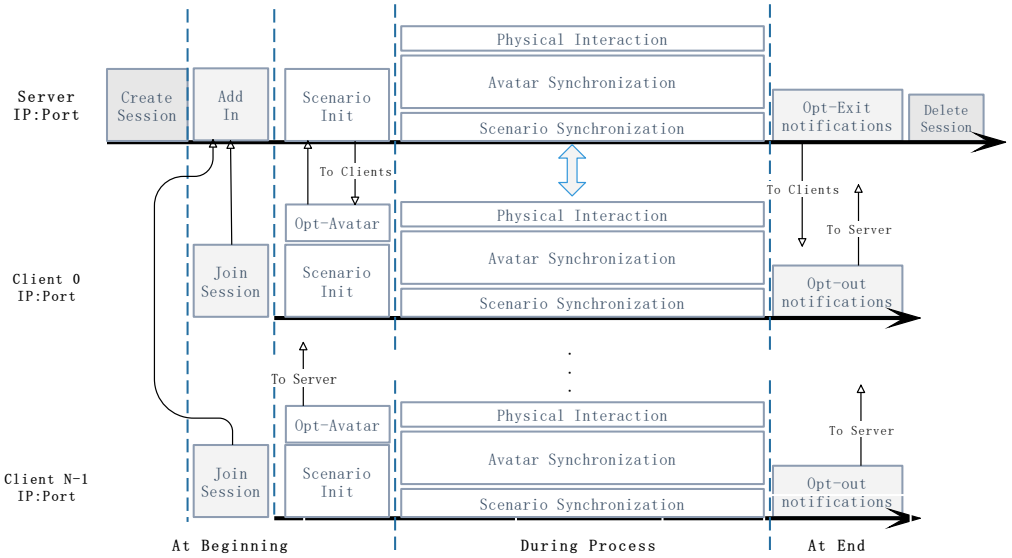

Figure 12. The entire distributed simulation session. The session mainly organizes the server and clients in long-term connections and automatically delivers data for Scenario Synchronization, Avatar Synchronization and Physical Interaction.

At the same time, we also made a chord graph of the total data throughput of the server and 15 clients in Figure 13. Through comparison, it can be found that the data throughput on the server side reaches almost half of the total throughput, and the throughput of each client is different. By observing the total amount of data input and output of each end, it can also be found that the input amount of

 doi:10.20944/preprints202402.1124.v1

the client is slightly greater than the output, while the output of the server side is slightly larger than the input.

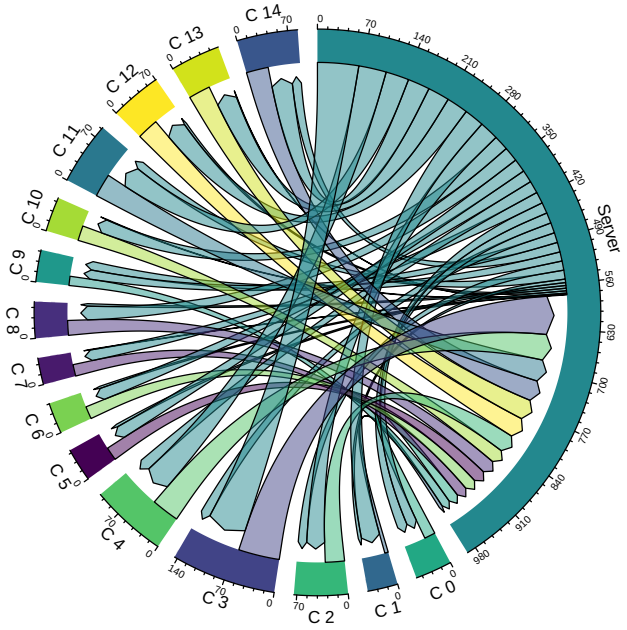

Figure 13. A Chord diagram for throughput within the server and 15 clients.

5.3. Scale and bandwidth

Based on several experiments, we analyze the relationship between the physical object scale of the robot system (equivalent to the client node scale of the distributed virtual simulation network within this framework) and the communication bandwidth in the network.

In the process of data analysis, because the theoretical value of maximum bandwidth is almost impossible due to the existence of synchronization strategy, but assuming that the distribution of bandwidth has specific mean point characteristics, we select the mean bandwidth as the fitted sample point to obtain the distribution relationship of bandwidth with the scale of clients and the total number of attribute values as shown in the Figure 14.

$$S_{mean} = a \cdot n \cdot D_n - b \cdot n + c \cdot D_n + d \tag{10}$$

Among the parametre, a stands for average size of each properties, b represents the average output of each client; c is a relative ratio of the properties volumn maintained by the server, and d could be some necessary command and debug data. As a numeric result we got $a = 0.002083$KB/s, $b = 0.1818$KB/s, $c = 0.013199$KB/s, $d = 2.257655$KB/s. Compared with the form of the Formula.7, we can derive a similar form based on the formula.(10).

$$S = 0.002083 \cdot n(D_n - 87) + 0.013199 D_n + 2.257655 \tag{11}$$

Thus, its' parameters can be symbolized as ,

$$S = p_u \cdot n(D_n - \bar{D}) + cD_n + d \tag{12}$$

Where the p_u represents property replication traffic related to client scale, $\bar{D}$ is the part of the property that is independent of the number of clients.

 doi:10.20944/preprints202402.1124.v1

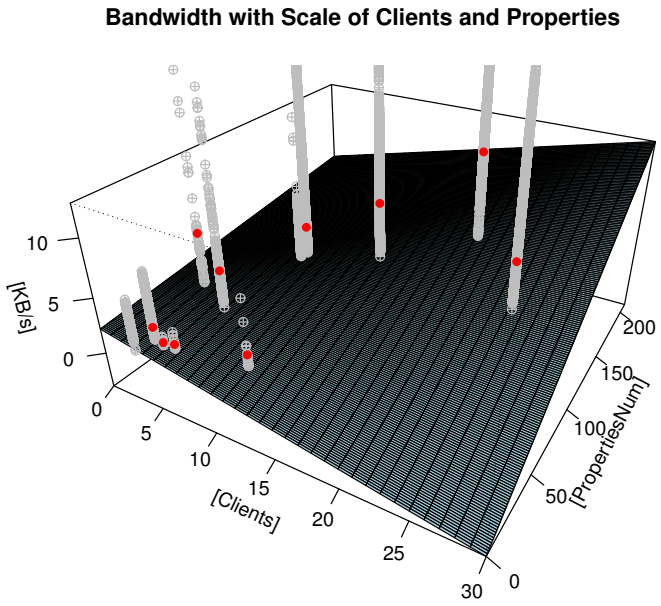

Figure 14. The bandwidth fits with scale of clients and properties. The grey points refer to the bandwidth from experiment data grouped by the scale of Clients and Properties, while the red points are the mean value in that summaries of each group. And the light-blue surface fit the red points as formula.10

6. Conclusion and Future Work

In the current work, we propose a three-layer distributed network architecture according to the simulation requirements of multi-robot collaborative tasks and build a verification platform with models accessed from the open-source community. These three layers are,

1. The distributed virtual simulation network based on a game engine;
2. An end-to-end Perception and control sub-network;
3. The Collaborative communication service network based on distributed data service.

At the experiment section. We design a certain scenario and place some simple UAVs and UGVs with their controller and collaboration algorithms, and finally analyze the network data traffic of the platform,

1. Obtain the characteristics of positive correlation between attribute update frequency and object dynamics;
2. The distribution of network bandwidth occupation in simulation time;
3. Roughly verify the estimation model(formula.7) of the maximum bandwidth demand of the overall network.

In future research, we will improve the following key technical details based on this distributed simulation framework,

1. Although the dynamic model update method of coupled objects has been given in this architecture, efforts are still needed to further refine the method to support the simulation theory of the system coupling relationship between objects.;
2. The synchronous management adopted in this architecture has already partially improved the communication delay problem, it is still necessary to study the influence of the spatiotemporal consistency problem of distributed system on multi-robot control and decision-making.;

 doi:10.20944/preprints202402.1124.v1

3. Because the default collaborative network in this study is a simple event-based communication mechanism, only integrates data services and designs and implements the basic collaboration framework, and does not consider its impact on the overall distributed network communication bandwidth, the in-depth study of the collaborative framework will be further promoted in the future in combination with the specific tasks of multi-robot system collaborative work.

7. Patents

This section describes a distributed simulation method and system for multi-UAV (Unmanned Aerial Vehicle) systems. The method decomposes the distributed simulation tasks of multi-UAV systems into three logical concepts: virtual simulation, control perception, and task coordination. The distributed virtual simulation network, perception control subnetwork, and DDS-based collaborative communication service network are established accordingly. The virtual simulation tasks for multi-UAV systems are deployed on distributed nodes. The comprehensive server in the distributed virtual simulation network provides services such as rule arbitration, map scene management, real-time state synchronization, multi-point session management, intelligent simulation, and real-time situational awareness. The client in the distributed virtual simulation network provides local computing capabilities including dynamics, collision interaction, damage simulation, and graphic rendering. The system is designed to implement the above-described method. This invention offers advantages such as simplicity, wide applicability, and high integration.

Classification: H04L41/042; H04L47/783; H04L9/40; H04L67/08; H04L67/59 Application Number: 202310416953.5

Author Contributions: Conceptualization, J.G. and J.H.; methodology and writing—original draft, J.G.; writing—review and editing, Q.L., H.C., H.D., L.Z., L.S. and J.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: While developing this architecture and the corresponding platform, the open-source community provides the model objects, and all enthusiasts contribute to robot systems that lay a good foundation for the design and implementation of our architecture. We would also thank Professor Huang for allowing us to conduct system performance testing in the multi-robot system simulation course, as well as to the 12 students who provided the hardware equipments and took the time to complete the test.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Makoviychuk, V.; Wawrzyniak, L.; Guo, Y.; Lu, M.; Storey, K.; Macklin, M.; Hoeller, D.; Rudin, N.; Allshire, A.; Handa, A.; et al. Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning. *ArXiv* **2021**, *abs/2108.10470*.
2. Michel, O. Cyberbotics Ltd. Webots™: Professional Mobile Robot Simulation. *International Journal of Advanced Robotic Systems* **2004**, *1*, 5, [https://doi.org/10.5772/5618].
3. Carpin, S.; Lewis, M.; Wang, J.; Balakirsky, S.; Scrapper, C. USARSim: a robot simulator for research and education. In Proceedings of the Proceedings 2007 IEEE International Conference on Robotics and Automation, 2007, pp. 1400–1405. https://doi.org/10.1109/ROBOT.2007.363180.
4. Wang, S.; Dai, X.; Ke, C.; Quan, Q. RflySim: A Rapid Multicopter Development Platform for Education and Research Based on Pixhawk and MATLAB. In Proceedings of the 2021 International Conference on Unmanned Aircraft Systems (ICUAS), 06 2021, pp. 1587–1594. https://doi.org/10.1109/ICUAS51884.2021.9476786.
5. Dosovitskiy, A. CARLA:An Open Urban Driving Simulator. In Proceedings of the Conference on Robot Learning, 2017.

 doi:10.20944/preprints202402.1124.v1

6. Song, Y.; Naji, S.; Kaufmann, E.; Loquercio, A.; Scaramuzza, D. Flightmare: A Flexible Quadrotor Simulator. In Proceedings of the Conference on Robot Learning, 2020.
7. Shah, S.; Dey, D.; Lovett, C.; Kapoor, A. Airsim: High fidelity visual and physical simulation for autonomous vehicles. In Proceedings of the Field and Service Robot, 2018, pp. 621–635.
8. Furrer, F.; Burri, M.; Achtelik, M.; Siegwart, R. *Rotors—a modular gazebo mav simulator framework*; Springer, 2016; pp. 595–625.
9. Koenig, N.; dAndrew Howard. Design and Use Paradigms for Gazebo An Open-Source Multi-Robot Simulator. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004, pp. 2149–2154.
10. Guerra, W.; Tal, E.; Murali, V.; Ryou, G.; Karaman, S. FlightGoggles: Photorealistic Sensor Simulation for Perception-driven Robotics using Photogrammetry and Virtual Reality. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE Press, 2019, p. 6941–6948. https://doi.org/10.1109/IROS40897.2019.8968116.
11. Guerra, W.; Tal, E.; Murali, V.; Ryou, G.; Karaman, S. FlightGoggles: A Modular Framework for Photorealistic Camera, Exteroceptive Sensor, and Dynamics Simulation. *ArXiv* **2021**, [arXiv:1905.11377v2].
12. Kong, F.; Liu, X.; Tang, B.; Lin, J.; Ren, Y.; Cai, Y.; Zhu, F.; Chen, N.; Zhang, F. MARSIM: A Light-Weight Point-Realistic Simulator for LiDAR-Based UAVs. *IEEE Robotics and Automation Letters* **2022**, *8*, 2954–2961.
13. Xiao, K.; Tan, S.; Wang, G.; An, X.; Wang, X.; Wang, X. XTDrone: A Customizable Multi-rotor UAVs Simulation Platform. *ArXiv* **2020**, [arXiv:2003.09700].
14. Xiao, K.; Ma, L.; Tan, S.; Cong, Y.; Wang, X. Implementation of UAV Coordination Based on a Hierarchical Multi-UAV Simulation Platform. *ArXiv* **2020**, [arXiv:2005.01125].
15. Vaughan, R.T. Massively multi-robot simulation in stage. *Swarm Intelligence* **2008**, *2*, 189–208.
16. Diankov, R. Automated Construction of Robotic Manipulation Programs. PhD thesis, Carnegie Mellon University, Robotics Institute, 2010.
17. Gwon, D.H.; Kim, J.; Kim, M.H.; Park, H.G.; Kim, T.Y.; Kim, A. Development of a side scan sonar module for the underwater simulator. In Proceedings of the 2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), 2017, pp. 662–665. https://doi.org/10.1109/URAI.2017.7992789.
18. Jansen, W.; Verreycken, E.; Schenck, A.; Blanquart, J.E.; Verhulst, C.; Huebel, N.; Steckel, J. COSYS-AIRSIM: A Real-Time Simulation Framework Expanded for Complex Industrial Applications. In Proceedings of the 2023 Annual Modeling and Simulation Conference (ANNSIM), 2023, pp. 37–48.
19. Kumar, A.; Li, Z.; Zeng, J.; Pathak, D.; Sreenath, K.; Malik, J. Adapting Rapid Motor Adaptation for Bipedal Robots. In Proceedings of the 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2022, pp. 1161–1168. https://doi.org/10.1109/IROS47612.2022.9981091.
20. Jiang, Y.; Zhang, T.; Ho, D.; Bai, Y.; Liu, C.K.; Levine, S.; Tan, J. SimGAN: Hybrid Simulator Identification for Domain Adaptation via Adversarial Reinforcement Learning. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), 2021, pp. 2884–2890. https://doi.org/10.1109/ICRA48506.2021.9561731.
21. Pinciroli, C.; Trianni, V.; O'Grady, R.; Pini, G.; Brutschy, A.; Brambilla, M.; Mathews, N.; Ferrante, E.; Di Caro, G.; Ducatelle, F.; et al. ARGoS: A modular, multi-engine simulator for heterogeneous swarm robotics. In Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2011, pp. 5027–5034. https://doi.org/10.1109/IROS.2011.6094829.
22. Prats, M.; Pérez Soler, J.; Fernandez, J.; Sanz, P. An open source tool for simulation and supervision of underwater intervention missions. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, 10 2012, pp. 2577–2582. https://doi.org/10.1109/IROS.2012.6385788.
23. Li, C.; Xia, F.; Martin-Martin, R.; Lingelbach, M.; Srivastava, S.; Shen, B.; Vainio, K.; Gokmen, C.; Dharan, G.; Jain, T.; et al. IGibson 2.0: Object-Centric Simulation for Robot Learning of Everyday Household Tasks. *ArXiv* **2021**.
24. Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; Zaremba, W. OpenAI Gym. *ArXiv* **2016**, *abs/1606.01540*.
25. Li, H.; Liu, S.; Ma, M.; Hu, G.; Chen, Y.; Zhao, D. NeuronsGym: A Hybrid Framework and Benchmark for Robot Tasks with Sim2Real Policy Learning. *ArXiv* **2023**, *abs/2302.03385*.

26. Todorov, E.; Erez, T.; Tassa, Y. MuJoCo: A physics engine for model-based control. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2012, pp. 5026–5033. https://doi.org/10.1109/IROS.2012.6386109.
27. Zhu, Y.; Wong, J.; Mandlekar, A.; Mart'in-Mart'in, R. robosuite: A Modular Simulation Framework and Benchmark for Robot Learning. *ArXiv* **2020**, *abs/2009.12293*.
28. Statzer, J.M. AdvancedSessionsPlugin. https://github.com/mordentral/AdvancedSessionsPlugin, 2017.
29. Schlesselman, J.; Pardo-Castellote, G.; Farabaugh, B. OMG data-distribution service (DDS): architectural update. In Proceedings of the IEEE MILCOM 2004. Military Communications Conference, 2004., 2004, Vol. 2, pp. 961–967 Vol. 2. https://doi.org/10.1109/MILCOM.2004.1494965.
30. Müller, M.; Macklin, M.; Chentanez, N.; Jeschke, S.; Kim, T.Y. Detailed Rigid Body Simulation with Extended Position Based Dynamics. *Computer Graphics Forum* **2020**, *39*.
31. Kang Feng, Q.M. *Symplectic Geometric Algorithms for Hamiltonian Systems*; Zhejiang Science and Technology Press, 2003.
32. Wang, K.; Ju, H. An explicit modelling method of joint-space inertia matrix for tree-chain dynamic system. *International Journal of Non-Linear Mechanics* **2022**, *144*, 104033. https://doi.org/https://doi.org/10.1016/j.ijnonlinmec.2022.104033.
33. Huang, Z.; Chen, J.; Zhang, Z.; Tian, Q. Lie group variational integral algorithm for multi-rigid body dynamics simulation(in Chinese). *Jounral of Dynamics and Control* **2022**, *20*, 8. https://doi.org/10.6052/1672-6553-2021-021.
34. A. V. Borisov, I.S.M., 1 Rigid Body Equations of Motion and Their Integration. In *Rigid Body Dynamics Hamiltonian Methods, Integrability, Chaos*; HIGHER EDUCATION PRESS: Beijing, 2018; pp. 19 – 78.
35. Udwadia, F.E. Constrained Motion of Hamiltonian Systems. *Nonlinear Dynamics* **2015**. https://doi.org/10.1007/s1071-015-2558-3.
36. Celledoni, E.; Leone, A.; Murari, D.; Owren, B. Learning Hamiltonians of constrained mechanical systems. *Journal of Computational and Applied Mathematics* **2022**, *417*, 114608. https://doi.org/10.1016/j.cam.2022.114608.
37. Deng, H.; Huang, J.; Liu, Q.; Zhou, C.; Gao, J. BGSD: A SBERT and GAT-based Service Discovery Framework for heterogeneous distributed IoT. *Computer Networks* **2022**. https://doi.org/10.1016/j.comnet.2022.109488.
38. Xiang, J.; Dong, X.; Ding, W.; Suo, J.; Shen, L.; Xia, H. Key technologies for autonomous cooperation of unmanned swarm systems in complex environments. *Acta Aeronautica et Astronautica Sinica* **2022**, *43*. https://doi.org/10.7527/S10006893.2022.27570.
39. Rizk, Y.; Awad, M.; Tunstel, E.W. Cooperative Heterogeneous Multi-Robot Systems: A Survey. *ACM Comput. Surv.* **2019**, *52*. https://doi.org/10.1145/3303848.
40. Ding, Y.; Xiong, Z.; Xiong, J.; Cui, Y.; Cao, Z. OGI-SLAM2:A hybrid map SLAM framework grounded in inertial-based SLAM. *IEEE Transactions on Instrumentation and Measurement* **2022**, *PP*, 1–1. https://doi.org/10.1109/TIM.2022.3209726.
41. Xun, Z.; Huang, J.; Li, Z.; Xu, C.; Gao, F.; Cao, Y. CREPES: Cooperative RElative Pose EStimation towards Real-World Multi-Robot Systems. *ArXiv* **2023**. https://doi.org/10.48550/arXiv.2302.01036.
42. Wang, Z.; Xu, C.; Gao, F. Robust Trajectory Planning for Spatial-Temporal Multi-Drone Coordination in Large Scenes. *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* **2021**, pp. 12182–12188.
43. Wang, J.; Wu, Y.; Chen, Y.; Ju, S. Multi-UAVs collaborative tracking of moving target with maximized visibility in Urban Environment. *Journal of the Franklin Institute* **2022**, *359*. https://doi.org/10.1016/j.jfranklin.2022.05.004.
44. Li, W.; Zhang, H.; Gao, Z.; Wang, Y.; Sun, J. Fully Distributed Event/Self-Triggered Bipartite Output Formation-Containment Tracking Control for Heterogeneous Multiagent Systems. *IEEE Transactions on Neural Networks and Learning Systems* **2022**, *PP*, 1–10. https://doi.org/10.1109/TNNLS.2022.3146814.
45. Badrno, H.; Baradarannia, M.; Bagheri, P.; Badamchizadeh, M.A. Distributed Predictive Consensus Control of Uncertain Linear Multi-agent Systems with Heterogeneous Dynamics. *Iranian Journal of Science and Technology, Transactions of Electrical Engineering* **2022**, *47*. https://doi.org/10.1007/s40998-022-00544-y.
46. Raziei, Z.; Moghaddam, M. Adaptable automation with modular deep reinforcement learning and policy transfer. *Engineering Applications of Artificial Intelligence* **2021**, *103*, 104296. https://doi.org/10.1016/j.engappai.2021.104296.

 doi:10.20944/preprints202402.1124.v1

47. Yang, S.M.; Kim, K. Implementation of the conversation scheme in message-based distributed computer systems. *IEEE Transactions on Parallel and Distributed Systems* **1992**, *3*, 555–572. https://doi.org/10.1109/71.159039.
48. Xu, J.; Romanovsky, A.; Randell, B. Concurrent exception handling and resolution in distributed object systems. *IEEE Transactions on Parallel and Distributed Systems* **2000**, *11*, 1019–1032. https://doi.org/10.1109/71.888642.
49. Chen, J.; Huang, L. Supporting Dynamic Service Updates in Pervasive Applications. In Proceedings of the 2011 IEEE 35th Annual Computer Software and Applications Conference, 2011, pp. 273–278. https://doi.org/10.1109/COMPSAC.2011.43.