Analysis of an Event-Driven Data Relay Based Approach in Ramp Patrols

Yongsheng Shi and Yue Zhang
College of Aeronautical Engineering, Civil Aviation University of China, Tianjin, China

Keywords: Event-Driven Architecture, Application Layer, Information Delay.

Abstract: Aiming at the data relaying task of multiple devices in ramp inspection, this paper proposes an Event-Driven Architecture data relaying method running at the application layer. The method organizes devices of the same type through events, performs corresponding data relay operations, and distinguishes different types of data. The data relay task contains two types of subtasks: the data receive task and the data send task. For the data receiving task, it is regarded as a kind of event response; for the data sending task, it is regarded as a kind of class method, which is invoked through the class name. The method in this paper conducts comparative experiments and analyses the experimental data with the length of data relay as a variable. The results show that the method in this paper only increases the average transit time by 7.23ms when the length of transit data grows to 8.6 times of the original one, which can realize the data transit task and has a certain adaptability to the growth of data length.

1 INTRODUCTION

With the rapid growth of the civil aviation industry, there is an increasing demand for improved efficiency in ramp inspections (Rosa-Bilbao, 2023). A crucial area of research and development is the utilization of mobile robot inspection formations to optimize the efficiency of these inspections. Currently, the data transmission for apron inspection equipment relies heavily on third-party software, resulting in a bottleneck in the data interaction process (Lombardi, 2019). Therefore, it is imperative to design and implement a method that allows for seamless data interaction for inspection equipment (Li, 2021).

Event-driven based design thinking is widely used. By utilizing Event-Driven architecture, this paper proposes a data relay method that operates at the application layer (Wang, 2021). This method organizes different robots through Event-Driven Architecture, allowing for efficient communication and coordination. When receiving data relay tasks, the method handles them by responding to the corresponding events (Pogiatzis, 2020). On the other hand, when initiating data transfer tasks, the method is invoked through a class method (Rahmani, 2021). Event-based forwarding method is more suitable for real-time data processing (Fertier, 2020). At the same time, the privacy-preserving end-to-end data forwarding has also aroused research work (Zhang, 2019). Related work on event-based transit methods in industrial robots has also been carried out (Semeniuta, 2019).

This paper also addresses the roles of the server-side and client-side in this system. The proposed method is implemented and validated, demonstrating its effectiveness in improving efficiency and streamlining data interaction in ramp inspections.

2 DIVISION OF DATA RELAY TASKS

Ramp inspection equipment in the process of performing the task will produce a variety of types of inspection task data, such as inspection log files, etc., the above data need to be shared by the inspection equipment, and through the server which will be transferred to another inspection equipment. The above process consists of two basic operation units: data receiving operation and data sending operation as shown in the Figure 1.

Shi, Y. and Zhang, Y.
Analysis of an Event-Driven Data Relay Based Approach in Ramp Patrols.
DOI: 10.5220/0012880100004536

In *Proceedings of the 1st International Conference on Data Mining, E-Learning, and Information Systems (DMEIS 2024)*, pages 69-74
ISBN: 978-989-758-715-3

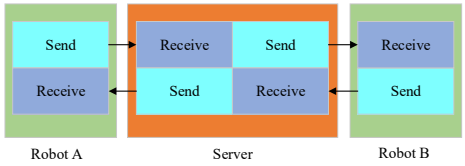

Figure 1: Data relay role interaction diagram.

The data relay method proposed in this paper involves the interaction of three actors: Robot A, Robot B, and the server. The process begins with Robot A and Robot B generating data, which is then relayed by the server. Each actor is equipped with communication modules that facilitate data transmission and reception.

To initiate the data relay process, Robot A's sending module interacts with the server's receiving module. This allows the data generated by Robot A to be relayed to the server. Once the server receives the data through its receiving module, it proceeds to forward it to Robot B using its own sending module. Finally, Robot B obtains the data by utilizing its data receiving module.

It is important to note that the roles of Robot A and Robot B can also be reversed in this data relay process. This means that Robot B can become the sender, sending data to the server, which then relays it to Robot A. This interchange of roles enables a dynamic and flexible data transmission between the two robots and the server.

By employing this data relay method, efficient and reliable data transmission is achieved among the three actors involved in the system. This enables seamless communication and coordination, ultimately improving the overall efficiency and effectiveness of ramp inspections.

3 DESIGN OF DATA RELAY METHODS

3.1 Server Side Design

The server side plays a crucial role in the data relay process, as it is responsible for both data reception and data forwarding. In order to effectively organize the different types of devices involved, the server side utilizes separate classes for each device type. This allows for a structured and organized approach to managing the data transmission process.

To provide a visual representation of the system's architecture, a class diagram is drawn. This diagram depicts the relationships and interactions between the various classes involved in the data relay process. It serves as a blueprint for understanding the system's design and functionality.

The class diagram illustrates the different classes for each type of device, highlighting the specific methods and attributes associated with them. It provides a clear overview of how the server side organizes and manages the data transmission between the devices. Utilizing this class diagram as a design reference, the server side can effectively handle the tasks of data reception and forwarding. This structured approach enhances the overall efficiency and reliability of the data relay process on the server side.

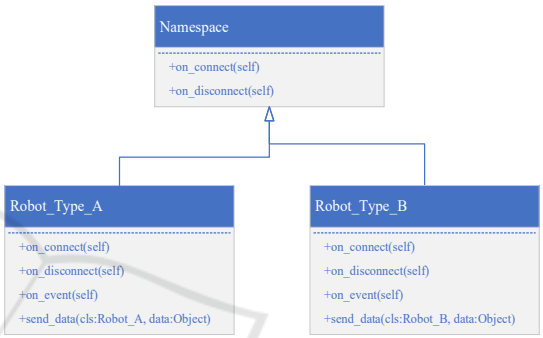

Figure 2: Server-side function class design.

The JSON message at the top of the data relay process serves as a comprehensive and specific source of information about the data that needs to be relayed for the ramp patrol scenario. It includes a detailed description of the data elements and their corresponding values that are essential for the effective implementation of the data relay process.

The data format of the server-side interaction with the client is encapsulated by json in the following Table 1.

By encapsulating the relevant data in this JSON message, it provides a standardized and structured format for the transmission of information. This ensures that all the necessary details required for the ramp patrol scenario are included and can be easily interpreted by the receiving entities.

The JSON message contains key information such as the location of the ramp, the time of the patrol, the status of the inspected objects, and any anomalies or issues detected during the inspection. Additionally, it may include details about the robots involved, their specific roles, and any specific instructions or tasks assigned to them.

In conclusion, the JSON message at the top of the data relay process serves as a complete and detailed repository of information for the ramp patrol scenario.

Its inclusive nature facilitates smooth and efficient data transmission, enabling seamless coordination and effective execution of the inspection tasks.

The event-driven model allows the server to handle a large volume of inspection events in a non-blocking manner, without having to create and manage a thread or process for each event. This approach can improve server throughput, reduce resource consumption, and better accommodate peak traffic.

Table 1: This caption has one line so it is centered.

Key	Byte Size of Value
Time Stamp	24
Move Command	45
Control Command	45
Device Data	60
Reserved Data	10

3.1.1 Data Reception Operations

Listening for Events. The server maintains one or multiple event listeners, which are dedicated to detecting when new events occur. In the context of apron inspection data, these events typically represent notifications of new inspection data. Event listeners are usually bound to one or more event queues, which are managed by a messaging middleware to handle the data flow.

Receiving Events. Once a new inspection event is generated by robot and sent over the network to the server, it enters the pre-established event queue. The server's event listener pulls the event from the queue, ready for processing. This process is asynchronous to ensure the server can handle a high volume of concurrent event streams.

Processing Events. The event handler on the application server is responsible for processing each event.

Generating a Response. Once an event is processed, the server might need to generate a response to inform the inspector that the data has been received and processed. The response can be sent back to the inspector through the same event-driven mechanism or through other response systems (like real-time messaging systems or email notifications) to communicate the outcome.

3.1.2 Data Sending Operations

Triggering Notifications: Should the monitored data exceed predefined thresholds or reveal anomalies, the server will generate alarm or notification messages.

Updating Status: The server will transmit data to the apron monitoring panel or related systems to update the status of aircraft and equipment or the results of monitoring. This enables real-time monitoring, allowing ground staff to be immediately informed of the latest conditions on the apron.

Scheduling Subsequent Processes: Based on inspection results, the server may need to automatically schedule maintenance crews for repairs or assign follow-up routine inspections. Relevant scheduling information will be sent to scheduling systems or posted to workforce management systems.

Data Synchronization: The server may send processed data to other systems for data synchronization, such as transmitting safety inspection results to the safety management system.

Data Backup: Periodically or after a critical event, the server will also transfer data to backup systems to ensure data security.

Through this process, the server not only processes and stores data from the apron inspections but also feeds back crucial information to relevant personnel and systems, ensuring the apron operates safely and efficiently. The application of an event-driven architecture accelerates server responsiveness, enabling it to promptly respond to various events, thus maintaining the operational efficiency of the entire airport.

3.2 Robot Side Designs

In the context of ramp inspections, the architectural framework employed for the development of the robot and server sides exhibits remarkable similarities, as depicted in Figure 3. This resemblance, however, belies a crucial divergence in functionality—the robot side places a far greater emphasis on the autonomous dispatch of data that is intended for subsequent relaying. This pivotal feature necessitates the implementation of a sophisticated event-driven mechanism capable of local monitoring.

To address this requirement, the event listening architecture is meticulously crafted and presented in Figure 4. It is engineered to facilitate real-time surveillance of specific activities within the robot's operational environment. At the heart of this system is the capability to observe and react to changes occurring in files, a fundamental necessity for the robot to fulfil its objective of data transmission.

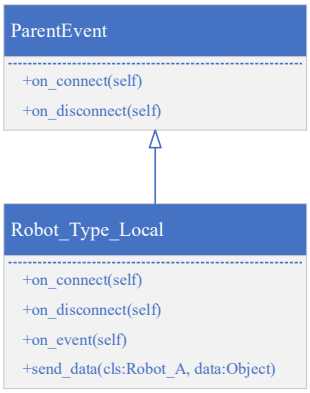

Figure 3: Robot-side function class design.

Upon acquisition, the data collated by the inspection robot undergoes a process of transformation, relaying into a persistent file-based format. Consequently, these files are stored within a designated directory whose alterations must be closely observed. To that end, the event listening system is primed to detect any file modification events within the target folder.

The occurrence of such an event triggers a meticulously defined response protocol. This protocol entails the invocation of a corresponding class method—one that is adept at handling the data sending operation. The class, specified by its name, is effectively engaged through this listener system, thereby catalysing the data forwarding function.

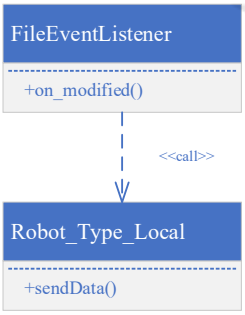

Figure 4: File event listener calling a robot's data sending static method.

To elaborate, when the designated file directory experiences an addition, deletion, or amendment, the event listening system springs into action. It instantaneously signals the specified class, which, in turn, powers up the necessary method to process and expeditiously dispatch the data to its intended destination. This automated synergy between file modifications and data transmission ensures a seamless and uninterrupted flow of information, a critical aspect of the inspection robot's utility.

Overall, the design philosophy underlying the inspection robot's class structure is to deliver a robust and autonomous system tailored for the dynamic ramp inspection environment. As a result, the inspection robot performs its duties with a high degree of efficiency, accuracy, and reliability—key qualities necessary for the demanding tasks it is entrusted with.

4 TEST FOR DATA RELAY METHOD

Experiments are designed to test the relay performance of the method using the average statistical time as an index. The method of counting the relay time is to record the timestamp at the sending end when sending, and get the current timestamp at the receiving end to obtain the relay time.

4.1 Experiment 1

Repeatedly forward 500 pieces of data with a data length of 184Bytes and a time interval of 1s, and calculate the time for each relay. The vertical axis is the time difference in milliseconds, and the horizontal axis is the number of forwarding times. The experimental results are shown below.

From the above graphs, we can get that with the data length of 184Bytes in Exp.1, most of the relay time of this paper's method is between 20-70ms, and in a few cases, the relay time goes beyond this range, with a minimum value of 14.169ms or a maximum value of 139.932ms. In Exp.1, the geometric mean of the relay time is 43.98ms, the geometric standard deviation is 19.09 and the median is 47.49ms.

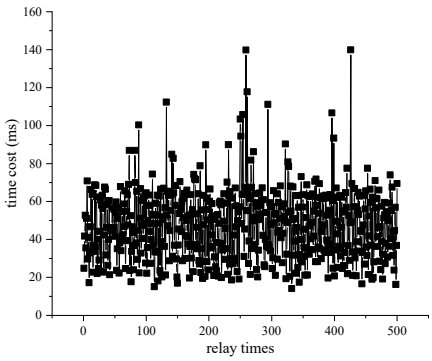

Figure 5: Raw data on relay time of Exp.1.

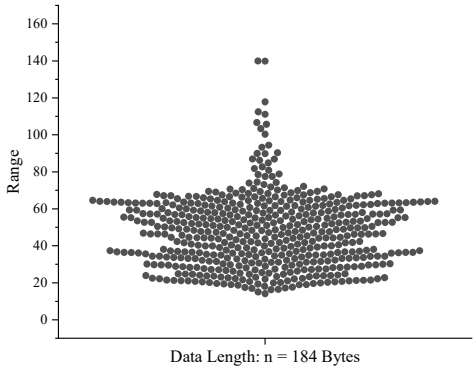

Figure 6: Beeswarm plot of Exp.1.

4.2 Experiment 2

Repeatedly forward 500 pieces of data with a data length of 1584Bytes and a time interval of 1s to calculate the time of each relay. The vertical axis is the time difference in milliseconds, and the horizontal axis is the number of forwarding times. The experimental results are shown below.

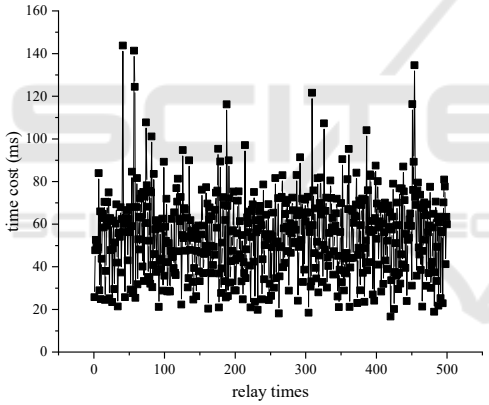

Figure 6: Raw data on relay time of Exp.2.

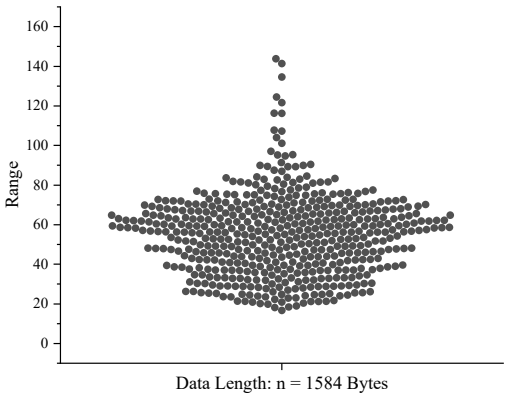

Figure 7: Beeswarm plot of Exp.2.

From the above graphs, we can get that with the data length of 1584Bytes in Exp.2, most of the relay time of this paper's method is between 20-80ms, and in a few cases, the relay time goes beyond this range, with a minimum value of 16.682ms or a maximum value of 143.802ms.In Exp.2, the geometric mean of the relay time is 51.21ms, geometric standard deviation is 1.47, and the median is 56.23ms

5 CONCLUSION

In this paper, to provide a comprehensive comparison of the outcomes from two distinct experiments, we employ graphical representations in the form of box-scatter plots and distribution-axis-whisker plots. The usage of these plots allows for an effective visual comparison and analysis of the data distributions, statistical variations, and potential outliers. The effects of the two experiments are shown in Fig.9 and Fig. 10.

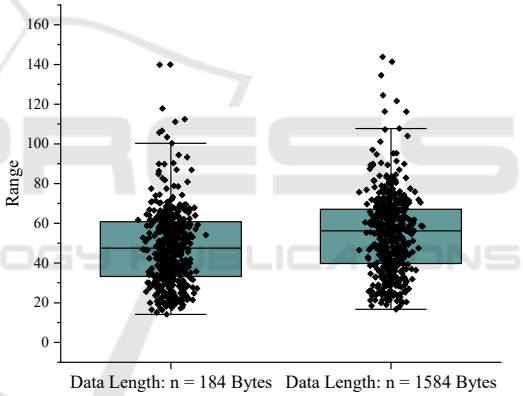

Figure 8: Box-and-Scatter Chart of Exp.1 and Exp2.

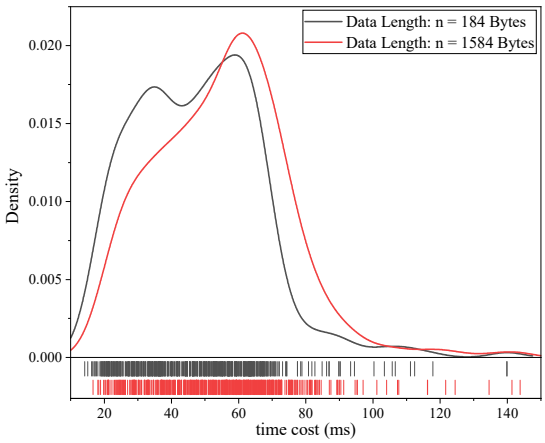

Figure 9: Distribution map - Axis whiskers plot of Exp.2.

From the above graphs, it can be obtained that with the increase of data length of each relay, the position of the box in the box plot is shifted upward, i.e., there is a smaller increase in the relay time of each relay accordingly. And from Fig. 10, it can be obtained that with the increase of data relay length, the density of data distribution decreases in the range of 20ms-30ms and increases near 70ms-80ms.

Overall, the method proposed in this paper is able to realize the data relay function in the ramp inspection scenario, and the data relay time is kept in a suitable range. The results show that the method in this paper only increases the average transit time by 7.23ms when the length of transit data grows to 8.6 times of the original one, which can realize the data transit task and has a certain adaptability to the growth of data length. The increase in data length has an effect on the data relay time, but it does not cause dramatic fluctuations in data relay time.

REFERENCES

Tian, Y., Meng, Y., Zhao, X., Wen, X., Meng, L., Qin, X., 2023. *Age-aware relay strategy with simultaneous wireless information and power transfer in remote monitoring systems*. IET Communications 17, 1418–1431.

Rosa-Bilbao, J., Boubeta-Puig, J., Rutle, A., 2023. *CEPEDALoCo: An event-driven architecture for integrating complex event processing and blockchain through low-code.* Internet of Things 22, 100802.

Lombardi, F., Muti, A., Aniello, L., Baldoni, R., Bonomi, S., Querzoni, L., 2019. *PASCAL: An architecture for proactive auto-scaling of distributed services.* Future Generation Computer Systems 98, 342–361.

Li, W., Li, H., Wang, S., 2021. *An event-driven multi-agent based distributed optimal control strategy for HVAC systems in IoT-enabled smart buildings.* Automation in Construction 132, 103919.

Wang, Yahui, Zheng, L., Wang, Yiwei, 2021. *Event-driven tool condition monitoring methodology considering tool life prediction based on industrial internet.* Journal of Manufacturing Systems 58, 205–222.

Pogiatzis, A., Samakovitis, G., 2020. *An Event-Driven Serverless ETL Pipeline on AWS.* Applied Sciences 11, 191.

Rahmani, A.M., Babaei, Z., Souri, A., 2021. *Event-driven IoT architecture for data analysis of reliable healthcare application using complex event processing.* Cluster Computing 24, 1347–1360.

Fertier, A., Montarnal, A., Barthe-Delanoë, A.-M., Truptil, S., Bénaben, F., 2020. *Real-time data exploitation supported by model- and event-driven architecture to enhance situation awareness, application to crisis management.* Enterprise Information Systems 14, 769–796.

Zhang, W., 2019. *A data fusion privacy protection strategy with low energy consumption based on time slot allocation and relay in WBAN*. Peer-to-Peer Networking and Applications 12, 1575–1584.

Semeniuta, O., Falkman, P., 2019. *Event-driven industrial robot control architecture for the Adept V+ platform*.