

Optimizing Elevator Performance with SARL Multi-Agent Systems: A Distributed Approach for Enhanced Responsiveness and Efficiency

Vy Le, Oliver Harold Joegensen, Tin Nguyen, Khang Nguyen Hoang and Ginel Dorleon^a
School of Science, Engineering and Information Technology, RMIT University, Ho Chi Minh City, Vietnam

Keywords: MAS, Optimization, Distributed System, Centralized System.

Abstract: Elevators play a pivotal role in modern urban living, boosting productivity and convenience efficiently. In elevator systems, the optimization of Multi-Agent Systems (MAS) is indispensable as it enhances agent coordination, adaptability, delay reduction, client satisfaction, and resource use. In this paper, we introduce an algorithm based on SARL MAS designed to enhance elevator controller performance. Our approach compares Centralized and Distributed Agent Systems, demonstrating the superiority of Distributed Agent Systems due to their improved responsiveness, efficiency, and adaptability. Our findings provide valuable insight into the use of SARL MAS not only for elevator control but also for other applications such as queue management systems and resource allocation in computing, highlighting the benefits of a distributed approach.

1 INTRODUCTION


More than just a mechanical invention, the elevator is a transformative innovation that transformed our modern infrastructure. It reshaped how we think about and construct multi-story structures, forever altering the landscape of cities. The early elevator dispatch systems were rudimentary, relying on a simple button-up /-down system that randomly assigned elevators to passengers (Al-Kodmany, 2023). Subsequent advances in dispatch algorithms have transcended these basic methods, resulting in notable reductions in waiting times, travel distances, and energy consumption (Pepyne and Cassandras, 1998).

This study seeks to revolutionize the application of multi agent systems (MAS) in elevator operations, addressing inherent challenges and offering solutions that promise increased efficiency, time savings, and reduced energy usage. Focusing specifically on apartment buildings, our research aims to design an elevator algorithm using SARL (Rodriguez et al., 2014). The choice of SARL stems from its alignment with the simulator and its compatibility with Aspect-Oriented Programming (AOP), particularly suited for addressing complex AI problems. Furthermore, SARL incorporates holonic multi-agent systems (HMAS) (Giret et al., 2005), proven effective

in modeling traffic networks, implementing self-learning mechanisms for energy-predictive planning, and coordinating distributed sensors. Unlike real-life elevators, our system encompasses two distinct calls: the elevator call, where passengers press Up () or Down () to enter the car, and the car call, where passengers within the car select their desired floors. The system comprises multiple elevators, forming a cohesive Multi-Agent System (MAS).

In this paper, we undertake the design, development, and evaluation of two elevator controllers, each with a unique approach. The first is a centralized controller, responsible for allocating cars for the entire system, while the second is a distributed controller, providing each elevator with a separate and independent control mechanism. Multiple elevator cars share identical behaviors within the system. Several assumptions guide our study:

- Each car must complete its assignment before moving on to the next one, excluding reversals or skipping floors where passengers intend to exit.
- An empty car possesses the flexibility to execute any available option (stop, go up, or go down).
- The nearest floor is defined as the one that the car would reach first in its current direction. When the elevator is stationary, the current floor substitutes for the nearest floor.

^a  <https://orcid.org/0000-0003-2343-4445>

- Cars maintain a constant speed, ensuring entirely predictable travel times between any two floors based on the floor distance.
- Each car has a weight capacity of 800kg, with in-capacity leading to a restriction on serving more customers. If an elevator cannot accommodate all passengers within a hall call, remaining passengers initiate a new hall call. When an elevator is in motion and reaches its maximum capacity, it cannot stop to answer a hall call.

Our contributions in this paper can be summarized as follows:

1. We introduce a novel algorithm founded on an estimated time of dispatch, providing precise guidance to every elevator within the Multi-Agent System (MAS). This algorithm optimizes the decision-making process, improving overall elevator performance.
2. We successfully developed two distinct types of elevator system controllers, both leveraging the proposed algorithm. Through comprehensive testing, these controllers showcase the algorithm's efficacy in improving the efficiency and responsiveness of elevators, thereby contributing to the advancement of elevator control systems.
3. Using the proposed algorithm, we demonstrate a significant improvement in the efficiency of the distributed system controller. This optimization aspect not only enhances elevator responsiveness but also maximizes resource utilization, contributing to a more sustainable and effective operation of the distributed elevator system.

The rest of this paper follows this structure: in Section 2, we summarize the different existing methods that address elevator controllers optimization. In Section 3, we introduce our approach, define our agents with their skills and the different controllers that will be using in the experiment. Experimental results are described and analyzed in Section 4 Chapter. Conclusion and perspectives are presented in Section 5.

2 RELATED WORK

To establish a solid foundation for our understanding of elevator dispatch systems, we have delved into existing literature. Our focus in this study centers on passengers' requests (P_n) and Estimated Time to Dispatch (ETD) estimation. We are particularly intrigued by these aspects due to their pivotal role in optimizing elevator performance and enhancing user experience.

This exploration serves as a strategic basis for our development of a more robust algorithm that addresses the core challenges within elevator control systems.

2.1 The Three-Passenger Approach

In the field of elevator control, the Three-Passenger Approach, as introduced by Rong, Hakonen, and Lahdelma in their elevator group control algorithm (Rong et al., 2003), offers a refined strategy for categorizing passengers, aiming to optimize system efficiency with scientific precision. This innovative approach meticulously classifies passengers into three distinct categories: Passenger one (P1) requests, efficiently served in the current travel direction of the elevator; passenger two (P2) requests, requiring a single direction reversal; and passenger three (P3) requests, necessitating two reversals.

Scientifically, these categorizations are pivotal in the assignment of passenger requests to elevators, directly impacting the overall system performance. By strategically considering the travel characteristics associated with each passenger type, this approach minimizes unnecessary direction changes, reduces energy consumption and optimizes the operational efficiency of the elevator system.

The Three-Passenger Approach, grounded in empirical research and theoretical modeling (Rong et al., 2003), underscores its scientific relevance by providing a systematic framework to handle diverse passenger scenarios. Its application in elevator control systems aligns with the broader goal of advancing efficiency and sustainability in urban transportation infrastructures. In conclusion, the Three-Passenger Approach represents a scientifically informed strategy, contributing significantly to the optimization of elevator systems and paving the way for more sophisticated control algorithms in the field. The concept is described in Figure 1 below:

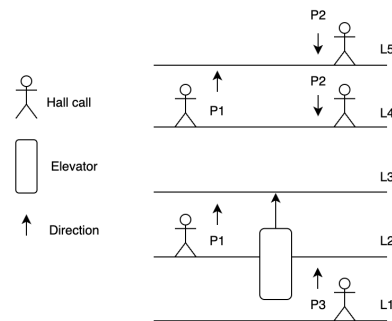


Figure 1: The three-passenger concept.

2.2 Estimated Time of Dispatch (ETD)

Estimated Time to Dispatch (ETD) emerges as a pivotal metric in the domain of destination-based elevator dispatching algorithms (Muñoz et al., 2006; Rong et al., 2003; Sorsa et al., 2009). This metric is strategically designed to minimize the collective travel time of passengers, encompassing both waiting time and travel time in the car. By optimizing this temporal aspect, the system aims to elevate passenger service quality and increase the overall handling capacity of the elevator system (Latif et al., 2016)

In its nuanced approach, the ETD system dynamically allocates elevator cars based not only on the proximity of calls but also on the occupancy status of the elevator car. This dual consideration results in a more sophisticated and responsive dispatching strategy, enhancing the user experience by minimizing unnecessary stops (Fujino et al., 1997; Sorsa et al., 2009; Tanaka et al., 2016). Scientifically, the incorporation of ETD has been shown to yield tangible benefits, with studies indicating a remarkable 25% reduction in overall trip times coupled with a substantial 30% increase in elevator capacity (Smith, 2002).

Furthermore, the efficiency gains attributed to ETD underscore its scientific significance, making it a valuable tool in modern elevator control systems. In conclusion, ETD stands as a scientifically substantiated approach that not only optimizes elevator system performance but also significantly enhances the user experience through its nuanced and adaptive dispatching algorithm.

3 OUR PROPOSED APPROACH

We introduce our approach by addressing the fundamental question: What constitutes a proficient elevator controller? In pursuit of this query, we meticulously consider several key aspects that collectively contribute to the overarching goal of designing a more intelligent elevator system. These crucial properties, which include responsiveness, efficiency, fairness, accessibility, and adaptability, are essential to improve the performance of the system and its ability to adapt to evolving user demands.

To tackle the imperative of responsiveness, the controller must swiftly incorporate user signals into the query queue for prompt processing, leveraging Estimated Time to Dispatch (ETD) techniques to effectively group users based on their chosen floors. This strategic approach not only reduces user waiting times but also ensures the smooth operation of the elevator infrastructure. Efficiency, a critical criterion, in-

volves minimizing the total travel distance covered by the elevator. This is achieved by thoughtfully ordering the sequence in which floors are visited, thereby mitigating unnecessary travel. The controller must also guard against bunching, optimizing the elevator's movement for optimal efficiency.

Fairness in elevator operations mandates universal access to every floor and equal waiting times between floors. The controller discourages prioritizing idle elevators on a specific floor, ensuring equitable service distribution. Additionally, integrating accessibility features, such as audio and visual cues, caters to passengers with diverse needs. Lastly, the adaptability of the elevator controller describes its capacity to adjust based on factors like capacity considerations, skipping floors when at full capacity, and seamlessly adapting to elevators out of service while upholding the aforementioned properties.

To enhance adaptability further, predictive analytics can be employed, leveraging historical data to predict peak usage times and adjusting elevator allocation accordingly. Proactive measures, like anticipating high-demand periods and allocating additional resources during those times, contribute to minimizing waiting times. In essence, a comprehensive elevator controller must balance these multifaceted considerations to optimize its performance, ensuring a responsive, efficient, fair, accessible, and adaptable system that meets the diverse needs of its users.

3.1 Defined Agents in the MAS

In order to setup our comparative study, we have defined in Table 1 below the list of agents that we use and are part of our MAS. For better presentation, we shorten ElevatorSimPercept¹ into ESP.

3.2 The Rule-Based Approach

From the background of three passengers concepts and ETD, the overall strategy of our design is to have the shortest ETD serve to passengers. This will form the decision-making process of the agents. To achieve this, we follow the rule-based approach. Our rule, in short, involves checking if a new hall call (i) is a duplicate call to an unassigned request, (ii) has already been assigned to an elevator. Subsequently, the system calculates the ETD for the unfilled cars and assigns the most suitable car for the call, in order to achieve the lowest possible ETD. This rule-based approach offers us some uniqueness in comparison with other planning mechanisms (Sakita, 2001). For example, it fastens the overall decision making process

¹ESP: ElevatorSimPercept

by removing unqualified situations as soon as possible before doing any computations. In addition, the rule-based approach enables real-time responsiveness, allowing the system to quickly adapt to changing environments and make decisions efficiently.

3.3 Estimation & Allocation Approach

Due to the limitation of the simulation environment used for evaluation, we defined our own ETD metric, inspired by a thorough literature review (Rong et al., 2003). We define ETD of car i with an existing plan P through the following formula:

$$t_i^{\text{total}} = \sum_{j \in P'} t_{i,j}^{\text{stop-time}} + t_i^{\text{reach}} \quad (1)$$

Where P' is the stops planned by the elevator between the current floor and the destination floor, taking into account $P1, P2$, and $P3$ calls, $t_{i,j}^{\text{stop-time}}$ and t_i^{reach} are constants that are specific for elevator i and floor j .

The ETD calculation is the main logic for our system's elevator allocation approach. When given a new request (tentative request), the system will decide which elevator to serve that new request through these ETD estimations. This strategy is applied in both Central and Distributed Controllers. To allocate a tentative request to an elevator, we estimate the ETD for each car, given its existing plan, and greedily assign it. We present below the pseudo-code of our algorithm:

```

Data: List  $i$ ; Request  $R$ ; ElevatorState  $C$ 
Result: Total estimated time of destination (ETD)
 $destination \leftarrow \text{estimateDestination}(R, C);$ 
 $P.add(destination);$ 
 $P.sortByExecution(C);$ 
   $\triangleright$  Sorts  $P$  in  $P_1, P_2, P_3$   $totalTime \leftarrow 0;$ 
for  $i \in P.length$  do
   $totalTime \leftarrow totalTime +$ 
     $\text{travelTime}(P[i], P[i+1]);$ 
   $totalTime \leftarrow totalTime +$ 
     $\text{waitingTime}(P[i]);$ 
  if  $P[i]$  is external then
     $P.add(\text{estimateDestination}(P[i], C));$ 
  end
  if  $P[i] == destination$  then
     $\text{Return}(totalTime);$ 
  end
end

```

Algorithm 1: ETD Calculation Algorithm.

To explain it further, when it comes to assigning elevators, the first crucial step is to evaluate the elevator's capacity in order to avoid overcrowding. Next,

Table 1: Agent, skills and description.

Agent	Skills	Description
Orchestrator	<code>perceive():SystemState</code> <code>reason(SystemState): List(Action)</code> <code>actuate(List(Action)): void</code>	The Orchestrator acts as a conductor, focusing on intra-agent communication and orchestration. It has three actions: 'perceive' retrieves the system state, 'reason' signals the Resource Allocator to plan, 'actuate' signals the Safety Override Logic to validate, and then dispatch the elevators.
Resource Allocator	<code>plan(SystemState): List(Action)</code> <code>checkCapacity(SystemState): List(bool)</code> <code>estimateTimeToDispatch(SystemState): List(int)</code>	The Ressource Allocator acts as a planner for the system. It: 'plan' is the main skill which derives a plan, 'checkCapacity' evaluates whether the elevators are over capacity, 'estimateTimeToDispatch' gets the ETD for each elevator.
Observer	<code>normalizeEvent(ESP): ESP</code> <code>addSensorInformation(ESP):ESP</code> <code>stateUpdateTrigger(ESP): void</code> <code>fires StateUpdate</code>	The Observer is responsible for interpreting the percepts retrieved from the simulator, and then add potential additional information. It has three action, 'normalizeEvent' is responsible for normalizing the percepts retrieved. 'addSensorInformation' is responsible for adding sensory information, like weight of the passengers. 'stateUpdateTrigger' signals the SystemStateAgent to update its internal state.
Safety Override Logic	<code>getEmergencyPlan(): List(Action)</code> <code>validate(List(ElevatorState), List(Action)): void</code> <code>fires DispatchElevator</code> <code>listenForFire(): void</code> <code>fires DispatchElevator</code>	Safety Override Logic is responsible for any unexpected behaviour during execution regarding the fire and emergency plan. The 'getEmergencyPlan' is to define the list of 'elevator's actions' to adapt the emergency situation. 'validate' action is for checking at anytime the building is in emergency. 'listenForFire' is to keep checking if there is any fire within the system.
SystemState Agent	<code>queryState(): SystemState</code> <code>updateState(ESP): void</code> <code>fires StateChange</code> <code>triggerReplan(): void</code> <code>fires StateChange</code>	The SystemStateAgent is responsible for saving and updating the current state of our system. The 'queryState' is to return the current SystemState, 'updateState' fires StateChange to update the current SystemState, and 'triggerReplan' is called when replanning is necessary after the updating.

it adds a request to the existing queue of tasks for the elevator. Consequently, these actions are arranged based on the ‘three-passenger principle’ to maximize efficiency. After that, the system determines the expected destination by computing the middle floor between the call and the top end of the same direction, to have the floor that may potentially be chosen. Next, the agent goes through an iterative procedure, in which the system repeatedly goes through the list of actions until it arrives at the estimated floor that matches the provisional request. Through each iteration, the ETD is determined by accumulating the total travel and stopping times for the elevator. This methodological methodology guarantees the efficient and effective administration of elevator operations.

3.4 The Centralized-Agent (CA) Smart Controller

Together with the proposed above method, we create a holonic agent to logically separate internal responsibilities associated with controlling an elevator system. This assures clear distinction between our proposed method and the remaining associated tasks (such as request management, state management, etc.). Its structure is defined in Figure 2 below.

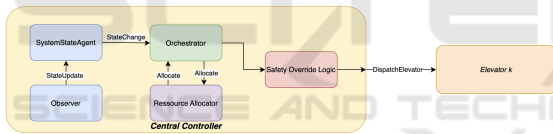


Figure 2: Centralized Controller Architecture.

First, the SystemStateAgent one of the critical agents that it receives from the environment through Observer agent and updates the states. The SystemStateAgent keeps track of the environment state (number of elevators, number of floors) and state of each elevator. Orchestrator gets the SystemState from SystemStateAgent, given that a request was created. It then collaborates with Resource Allocator to do the ETD calculations and allocates a car to the request. Note that, in a centralized agent controller, all the ETD calculation and distribution tasks will be done only in Resource Allocator. It is finally passed to the SafetyOverride logic, which serves to block elevator actions in case of emergencies.

3.5 The Distributed Smart Controller

Beside a central controller, we also define a distributed controller. Our distributed elevator control system will be structured similarly to the central elevator controller. The biggest difference is that each

elevator has its own dedicated controller. Orchestrator and SafetyOverride agents will keep the same design as before for reusability purposes, while Observer, SystemStateAgent and ResourceAllocator agents will be modified to accommodate for the needs of communication between distributed controllers. This distributed design has the advantage of being independent of the other elevators, e.g. should one elevator fail and shut down, the others will function as normally.

Additionally, the system scales horizontally, meaning possible expansions to the elevator system require no downtime, as the elevator can simply be added as an agent. This introduces a level of robustness and adaptability to the entire system, which are both desirable qualities. Structure of the Distributed Controller Architecture is presented below in Figure 3 below:

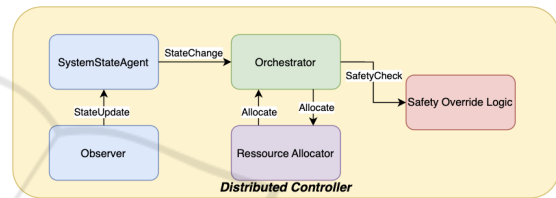


Figure 3: Distributed Controller Architecture.

As a consequence, each elevator is only responsible for calculating its own ETD. To facilitate communication between controllers we define a custom space, the NegotiationSpace. The NegotiationSpace inherits from SARL’s OpenEventSpace, and it facilitates three tasks:

- Negotiation between agents for elevator allocations based on lowest ETD estimate
- Sharing if a request has already been assigned
- Sharing when a request has been completed

All these tasks are synchronized using mutual exclusion locks and Java’s CyclicBarrier to ensure correct decision making. This system scales well and reuses existing internal agents.

However, it’s limited by the slowest controller due to synchronization. The impact of this limitation is minimal in our implementation, so possible mitigation solutions such as setting maximum waiting negotiation time are not required. A random controller among those with the minimum ETD will be selected for dispatch in order to prevent possible load imbalance.

4 EXPERIMENTS & RESULTS

In this section, we conduct a comprehensive comparative analysis of the designed controllers, evaluating their performance and ability to manage congestion on the floors. The experiments were carried out using the SARL Eclipse IDE, alongside an elevator simulator featuring three simulated elevators. For a similar setup, please refer to the SARL elevator simulator repository².

4.1 Evaluation Framework

We evaluate the controllers based on two main parameters: i) the number of people in the elevator and ii) the distribution of these people. For the first parameter, we consider scenarios with 20 and 60 people, representing light and heavy load conditions, respectively. Regarding the second parameter, we examine a variety of distribution scenarios, including random and uniform distributions, as well as two types of traffic bottlenecks: morning congestion, with passengers starting on the lowest floor, and evening congestion, with passengers distributed across upper floors attempting to reach the lowest floor. This range of scenarios ensures a comprehensive evaluation of the system's performance under different conditions.

4.1.1 Evaluation Metrics:

To evaluate the efficiency of the two controllers, we focus on key metrics: the average wait time and travel time for each passenger, which together reflect the total time required to fulfill a request. Additionally, we measure the average travel distance covered by each elevator to assess system effectiveness. The definitions of these metrics are summarized in Table 2 below.

To ensure reliable evaluation, we average results over 10 seeded runs, capturing diverse scenarios and minimizing random influences on the rule-based algorithm's performance.

Table 2: Evaluation metrics of the controller systems.

Metric	Definition
Average Total Time	Average time (ms) to answer a hall call and deliver the passenger to their destination.
Average Travel Distance	Average distance (floors) traveled by all elevator cars.

²<https://github.com/ssardina-agts/elevator-simulator>

4.2 Experimental Results

This section presents the results of our comparative study on centralized and distributed approaches to SARL multi-agent systems for elevator control, summarized in Tables 3 and 4. The analysis focuses on the following performance metrics: waiting time, travel distance, and energy consumption.

Table 3: Average total time in milliseconds: the average sum of waiting time and travel time for each passenger.

Passenger Count	Central		Distributed	
	20	60	20	60
Random	71,15	112,1	72,7	113,3
Morning	63,7	99,31	65,7	110,4
Evening	75,3	110,8	72,7	99,9

Table 4: Average car travel distance in floors: the average total travel distance for a simulation run.

Passenger Count	Central		Distributed	
	20	60	20	60
Random	169.7	177.7	176.5	184.1
Morning	163.7	235.0	166.7	215.0
Evening	287.5	315.0	330.0	374.2

While the results are somewhat consistent across scenarios, passenger volume significantly affects response time. Notably, in the evening scenario, system efficiency declines. This is likely because all requests are DOWN, effectively doubling travel distance, as elevators cannot pick up passengers while traveling UP.

4.3 Waiting Time

Our analysis, illustrated in Figure 4, shows a reduction in passenger waiting times with the distributed approach compared to the centralized approach. While Table 3 indicates that the distributed approach can sometimes result in longer total travel times, the Kernel Density Estimation (KDE) chart reveals that passengers generally experience shorter waiting times. This improvement stems from the autonomy of individual elevators in the distributed system, allowing them to respond dynamically to changes in passenger demand. This adaptability leads to significantly shorter waiting times, highlighting the operational efficiency of the distributed control paradigm.

4.4 Travel Distance

Travel distance for elevator cars emerged as a key metric in evaluating system efficiency. Table 4 presents the average travel distance, measured in

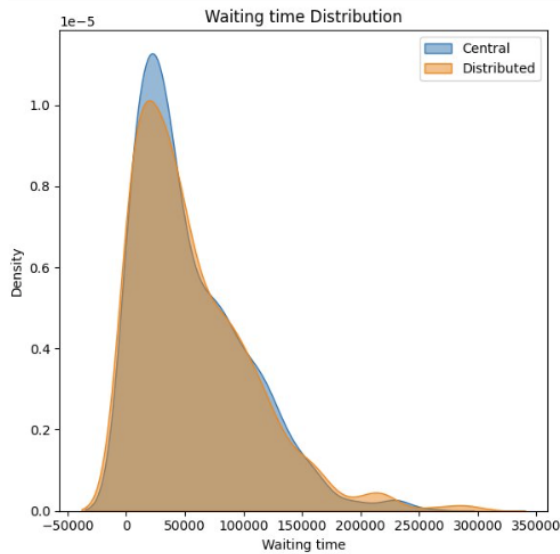


Figure 4: Waiting Time Distribution between two approaches.

floors, across various experimental scenarios.

The performance of centralized and distributed elevator systems is notably similar. In some scenarios, such as the morning period, the distributed approach not only matches but occasionally outperforms the centralized system, suggesting that the adaptability of the distributed approach can lead to more efficient operations. Future research could focus on improving algorithms for evening scenarios, where performance differences are more pronounced. Enhancing efficiency in high-traffic situations holds significant potential for optimization.

4.5 Energy Consumption

Energy consumption is a crucial factor in elevator systems, particularly for sustainability. Our findings indicate that the distributed approach, with its more efficient allocation of elevator cars, has the potential to reduce overall energy consumption compared to centralized systems. Future research should delve deeper into this by quantifying energy savings and evaluating their relevance in larger, more diverse building complexes.

4.6 DISCUSSION

Our evaluation reveals comparable performance between the controllers, with the distributed approach demonstrating notable advantages in robustness and efficiency. However, each controller has its own strengths and limitations.

Robustness of Distributed Systems. The centralized controller is vulnerable to total system failure, as it acts as a single point of failure. In contrast, the distributed controller enhances resilience by ensuring that a failure in one unit does not affect the entire system. This robustness is a significant advantage of the distributed approach. However, it also introduces the potential for inconsistent system states, a common challenge in distributed system theory (Sorsa et al., 2009). Future implementations could benefit from stronger consistency measures to address this limitation effectively.

Comparison Between Centralized and Distributed Controller. While the overall performance of both controllers is similar, they differ in various aspects:

- The centralized controller operates independently without the need to synchronize with external controllers, resulting in improved decision time.
- Maintenance shutdowns for the centralized controller entail a halt of the entire system in its current state.
- The distributed controller enhances robustness by eliminating a single point of failure.
- Although not evident during our evaluation, the distributed controllers may face challenges in maintaining a consistent state between them without appropriate consistency guarantees.

While not directly reflective of the explored method, it is important to note that during our experiments, we found that the solution implementation sometimes fails to respond to every request. We discovered that due to concurrency issues, race conditions arose where allocated requests would be lost. Our manual testing shows that this mostly happens during high-traffic scenarios (such as morning and evening scenarios), with a drop rate of 6.5%.

Despite identified issues in both systems, we assert that the distributed system exhibits greater resilience. The current challenges with the distributed controller are primarily implementation-related, and the observed increase in decision time during testing is deemed acceptable. Consequently, we recommend moving forward with the distributed controller for its overall advantages in robustness and performance.

5 CONCLUSION

This study successfully developed two distinct elevator system controllers, each demonstrating notable improvements in responsiveness and efficiency. The

first controller design focuses on optimizing communication protocols through a set of largely independent internal agents. The second design adopts a similar framework but places a greater emphasis on adaptability and robustness. After comprehensive evaluation and analysis, the distributed controller emerged as the preferred choice, primarily due to its resilience and potential for further improvements by addressing current implementation challenges.

The distributed approach is particularly advantageous due to its dynamic adaptation to fluctuating demands and its ability to operate independently. This flexibility enhances the efficiency of systems like queue management and resource allocation, where such adaptability is crucial for maintaining continuous and effective operation. When applied to queue management, it can dynamically adjust resource allocation based on real-time data, significantly reducing wait times and improving service efficiency. Similarly, in computing resource allocation, a distributed approach allows for a more agile response to workload changes, optimizing the use of computational resources and enhancing system performance.

Despite these advancements, the study faces certain limitations in both design and implementation. Notably, the absence of real-world usage data makes it challenging to refine Estimated Time to Dispatch (ETD) predictions. During testing, decision-making times require more detailed examination. While practical deployment could provide the necessary data, a deeper look at time efficiency will be a focus for future improvements. Another key consideration for future development is adhering to core principles of distributed systems, particularly in maintaining state consistency. It is important to note that any delays observed during testing of the distributed controller were deemed acceptable when weighed against the substantial benefits it offers.

In conclusion, we are confident that our proposed approach lays the foundation for developing more efficient and user-friendly elevator system controllers, enhancing the overall user experience. Future iterations will address the identified limitations and explore real-world deployment further, aiming for a more comprehensive understanding of the system's performance and its potential to elevate the standards of elevator control systems.

REFERENCES

- Al-Kodmany, K. (2023). Elevator technology improvements: A snapshot. *Encyclopedia*, 3(2):530–548.
- Fujino, A., Tobita, T., Segawa, K., Yoneda, K., and Togawa, A. (1997). An elevator group control system with floor-attribute control method and system optimization using genetic algorithms. *IEEE Transactions on Industrial Electronics*, 44(4):546–552.
- Giret, A., Botti, V., and Valero, S. (2005). Mas methodology for hms. In *Holonic and Multi-Agent Systems for Manufacturing: Second International Conference on Industrial Applications of Holonic and Multi-Agent Systems, HoloMAS 2005, Copenhagen, Denmark, August 22-24, 2005. Proceedings 2*, pages 39–49. Springer.
- Latif, M., Kheshaim, M., and Kundu, S. (2016). A review of elevator dispatching systems. *Engineering Letters*, 24(1):23–31.
- Muñoz, D. M., Llanos, C. H., Ayala-Rincón, M., van Els, R., and Almeida, R. P. (2006). Implementation of dispatching algorithms for elevator systems using reconfigurable architectures. In *Proceedings of the 19th annual symposium on Integrated circuits and systems design*, pages 32–37.
- Pepyne, D. L. and Cassandras, C. G. (1998). Design and implementation of an adaptive dispatching controller for elevator systems during uppeak traffic. *IEEE Transactions on control systems technology*, 6(5):635–650.
- Rodriguez, S., Gaud, N., and Galland, S. (2014). Sarl: a general-purpose agent-oriented programming language. In *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, volume 3, pages 103–110. IEEE.
- Rong, A., Hakonen, H., and Lahdelma, R. (2003). *Estimated time of arrival (ETA) based elevator group control algorithm with more accurate estimation*. Turku Centre for Computer Science.
- Sakita, M. (2001). Elevator system with multiple cars in one hoistway. *Elevator World*, 49(6):80–91.
- Smith, R. (2002). Etd algorithm with destination dispatch and booster options. *Elevator world*, pages 136–142.
- Sorsa, J. S., Ehtamo, H., Siikonen, M., Tyni, T., and Ylinen, J. (2009). The elevator dispatching problem. *Transportation Science*.
- Tanaka, S., Hoshino, D., and Watanabe, M. (2016). Group control of multi-car elevator systems without accurate information of floor stoppage time. *Flexible Services and Manufacturing Journal*, 28:461–494.