electronics

MDPI

Article

Task-Aware Closed-Loop Communication Delay Analysis and Optimization for Intelligent Machine Networks

Yimeng Wang [1], Zhiqing Wei [1,*], Tao Zhang [2], Yinglong Fan [1], Haotian Liu [1], Xiaohan Chen [1] and Ying Zhou [1]

[1] Key Laboratory of Universal Wireless Communications, Ministry of Education, Beijing University of Posts and Telecommunications, Beijing 100876, China; wang.yi.meng@bupt.edu.cn (Y.W.); fanyinglong@bupt.edu.cn (Y.F.); haotian_liu@bupt.edu.cn (H.L.); xhchen@bupt.edu.cn (X.C.); zhouying9705@bupt.edu.cn (Y.Z.)

[2] China Unicom Joint Innovation Center, Beijing 100048, China; zhangt91@chinaunicom.cn

* Correspondence: weizhiqing@bupt.edu.cn

Abstract

Intelligent machine networks provide wireless connectivity for automated guided vehicles (AGVs) in flexible final-assembly workshops. AGVs switch between assembly and non-assembly tasks with different closed-loop interaction periods, which changes packet-generation rates and creates time-varying uplink and downlink traffic demands. Such traffic asymmetry makes fixed or periodic time-division duplexing (TDD) configurations difficult to adapt to task variations. This paper proposes a task-aware closed-loop communication delay analysis and dynamic TDD resource allocation framework for multi-AGV networks. First, the relationship between AGV task states, interaction periods, and uplink/downlink packet arrival rates is established. Finite-blocklength transmission theory and M/M/1 queueing approximations are then employed to characterize transmission and queueing delays and obtain the overall closed-loop communication delay. Based on the task-dependent traffic demands, a reference TDD slot ratio is determined to reflect the required uplink and downlink resources. An event-triggered allocation method is further developed under a limited TDD reconfiguration budget, where the slot ratio is updated only when its deviation from the reference ratio exceeds an offline-selected threshold. To further enhance resource allocation adaptability, a PPO-based method is introduced to dynamically adjust the uplink–downlink slot ratio according to traffic demands, queue conditions, and the remaining reconfiguration budget. Numerical results demonstrate that both proposed methods reduce the average closed-loop communication delay compared with offline-optimized fixed-ratio and periodic allocation. At $N = 21$, PPO achieves 1.04770 ms, compared with 1.112 ms for the additional advantage actor–critic (A2C) baseline, corresponding to a 5.78% reduction. PPO also reduces the delay by 19.65% relative to Optimization-Fixed. These results show that adapting TDD resources to task-induced traffic variations can improve multi-AGV closed-loop communication performance under the evaluated settings and limited reconfigurations.

Keywords: intelligent machine networks; task-state switching; task-aware modeling; closed-loop communication delay; dynamic TDD; event triggering; finite blocklength; proximal policy optimization

Check for updates

Academic Editor: Toshifumi Moriyama

Received: 24 September 2026
Revised: 23 October 2026
Accepted: 26 October 2026
Published: 28 October 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Sixth-generation (6G) communication networks are emerging as a key enabler of Industry 4.0 and smart factories, providing low-latency and highly reliable connectivity for

automated guided vehicles (AGVs), industrial robots, and shopfloor devices on production lines and in warehousing, logistics, and assembly workshops [1–3]. Related research on intelligent reflection also highlights the integration of communication, sensing, and security for future Internet-of-Everything networks [4]. In a typical intelligent manufacturing scenario, an AGV uploads the motion and status data required by its current task to a base station and edge server. The edge server processes these data and returns feedback or control commands over the downlink, forming a closed-loop communication process [5]. The resulting closed-loop communication delay directly affects multi-AGV coordination efficiency, operational safety, and system stability and is therefore an important performance metric for industrial wireless networks [6].

Existing research on low-latency industrial wireless communication primarily addresses short-packet transmission delay, end-to-end delay, and queueing delay. Link-level studies investigate the relationship among packet length, reliability, and transmission delay [7]; end-to-end models characterize the delay components from packet generation to network reception [8]; and queueing-theoretic approaches analyze ultra-reliable low-latency communication (URLLC) waiting time through arrival and service processes [9]. Closed-loop communication delay has also been modeled for industrial wireless networks [10]. However, some existing studies adopt predefined traffic arrival characteristics or simplified operating states, and the communication consequences of AGV task-state switching have received relatively limited attention. In particular, the chain from task state to interaction period, uplink/downlink traffic demand, and closed-loop communication delay has not been fully characterized.

In a flexible automotive final-assembly workshop, AGVs switch between assembly and non-assembly tasks as production progresses. Assembly tasks require frequent coordination with robotic arms and production equipment and therefore have short closed-loop interaction periods. Non-assembly tasks mainly involve transport, buffering, or waiting and permit longer interaction periods. A task-state switch consequently changes packet-generation frequency and aggregate uplink and downlink demand, creating dynamic traffic asymmetry in a TDD system. Because the uplink/downlink slot configuration directly affects closed-loop communication performance [11], the slot allocation should adapt to these task-induced demand changes. Existing fixed and semi-static TDD configurations [12,13] cannot readily match such variations. Flexible-duplex operation has also been investigated in cell-free low-altitude wireless networks for secure uplink communications [14]; although the architecture and optimization objectives differ from those considered here, this study illustrates the broader interest in adaptive duplex configurations. Periodic updates alleviate resource mismatch, but predetermined update times may respond late to a task switch and may also reconfigure the system when the demand change is small.

To address these issues, this paper develops a task-aware closed-loop communication delay model for flexible automotive final-assembly workshops. The assembly and non-assembly task states are mapped to their corresponding interaction periods and uplink/downlink arrival rates, thereby characterizing the task-dependent communication demands of AGVs. Finite-blocklength transmission and M/M/1 queueing then characterize how the resulting task-induced traffic affects transmission delay, queueing delay, and average closed-loop communication delay.

The methodological contribution lies in the explicit mapping from industrial task requirements to communication demands and its use in TDD allocation under a finite reconfiguration budget. In contrast to a traffic-aware formulation that takes arrival rates and packet sizes as given inputs, the proposed formulation derives these quantities from assembly and non-assembly task states through their interaction periods and uplink/downlink packet sizes. These quantities determine the transmission and queueing delays and the

demand-based reference slot ratio, linking task changes to both the timing and the target of resource reconfiguration. Finite-blocklength theory, the M/M/1 approximation, and PPO are established tools used to implement this framework; no new coding theorem, queueing theory, or PPO update rule is claimed.

Building on the task-to-traffic mapping, we treat the TDD uplink/downlink slot ratio as an adjustable resource variable and propose an event-triggered allocation method subject to a maximum update-count constraint. The method derives a reference slot ratio from the task-induced uplink and downlink demands and selects a best-performing triggering threshold from a predefined candidate set through offline search for the considered operating setting. Reconfiguration occurs only when the deviation between the reference and current ratios exceeds the threshold, allowing the system to respond to larger demand changes while limiting update frequency.

Although event-triggered allocation improves resource flexibility, its fixed threshold is generally determined from an offline statistical environment and cannot adapt its decision boundary to traffic load, channel quality, instantaneous queue-load ratios, and the remaining update budget. A single threshold may therefore be suboptimal when these quantities vary. The dynamic TDD slot-ratio problem has a sequential decision structure: the ratio selected at the current time determines the instantaneous transmission and service rates and hence the current closed-loop communication delay, while the selected configuration persists until the next effective update. Moreover, every effective reconfiguration consumes part of the finite update budget, so an early decision restricts the feasible decisions available later and may affect the long-term average delay [15]. Consequently, the decision process is temporally coupled through the retained slot ratio, elapsed time since the last update, and remaining reconfiguration opportunities, making reinforcement learning applicable to this problem [16].

The state space contains task-induced uplink and downlink demands, the current and reference slot ratios, channel conditions, instantaneous queue-load ratios, and the remaining update budget, thereby summarizing the variables used by the policy at each decision step. The discrete action space contains an action that retains the current configuration and actions that switch to candidate slot ratios, jointly representing whether to reconfigure and which ratio to adopt. This combination of multidimensional state information and a finite discrete action set can be handled by an actor–critic policy. In particular, proximal policy optimization (PPO) uses a clipped policy objective to limit excessively large policy updates and improve training stability in dynamic sequential decision problems [17].

Accordingly, we formulate dynamic TDD resource allocation as a sequential decision problem with aggregate observations and develop a PPO-based joint slot-ratio optimization method. The task-aware model first converts the current task configuration into uplink and downlink traffic demands. Together with the current and reference slot ratios, channel conditions, instantaneous queue-load ratios, and remaining update budget, these demands form the basis for sequential decisions. The learned policy jointly determines whether resources should be reconfigured and which slot ratio should be adopted, subject to slot-ratio bounds, the safety-load constraint, and the maximum number of updates.

The main contributions are as follows:

1. A task-to-communication mapping is developed for multi-AGV closed-loop operation in a flexible final-assembly workshop. Assembly and non-assembly task states determine interaction periods and uplink/downlink packet sizes, which are translated into arrival rates, service rates, and aggregate traffic demands. This mapping provides a task-dependent basis for closed-loop delay analysis and TDD reconfiguration under a finite update budget.

2. To mitigate the uplink/downlink resource mismatch caused by task-induced traffic changes while avoiding frequent reconfigurations, an event-triggered dynamic TDD slot-ratio allocation method is proposed under a maximum update-count constraint. An offline search over a predefined candidate set selects the best-performing threshold for the considered operating setting, and the slot ratio is updated when the resulting deviation condition is satisfied.
3. To overcome the limited state dependence of a fixed triggering threshold and avoid directly adopting the reference ratio after each trigger, a PPO-based joint slot-ratio optimization method is proposed. By considering task-induced traffic demands, channel conditions, instantaneous queue-load ratios, and the remaining update budget, the method jointly determines whether to reconfigure resources and which slot ratio to adopt for long-term delay minimization.

The remainder of this paper is organized as follows. Section 2 introduces the system model and defines the closed-loop communication process for multi-AGV networks. Section 3 develops the task-aware closed-loop communication delay model by establishing the mapping from AGV task states to interaction periods and uplink/downlink traffic arrival rates and derives the corresponding transmission and queueing delays. Section 4 presents the event-triggered dynamic TDD slot-ratio allocation method under a limited reconfiguration budget. Section 5 formulates dynamic TDD allocation as an MDP and develops the PPO-based joint slot-ratio optimization method. Section 6 presents the simulation settings and performance evaluation. Section 7 discusses the experimental observations, the roles of the proposed modules, and the limitations of the current model. Finally, Section 8 concludes the paper and outlines directions for future work.

2. System Model

Consider a flexible automotive final-assembly workshop consisting of N AGVs, a production task-management system, shopfloor data-acquisition devices, a wireless base station, an edge server, and production control units. Each AGV switches between two task states according to the production schedule and its current operating area: an assembly state and a non-assembly state. During an assembly task, the AGV frequently coordinates with robotic arms and other production equipment and therefore requires short interaction periods and frequent status-and-feedback exchanges. During a non-assembly task, the AGV mainly performs transport, buffering, or waiting operations and can use a longer interaction period. Consequently, task-state switching changes packet-generation frequency and produces time-varying uplink and downlink traffic loads.

For AGV n, let $\mathcal{M}_n(t) = \{1, \ldots, M\}$ denote the set of M sensing devices participating in its observation at time t. During each closed-loop interaction, every sensing device $m \in \mathcal{M}_n(t)$ observes the motion and status of the same AGV n. The task-relevant observations are then packaged and transmitted to the base station over the uplink. The base station forwards the packets to the edge server for data processing and command generation, and the resulting feedback is delivered to AGV n over the downlink. Thus, the sensing devices provide the data required for task execution, whereas the task state determines the interaction requirement and corresponding traffic load. The wireless network thereby completes a closed-loop process consisting of uplink transmission, edge processing, and downlink feedback, as shown in Figure 1.

As shown in Figure 1, the blue and orange regions represent the non-assembly and assembly areas, respectively, and the illustrated equipment includes the base station, sensors, AGVs, robotic arms, and storage racks.

In the proposed delay model, uplink and downlink communication are closely coupled. The closed-loop communication process is modeled as a tandem queueing system

comprising uplink and downlink queues. In this paper, closed-loop communication delay refers specifically to the communication-domain latency accumulated over the uplink state-data transmission and downlink feedback delivery paths; processing and actuation delays are outside the scope of the model. Accordingly, the closed-loop communication delay of AGV n at time t consists of uplink transmission delay, uplink queueing delay, downlink transmission delay, and downlink queueing delay [8]:

$$T_n^{\text{loop}}(t) = T_{n,t}^u(t) + T_{n,q}^u(t) + T_{n,t}^d(t) + T_{n,q}^d(t). \tag{1}$$

Here, $T_{n,q}^u(t)$ and $T_{n,q}^d(t)$ denote uplink and downlink queueing delays, respectively, and $T_{n,t}^u(t)$ and $T_{n,t}^d(t)$ denote the corresponding transmission delays.

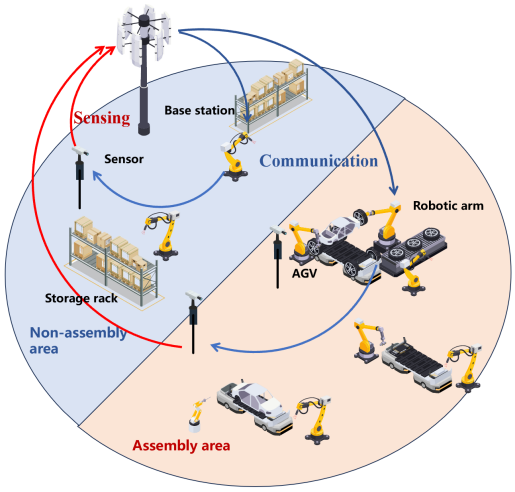


Figure 1. System model of a flexible automotive final-assembly workshop.

3. Task-Aware Closed-Loop Communication Delay Analysis

This section establishes the task-aware delay model used by the subsequent optimization methods. The central relationship is that a switch between assembly and non-assembly tasks changes the closed-loop interaction period, which changes uplink and downlink packet arrival rates and hence the traffic load presented to the dynamic TDD system. Finite-blocklength transmission and M/M/1 queueing are then used to translate the task-dependent load and slot ratio into per-AGV and system-average closed-loop communication delays.

3.1. Task-State and Arrival-Rate Mapping

In the workshop, multiple sensing devices observe the motion state of each AGV. Suppose that M sensors jointly observe the same AGV n during each closed-loop interaction. The observation obtained by sensor m at time t is [18]

$$X_n^m(t) = X_n(t) + N_n^m(t), \qquad 1 \le m \le M, \tag{2}$$

where $X_n(t)$ denotes the underlying motion state of AGV n and $N_n^m(t)$ denotes the observation noise of sensor m. The AGV's own observation is denoted by $X_n^0(t)$ and is assumed to follow the same observation relationship. To obtain a tractable multisource estimate, we assume that all $M+1$ observations are measurements of the same underlying state and that the observation noises $\{N_n^m(t)\}_{m=0}^M$ are mutually independent, zero-mean Gaussian variables with the same variance, i.e., $N_n^m(t) \sim \mathcal{N}(0, \sigma^2)$. Under these assumptions, each observation has the same statistical reliability, and the maximum-likelihood estimate can be expressed as the arithmetic mean of the available observations [18]:

$$\widehat{X}_n(t) = \frac{1}{M+1} \sum_{m=0}^{M} X_n^m(t). \tag{3}$$

The resulting multisource estimate $\widehat{X}_n(t)$ represents the estimated motion state of AGV n at time t. In the considered workshop, the operating area corresponds to the task currently performed by the AGV: an AGV located in the assembly area performs assembly-related coordination, whereas an AGV in the non-assembly area mainly performs transport, buffering, or waiting tasks. Therefore, the estimated state is used to identify the current task state. Define the task-state indicator as

$$z_n(t) = \begin{cases} 1, & \widehat{X}_n(t) \in \Omega_{\text{asm}}, \\ 0, & \widehat{X}_n(t) \in \Omega_{\text{non}}, \end{cases} \tag{4}$$

where Ω_{asm} and Ω_{non} denote the assembly and non-assembly areas, respectively. Accordingly, $z_n(t) = 1$ represents the assembly task state, whereas $z_n(t) = 0$ represents the non-assembly task state. These two task states impose different communication requirements. During an assembly task, the AGV frequently coordinates with robotic arms and production equipment and therefore requires timely status updates, feedback, and a short interaction period. During a non-assembly task, the communication demand is lower, allowing a longer interaction period. Thus, the interaction period of AGV n at time t is

$$T_{\text{ns},n}(t) = z_n(t)T_{\text{asm}} + \big(1 - z_n(t)\big)T_{\text{non}}, \tag{5}$$

where T_{asm} and T_{non} are the interaction periods associated with assembly and non-assembly task states, respectively.

The interaction period $T_{\text{ns},n}(t)$ determines the packet-generation frequency, whereas the index t denotes a resource-allocation decision step. Each decision step corresponds to a physical interval of $\Delta_{\text{dec}} = 5$ ms. The simulation horizon consists of $T = 2000$ decision steps, during which the service rates and communication delays are evaluated using the equivalent TDD resource-allocation model.

The mapping in (5) converts the discrete task state into a communication requirement. A task switch changes $T_{\text{ns},n}(t)$ from T_{asm} to T_{non}, or vice versa, and therefore changes the packet-generation frequency. Assuming that AGV n sends one uplink status packet to the base station in each interaction period, its uplink arrival rate is

$$\lambda_{u,n}(t) = \frac{1}{T_{\text{ns},n}(t)}. \tag{6}$$

The AGV's status information is sent to the controller over the uplink. The controller generates a control command from the received information and returns it over the downlink [19,20]. We assume that each uplink status packet corresponds to one downlink feedback packet. The packet arrivals in the two directions are therefore in one-to-one correspondence, and their packet arrival rates are approximated as equal. The downlink arrival rate is

$$\lambda_{d,n}(t) = \frac{1}{T_{\text{ns},n}(t)}. \tag{7}$$

For the task-aware resource-allocation evaluation, the generated task indicators $z_n(t)$ are supplied to the allocation methods at each decision step. The resulting interaction periods and packet sizes are used consistently to calculate traffic demands, the reference slot ratio, and queue-load indicators. This establishes a common task-information basis for comparing the resource-allocation schemes.

3.2. Closed-Loop Communication Delay Analysis

To accommodate time-varying industrial AGV traffic, we introduce a dynamic TDD slot-ratio parameter. Let $\eta(t)$ denote the uplink slot fraction at time t, with $0 < \eta(t) < 1$. The corresponding downlink fraction is $1 - \eta(t)$. For a total system bandwidth B, the equivalent uplink and downlink bandwidths are

$$B_u(t) = \eta(t)B, \qquad B_d(t) = \big(1 - \eta(t)\big)B. \tag{8}$$

Under equal resource allocation among devices, the equivalent bandwidths assigned to AGV n are

$$B_{u,n}(t) = \frac{\eta(t)B}{N}, \qquad B_{d,n}(t) = \frac{\big(1 - \eta(t)\big)B}{N}. \tag{9}$$

The bandwidth representation in (8) and (9) expresses the time-averaged service capacity of a TDD system. For active-direction spectral efficiency $r_{u,n}$ and bandwidth B/N, a TDD cycle of duration Δ_f allocates $\eta\Delta_f(B/N)r_{u,n}$ uplink information bits, yielding the equivalent mean rate $\eta(B/N)r_{u,n}$. Correspondingly, $\beta_n/R_{u,n}$ denotes the effective uplink transmission duration under the average-service representation. This formulation provides a consistent basis for evaluating different slot fractions through their resulting uplink and downlink service rates.

Closed-loop communication in flexible automotive final-assembly workshops typically involves short, delay-sensitive packets. In particular, AGVs performing assembly tasks frequently upload status data and require prompt feedback from the edge server. The asymptotic Shannon-capacity expression may become inaccurate for the short packets considered in industrial closed-loop communication. We therefore model achievable uplink and downlink rates using finite-blocklength transmission theory, drawing on existing short-packet and finite-blocklength studies [21–23]. The uplink rate of AGV n at time t is approximated by

$$R_{u,n}(t) = B_{u,n}(t)\left[\log_2\big(1 + \mathrm{SNR}_{u,n}(t)\big) - \sqrt{\frac{V_{u,n}(t)}{S_u}}Q^{-1}(\varepsilon)\right]. \tag{10}$$

Here, $\mathrm{SNR}_{u,n}(t)$ is the uplink signal-to-noise ratio (SNR), S_u is the uplink blocklength, ε is the block error rate, and $Q^{-1}(\cdot)$ is the inverse Gaussian tail function. The uplink channel dispersion is [24]

$$V_{u,n}(t) = \left[1 - \frac{1}{\big(1 + \mathrm{SNR}_{u,n}(t)\big)^2}\right](\log_2 e)^2. \tag{11}$$

Similarly, the downlink rate is

$$R_{d,n}(t) = B_{d,n}(t)\left[\log_2\big(1 + \mathrm{SNR}_{d,n}(t)\big) - \sqrt{\frac{V_{d,n}(t)}{S_d}}Q^{-1}(\varepsilon)\right], \tag{12}$$

where $\mathrm{SNR}_{d,n}(t)$ is the downlink SNR, S_d is the downlink blocklength, and the downlink channel dispersion is [24]

$$V_{d,n}(t) = \left[1 - \frac{1}{\big(1 + \mathrm{SNR}_{d,n}(t)\big)^2}\right](\log_2 e)^2. \tag{13}$$

The parameters S_u and S_d are codeword blocklengths in channel uses, and ε denotes the target block error probability per codeword. For example, at an uplink SNR of 15 dB with $S_u = 512$ and $\varepsilon = 10^{-5}$, the spectral-efficiency expression in (10) gives approximately

4.756 bits per channel use. The uplink and downlink transmission durations in (16) are then evaluated as the packet size divided by the corresponding effective transmission rate. This rate-based formulation consistently incorporates finite-blocklength spectral efficiency into the proposed task-aware delay model.

The service rates follow from the achievable rates and packet sizes. Let $\beta_n(t)$ and $\gamma_n(t)$ denote the uplink and downlink packet sizes of AGV n, respectively. The corresponding service rates are

$$\mu_{u,n}(t) = \frac{R_{u,n}(t)}{\beta_n(t)}, \qquad \mu_{d,n}(t) = \frac{R_{d,n}(t)}{\gamma_n(t)}. \tag{14}$$

Queue stability requires

$$\lambda_{u,n}(t) < \mu_{u,n}(t), \qquad \lambda_{d,n}(t) < \mu_{d,n}(t). \tag{15}$$

The uplink transmission delay is the wireless transmission time required to deliver an AGV state packet to the base station. The downlink transmission delay is the time required to deliver feedback from the base station to the AGV. These delays are [25]

$$T^u_{n,t}(t) = \frac{\beta_n(t)}{R_{u,n}(t)}, \qquad T^d_{n,t}(t) = \frac{\gamma_n(t)}{R_{d,n}(t)}. \tag{16}$$

To account for queueing in the AGV state-data and base-station feedback paths, the uplink and downlink queues are, for analytical tractability, approximated as locally stationary M/M/1 systems characterized by state-dependent mean arrival and service rates. Within each decision interval, $\lambda_{u,n}(t)$, $\lambda_{d,n}(t)$, $\mu_{u,n}(t)$, and $\mu_{d,n}(t)$ are treated as constant, while their values may change across intervals with the AGV task state, channel condition, and TDD slot ratio. The M/M/1 approximation characterizes the local queueing behavior through the task-dependent mean arrival rates and effective service rates, yielding analytically tractable waiting-time expressions. Under this local-stationarity approximation and the queue-stability conditions, the mean queueing delays are [26]

$$T^u_{n,q}(t) = \frac{\lambda_{u,n}(t)}{\mu_{u,n}(t)\big(\mu_{u,n}(t) - \lambda_{u,n}(t)\big)}, \tag{17}$$

$$T^d_{n,q}(t) = \frac{\lambda_{d,n}(t)}{\mu_{d,n}(t)\big(\mu_{d,n}(t) - \lambda_{d,n}(t)\big)}. \tag{18}$$

Equations (17) and (18) evaluate the mean queueing delay for the uplink and downlink using the current task-dependent arrival and service rates. At each resource-allocation step, these rates are updated according to the task states, channel conditions, and TDD slot ratio, allowing queueing delay to be incorporated into the dynamic closed-loop delay objective.

Combining the four delay components gives

$$\begin{aligned} T^{\text{loop}}_n(t) = {} & \frac{\beta_n(t)}{R_{u,n}(t)} + \frac{\lambda_{u,n}(t)}{\mu_{u,n}(t)\big(\mu_{u,n}(t) - \lambda_{u,n}(t)\big)} \\ & + \frac{\gamma_n(t)}{R_{d,n}(t)} + \frac{\lambda_{d,n}(t)}{\mu_{d,n}(t)\big(\mu_{d,n}(t) - \lambda_{d,n}(t)\big)}. \end{aligned} \tag{19}$$

Averaging over N AGVs and an observation horizon of length T yields the system-average closed-loop communication delay:

$$\overline{T}^{\text{loop}} = \frac{1}{TN} \sum_{t=1}^{T} \sum_{n=1}^{N} T^{\text{loop}}_n(t). \tag{20}$$

4. Event-Triggered Slot-Ratio Allocation

This section proposes an event-triggered slot-ratio allocation method that follows the task-to-traffic mapping established above. At each time step, the current task states $z_n(t)$ determine the interaction periods and arrival rates, which in turn determine the aggregate uplink and downlink demands. The slot ratio is treated as an adjustable resource variable, and an event-triggering metric measures the deviation between the task-induced reference ratio and the current configuration. The method searches over a predefined candidate threshold set under the update-count constraint and selects the threshold that yields the lowest average closed-loop communication delay for the specified simulation setting and generated random trajectories, while limiting resource reconfigurations.

The system first computes a reference uplink slot fraction from current traffic demands. Define the aggregate uplink and downlink demands at time t as

$$D_u(t) = \sum_{n=1}^{N} \lambda_{u,n}(t)\beta_n(t), \qquad D_d(t) = \sum_{n=1}^{N} \lambda_{d,n}(t)\gamma_n(t). \tag{21}$$

The reference uplink slot fraction is

$$\eta^{\mathrm{ref}}(t) = \frac{D_u(t)}{D_u(t) + D_d(t)}. \tag{22}$$

The reference ratio therefore connects task-state switching to TDD resource allocation through the chain $z_n(t) \to T_{\mathrm{ns},n}(t) \to \lambda_{u,n}(t), \lambda_{d,n}(t) \to D_u(t), D_d(t) \to \eta^{\mathrm{ref}}(t)$. Higher uplink demand increases $\eta^{\mathrm{ref}}(t)$, favoring more uplink slots, whereas higher downlink demand reduces it and reserves more resources for the downlink.

To prevent frequent reconfiguration due to minor load fluctuations, we introduce an event-triggering threshold θ. Let $\eta(t-1)$ denote the uplink fraction in use before the decision at time t. A reconfiguration is triggered when the deviation between the reference and current ratios exceeds θ and the cumulative number of updates remains below the maximum. The update rule is

$$\eta(t) = \begin{cases} \eta^{\mathrm{ref}}(t), & \left|\eta^{\mathrm{ref}}(t) - \eta(t-1)\right| > \theta, \quad |U(\theta)| < K_{\max}, \\ \eta(t-1), & \text{otherwise}, \end{cases} \tag{23}$$

where $U(\theta)$ is the set of slot-ratio configuration times generated under threshold θ, $|U(\theta)|$ is the number of counted configurations, and $K_{\max}$ is the maximum permitted number. Consistent with the simulation implementation, the initial task-based configuration is included as the first counted update, and the count used in the triggering condition refers to configurations completed before the current decision. Once $\eta(t)$ is obtained, the system-average delay can be evaluated. The optimization problem is

$$\begin{aligned} \textbf{P1:} \quad & \min_{\theta \in \Theta} \quad \overline{T}^{\mathrm{loop}} \\ \text{s.t.} \quad & \eta_{\min} \le \eta(t) \le \eta_{\max}, && \forall t, \\ & \mu_{u,n}(t) > \lambda_{u,n}(t), && \forall n, t, \\ & \mu_{d,n}(t) > \lambda_{d,n}(t), && \forall n, t, \\ & |U(\theta)| \le K_{\max}. \end{aligned} \tag{24}$$

The first constraint bounds the feasible uplink slot fraction. The limits $\eta_{\min}$ and $\eta_{\max}$ prevent either direction from receiving excessively few resources. The second and third constraints

ensure stable uplink and downlink queues by requiring service rates to exceed arrival rates. The fourth constraint limits the total number of updates.

For a given θ, the system generates the corresponding update-time set $U(\theta)$ and slot-ratio sequence $\eta(t)$, from which the average delay is calculated. For the considered candidate set and evaluation setting, the selected threshold is

$$\theta^* = \arg\min_{\theta \in \Theta} \overline{T}^{\text{loop}}(\theta), \qquad \text{s.t. } |U(\theta)| \leq K_{\max}, \tag{25}$$

with candidates subject to the feasibility constraints in (24). The slot-ratio sequence associated with the selected threshold is

$$\eta^*(t) = \eta_{\theta^*}(t). \tag{26}$$

Algorithm 1 summarizes the proposed method. During the offline stage, candidate thresholds are evaluated by generating slot-ratio sequences from AGV task states and current traffic demands. Reconfiguration occurs only when the reference-to-current deviation exceeds the threshold and the update budget permits it. Among feasible candidates, the threshold yielding the minimum average delay for the specified simulation setting and generated random trajectories is selected. During online operation, this threshold governs event-triggered allocation, adapting uplink and downlink resources to task-induced traffic changes with a limited number of reconfigurations.

Algorithm 1 Event-Triggered slot-ratio allocation.

Offline stage: candidate-threshold search
for $\theta \in \Theta$ **do**
  Initialize $\eta_\theta(0)$ from the initial task demands and set $|U(\theta)| = 1$.
  for $t = 1, \ldots, T$ **do**
    for $n = 1, \ldots, N$ **do**
      Determine $z_n(t)$ and calculate the task-dependent arrival rates and traffic demands of AGV n.
    end for
    Calculate $\eta^{\text{ref}}(t)$ from aggregate uplink and downlink demands.
    if $|\eta^{\text{ref}}(t) - \eta_\theta(t-1)| > \theta$ **and** $|U(\theta)| < K_{\max}$ **then**
      Set $\eta_\theta(t) = \eta^{\text{ref}}(t)$; add t to $U(\theta)$.
    else
      Keep the current slot ratio: $\eta_\theta(t) = \eta_\theta(t-1)$.
    end if
    Calculate the closed-loop communication delay of every AGV.
  end for
  Calculate $\overline{T}^{\text{loop}}(\theta)$.
  if $|U(\theta)| \leq K_{\max}$ and the candidate is feasible and minimizes delay **then**
    Set $\theta^* = \theta$.
  end if
end for
Online stage: event-triggered slot-ratio allocation
Apply the event-triggered allocation rule using θ^*.
Output: θ^*, $\eta_{\theta^*}(t)$, and $\overline{T}^{\text{loop}}(\theta^*)$.

5. PPO-Based Joint Slot-Ratio Optimization

The event-triggered method selects a threshold θ^* from a predefined candidate set through offline search for the considered operating setting. It determines reconfiguration from the deviation between the current and reference slot ratios and adopts $\eta^{\text{ref}}(t)$ when triggered, providing an interpretable allocation rule with low online computational complexity. The PPO formulation further extends this structure by jointly selecting the update timing and candidate slot ratio according to the observed system state.

To enable state-dependent joint decisions, we use an underlying Markov decision process (MDP) description while recognizing the partial observability of the aggregate policy inputs [16] and propose a PPO-based joint optimization method. Task-state switching

is first translated into traffic demands through the task-aware model; the policy then combines these task-induced demands with the current configuration, channel conditions, instantaneous load ratios, and remaining update budget to decide whether to reconfigure resources and which ratio to select. This extends fixed-threshold allocation to sequential decisions that respond to the communication consequences of task changes. Figure 2 illustrates the overall framework.

As shown in Figure 2, the actor maps the normalized 15-dimensional state to a categorical distribution over the keep action and candidate slot ratios, while the critic estimates the state value $V_\psi(s_t)$. The communication environment executes the selected slot ratio after feasibility checks, evaluates $T_{\text{keep}}(t)$ and $T_{\text{act}}(t)$, and calculates the reward according to (38). The collected interaction trajectories are then used for offline PPO training and policy updates.

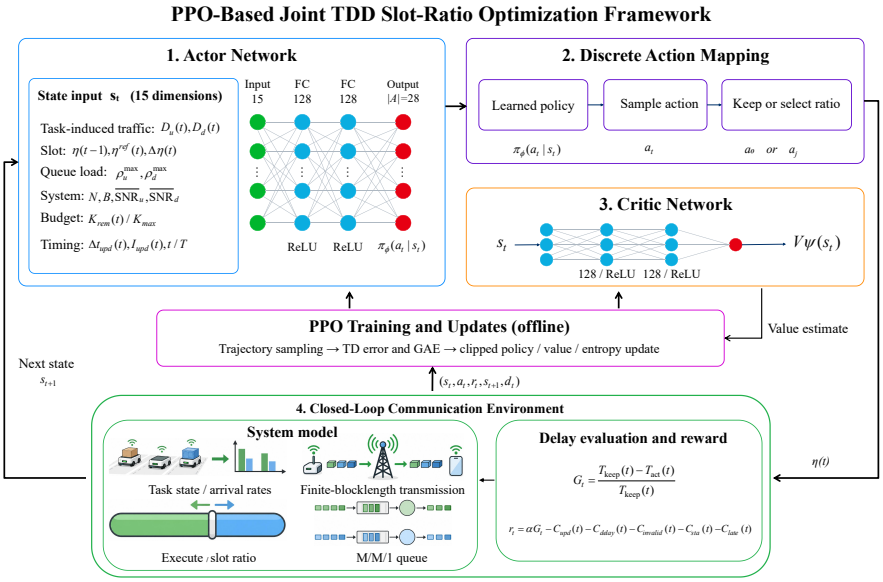


Figure 2. PPO-based framework for joint TDD slot-ratio optimization, showing the actor, discrete action mapping, critic, communication environment, and offline training flow. FC denotes a fully connected layer, ReLU denotes the rectified linear unit activation function, and TD denotes temporal difference.

5.1. Optimization Problem

Problem P1 indirectly generates a slot-ratio sequence for the entire horizon by selecting a fixed threshold θ^* from the predefined candidate set for the considered operating setting. During online operation, the triggering condition is

$$\left|\eta^{\text{ref}}(t) - \eta(t-1)\right| > \theta^*. \tag{27}$$

The fixed triggering rule defines a common reference-deviation criterion for the event-triggered method. Dynamic TDD allocation also exhibits temporal dependence [27]: a reconfiguration at time t affects the current transmission and queueing delays, consumes part of the update budget, and determines the configuration retained at subsequent steps. PPO incorporates this temporal relationship through its state-dependent resource-allocation policy.

In this sequential decision formulation, the agent selects action a_t from state s_t at each slot. After the action is executed, the communication environment evaluates the resulting closed-loop communication delay using the task-aware model established in Section 3, calculates the immediate reward r_t, and transitions to s_{t+1} [28]. Through repeated interaction, the agent learns the state-to-action policy [17]

$$\pi_\phi(a \mid s_t), \tag{28}$$

where ϕ denotes the actor-network parameters. Unlike P1, which optimizes a single offline threshold and only indirectly determines the slot-ratio sequence, the MDP formulation optimizes a state-dependent policy that directly selects resource-allocation actions according to the current system state. This policy-based formulation represents the temporal coupling among the current closed-loop communication delay, the slot ratio retained for subsequent steps, and the remaining reconfiguration budget. Accordingly, P1 is generalized to the following policy-optimization problem:

$$\textbf{P2:} \qquad \min_{\pi_\phi \in \Pi} \ \overline{T}^{\text{loop}}_{\pi_\phi}, \tag{29}$$

where

$$\overline{T}^{\text{loop}}_{\pi_\phi} = \frac{1}{TN} \sum_{t=1}^{T} \sum_{n=1}^{N} T_n^{\text{loop}}\left(\eta_{\pi_\phi}(t)\right), \tag{30}$$

subject to

$$\begin{aligned} \eta_{\min} \le \eta_{\pi_\phi}(t) \le \eta_{\max}, \quad & \forall t, \\ \rho_{u,n}(t;\eta_j) < \rho_{\text{safe}}, \quad & \forall n, t \text{ for accepted update actions}, \\ \rho_{d,n}(t;\eta_j) < \rho_{\text{safe}}, \quad & \forall n, t \text{ for accepted update actions}, \\ |U_{\pi_\phi}| \le K_{\max}. & \end{aligned} \tag{31}$$

Here, Π is the PPO policy space, $\eta_{\pi_\phi}(t)$ is the uplink fraction actually executed after action mapping and feasibility checks under policy π_ϕ in state s_t, and U_{π_ϕ} is the set of effective update times generated by the policy over the decision horizon. In the implementation, a requested update action is accepted only if its candidate ratio satisfies $\rho_{\text{safe}} = 0.90$ in both directions, which is stricter than the theoretical M/M/1 stability boundary $\rho < 1$. Because channel conditions can change while a configuration is retained, the execution layer separately checks whether preventive recovery is required.

5.2. State Space

To represent the industrial communication environment, the PPO state combines task-induced traffic, resource configuration, channel conditions, instantaneous queue-load ratios, and reconfiguration constraints. Because the mapping in Section 3 converts all AGV task states into aggregate demands, $D_u(t)$ and $D_d(t)$ provide a compact representation of the communication effect of the current task configuration. The state at time t is

$$\begin{aligned} s_t = \big[& D_u(t), D_d(t), \eta^{\text{ref}}(t), \eta(t-1), \Delta\eta(t), \\ & K_{\text{rem}}(t)/K_{\max}, \Delta t_{\text{upd}}(t), I_{\text{upd}}(t), t/T, \\ & N, B, \overline{\text{SNR}}_u(t), \overline{\text{SNR}}_d(t), \rho_u^{\max}(t), \rho_d^{\max}(t) \big]. \end{aligned} \tag{32}$$

Here, $\Delta\eta(t) = |\eta^{\text{ref}}(t) - \eta(t-1)|$ is the reference-to-current deviation; $K_{\text{rem}}(t) = K_{\max} - K_{\text{used}}(t)$ is the remaining reconfiguration budget; $\Delta t_{\text{upd}}(t)$ is the number of decision steps since the last effective update; and $I_{\text{upd}}(t)$ is the binary availability indicator for an ordinary update request, reflecting the applicable update-interval and budget conditions.

To characterize the largest instantaneous M/M/1 load ratios under the current arrivals, channel realization, and slot ratio, define [26]

$$\rho_u^{\max}(t) = \max_n \frac{\lambda_{u,n}(t)}{\mu_{u,n}(t)}, \qquad \rho_d^{\max}(t) = \max_n \frac{\lambda_{d,n}(t)}{\mu_{d,n}(t)}. \tag{33}$$

The 15-dimensional observation s_t combines aggregate task-induced demand, channel-quality summaries, queue-load indicators, and resource-update information. This compact representation is used as input to the feedforward actor and critic, enabling the policy to select actions based on communication conditions and the remaining reconfiguration budget at each decision step.

Table 1 summarizes the state variables and their physical meanings.

Table 1. Components of the PPO state vector and their physical meanings.

State Variable	Physical Meaning
$D_u(t), D_d(t)$	Task-induced uplink and downlink traffic demands, reflecting traffic asymmetry.
$\eta^{\mathrm{ref}}(t)$	Reference uplink slot fraction derived from traffic demands.
$\eta(t-1)$	Uplink slot fraction currently in use.
$\Delta\eta(t)$	Deviation between reference and current slot ratios.
$K_{\mathrm{rem}}(t)/K_{\max}$	Fraction of the reconfiguration budget remaining.
$\Delta t_{\mathrm{upd}}(t)$	Number of decision steps since the last effective resource update.
$I_{\mathrm{upd}}(t)$	Indicator of ordinary-update availability under interval and budget conditions.
t/T	Normalized progress through the current episode.
N, B	Number of AGVs and total system bandwidth.
$\overline{\mathrm{SNR}}_u, \overline{\mathrm{SNR}}_d$	Average uplink and downlink SNRs.
$\rho_u^{\max}(t), \rho_d^{\max}(t)$	Maximum instantaneous uplink and downlink load ratios across the AGVs.

All 15 observation components are clipped to $[0,1]$ after normalization. For direction $q \in \{u,d\}$, demand is scaled as $(D_q - D_q^{\min})/(D_q^{\max} - D_q^{\min})$, with $D_q^{\min} = N\ell_{q,\mathrm{non}}/T_{\mathrm{non}}$ and $D_q^{\max} = N\ell_{q,\mathrm{asm}}/T_{\mathrm{asm}}$, where $\ell_u = \beta$ and $\ell_d = \gamma$. Current and reference ratios are scaled by $(\eta - 0.20)/0.65$, and their absolute difference by 0.65. The remaining budget is divided by $K_{\max}$; the update gap is scaled as $\min\{(t - t_{\mathrm{last}})/11, 1\}$; update availability is a binary flag; and episode progress is t/T. The remaining scales are $(N-1)/24$, $(B/10^6 - 15)/10$, $(\overline{\mathrm{SNR}}_{u,\mathrm{dB}} - 8)/12$, $(\overline{\mathrm{SNR}}_{d,\mathrm{dB}} - 11)/12$, and $\min\{\rho_q^{\max}/1.5, 1\}$. The code protects normalization denominators against zero. These fixed scaling rules provide consistent normalization for the training and evaluation states, with each normalized component bounded in $[0,1]$.

5.3. Action Space

A discrete action space is used to select the dynamic TDD slot ratio. Let $\mathcal{A}_\eta$ denote the set of J candidate slot ratios. The complete action space is

$$\mathcal{A} = \{a_0, a_1, \ldots, a_J\}. \tag{34}$$

Action a_0 retains the current configuration:

$$a_0: \quad \eta(t) = \eta(t-1). \tag{35}$$

Each remaining action a_j, $1 \le j \le J$, selects a specific candidate:

$$a_j: \quad \eta(t) = \eta_j \in \mathcal{A}_\eta. \tag{36}$$

A single discrete action therefore implements two decisions: whether the current state warrants reconfiguration and, if so, which target ratio to use. Whereas event-triggered allocation directly adopts $\eta^{\mathrm{ref}}(t)$ after triggering, PPO learns both whether to update and which ratio to select, jointly optimizing reconfiguration triggering and slot-ratio selection.

5.4. Reward Function

The evaluation objective is the average closed-loop communication delay in (29). During training, the implementation uses a shaped reward combining relative delay im-

provement, an absolute-delay term, and action penalties. Let $T_{\text{keep}}(t)$ and $T_{\text{act}}(t)$ be the mean per-AGV delays before and after the executed action in the same current task/channel configuration, measured in seconds. The implemented relative gain is

$$G_t = \text{clip}\left(\frac{T_{\text{keep}}(t) - T_{\text{act}}(t)}{\max\{T_{\text{keep}}(t), 10^{-9}\}}, -1, 1\right). \tag{37}$$

The reward is

$$\begin{aligned} r_t = 13G_t &- C_{\text{upd}}(t) - C_{\text{delay}}(t) \\ &- C_{\text{invalid}}(t) - C_{\text{tiny}}(t) - C_{\text{sta}}(t) \\ &- C_{\text{risk}}(t) - C_{\text{late}}(t). \end{aligned} \tag{38}$$

Let $I_{\text{exec}}(t) = 1$ if an update is actually executed during training and 0 otherwise, distinguishing the executed-update indicator from the update-availability indicator $I_{\text{upd}}(t)$ in Section 5.2. Let $b_t = K_{\text{used}}(t)/\max\{K_{\max}, 1\}$, with the used count evaluated after the action. The penalties are

$$\begin{aligned} C_{\text{upd}}(t) &= I_{\text{exec}}(t)(0.012 + 0.020b_t), \\ C_{\text{delay}}(t) &= 0.005 \min\{T_{\text{act}}(t)/(5 \times 10^{-3}), 4\}, \\ C_{\text{invalid}}(t) &= 0.030 I_{\text{invalid}}(t), \\ C_{\text{tiny}}(t) &= 0.010 I_{\text{tiny}}(t), \\ C_{\text{sta}}(t) &= 0.200 I_{\text{sta}}(t). \end{aligned}$$

For a requested candidate action, the checks are applied in order: $I_{\text{invalid}} = 1$ if the budget is exhausted or the elapsed update interval is not greater than 10; otherwise, $I_{\text{tiny}} = 1$ if the ratio change is less than 0.015; otherwise, $I_{\text{sta}} = 1$ if the candidate fails $\rho_{\max} < 0.90$. These flags are zero for a keep action, and rejected requests retain the current ratio. Thus, C_{sta} penalizes a rejected safety-margin request, which need not constitute an actual queue-stability violation. For the load ρ_t of the executed ratio, the implemented risk term is

$$C_{\text{risk}}(t) = \begin{cases} 0.020(\rho_t - 0.85)/0.05, & 0.85 < \rho_t < 0.90, \\ 0, & \text{otherwise.} \end{cases}$$

This piecewise term assigns a graduated penalty for loads between 0.85 and 0.90; candidate-action admission separately uses the safety-load check. The late-budget penalty is

$$C_{\text{late}}(t) = 0.025\, \mathbf{1}\{I_{\text{exec}}(t) = 1,\ K_{\text{used}}(t) \geq 0.70K_{\max},\ G_t < 0.004\}.$$

All reward coefficients are dimensionless. Relative improvement encourages beneficial adjustments, the absolute-delay term discourages high delays, and the remaining terms discourage budget consumption and rejected or low-benefit requests. The penalty terms shape the training objective by balancing relative delay reduction, absolute delay, and use of the reconfiguration budget. PPO optimizes the expected discounted reward with discount factor 0.999, while the policy is evaluated using the average closed-loop communication delay defined in P2.

The evaluated PPO configuration uses the reward weights in (38), $K_{\max} = 20$, and candidate slot ratios spaced by 0.025. The reward components encourage delay reduction while accounting for the effective-update budget, and the discrete action set supports joint selection of reconfiguration timing and target ratio.

5.5. Actor–Critic Networks and PPO Policy Updates

The PPO agent adopts an actor–critic architecture [17]. Both networks receive the normalized 15-dimensional state vector and use two fully connected hidden layers with 128 rectified linear unit (ReLU) activations per layer. The actor produces the categorical distribution $\pi_\phi(a_t \mid s_t)$ over the keep action and candidate slot ratios through a Softmax output layer [29], whereas the critic estimates the state value $V_\psi(s_t)$.

During training, trajectories of length H are collected using the current policy. The temporal-difference error and generalized advantage estimate are computed as [30]

$$\delta_t = r_t + \gamma(1-d_t)V_\psi(s_{t+1}) - V_\psi(s_t), \qquad A_t = \delta_t + \gamma\lambda_{\mathrm{GAE}}(1-d_t)A_{t+1}, \tag{39}$$

where d_t indicates episode termination, γ is the discount factor, and λ_{GAE} is the GAE parameter. With $R_t = A_t + V_\psi(s_t)$ and $q_t(\phi) = \pi_\phi(a_t \mid s_t)/\pi_{\phi_{\mathrm{old}}}(a_t \mid s_t)$, the clipped policy loss is [17]

$$L_{\mathrm{CLIP}} = -\mathbb{E}_t[\min(q_t A_t, \mathrm{clip}(q_t, 1-\delta_c, 1+\delta_c)A_t)], \tag{40}$$

where δ_c is the clipping parameter. The actor and critic are jointly updated using

$$L_{\mathrm{Total}} = L_{\mathrm{CLIP}} + \frac{c_1}{2}\mathbb{E}_t\Big[\big(V_\psi(s_t) - R_t\big)^2\Big] - c_2\mathcal{H}\big[\pi_\phi(\cdot \mid s_t)\big], \tag{41}$$

where c_1 and c_2 weight the value loss and entropy regularization, respectively. The collected samples are optimized over several mini-batch epochs. After offline training, the actor parameters are fixed for online slot-ratio decisions, while the execution layer continues to enforce the update budget, minimum update interval, minimum effective ratio change, and safety-load constraint.

5.6. PPO Algorithm Procedure

Algorithm 2 comprises offline policy training and online dynamic decision-making. Offline training begins by initializing the actor, critic, candidate slot-ratio set, and maximum reconfiguration count. At the start of each episode, AGV task-state and uplink/downlink channel trajectories are generated randomly. The initial task-based slot ratio is projected to a nearest safe candidate when available, or to a minimum-load candidate otherwise, and counted as the first configuration update. At each decision step t, the system calculates arrival rates and traffic demands from AGV task states and obtains $\eta^{\mathrm{ref}}(t)$. It then constructs s_t from the current ratio, traffic conditions, channel states, instantaneous load ratios, and remaining budget. The actor samples a keep action or a candidate-ratio action according to $\pi_\phi(a_t \mid s_t)$. For an update request, the environment checks the maximum update count, minimum update interval, minimum ratio change, and the $\rho < 0.90$ safety-load constraint. A feasible candidate is executed and counted as an effective reconfiguration; otherwise, the current configuration is retained.

The environment calculates $T_{\mathrm{act}}(t)$ and $T_{\mathrm{keep}}(t)$ and stores the interaction sample $(s_t, a_t, r_t, s_{t+1}, d_t)$ in the trajectory buffer. Once the experience horizon H is reached, the critic provides TD errors, GAE estimates the advantages, and the clipped PPO objective updates the actor and critic. Training continues over multiple episodes until the average reward stabilizes, yielding the trained policy π_ϕ^*. During evaluation, the actor parameters are fixed and actions are selected from the current simulated state. The evaluation executor checks for preventive recovery before constructing the policy observation. Define $\rho_{\max}(\eta, t) = \max_n\{\rho_{u,n}(\eta, t), \rho_{d,n}(\eta, t)\}$ and $\mathcal{F}(t) = \{\eta_j \in \mathcal{A}_\eta : \rho_{\max}(\eta_j, t) < 0.90\}$. When the current load reaches 0.95 and $K_{\mathrm{used}} < K_{\max}$, recovery selects the nearest candidate in $\mathcal{F}(t)$; if this set is empty, it selects a candidate minimizing $\rho_{\max}$ over the entire grid. A change of at least 0.015 is executed, counted against the budget, and recorded as the latest

update. Recovery bypasses the ordinary minimum-update-interval check. Once the budget is exhausted, the executor retains the current ratio according to the update-count constraint. The fallback selects the candidate having the lowest maximum load within the available grid. Ordinary policy updates require a safe candidate, remaining budget, an elapsed interval greater than 10 decision steps, and a ratio change of at least 0.015. The training stage applies the ordinary action-feasibility checks, while the evaluation executor additionally implements preventive recovery. Initialization is counted as the first configuration and uses the same nearest-safe or minimum-load fallback rule.

Algorithm 2 PPO-based joint TDD slot-ratio optimization.

Input: System parameters, $\mathcal{A}_\eta$, $K_{\max}$, number of training episodes E, and experience horizon H.
Output: Trained PPO policy π_ϕ^* and dynamic slot-ratio sequence $\eta^*(t)$.

Offline stage: PPO policy training
Initialize actor π_ϕ, critic V_ψ, $\mathcal{A}_\eta$, and PPO training parameters.
for $e = 1, \ldots, E$ **do**
  Randomly generate AGV task-state and uplink/downlink channel trajectories.
  Initialize $\eta(0)$ using the nearest-safe or minimum-load fallback rule, set $K_{\text{used}} = 1$, and initialize the experience trajectory.
  for $t = 1, \ldots, T$ **do**
    Map AGV task states to arrival rates and calculate $D_u(t)$, $D_d(t)$, and $\eta^{\text{ref}}(t)$.
    Obtain channel states, instantaneous load ratios, and the remaining budget; construct s_t.
    Select a_t according to $\pi_\phi(a_t \mid s_t)$.
    if $a_t \neq a_0$ **then**
      Check $K_{\text{used}} < K_{\max}$ and the minimum update interval.
      Check the minimum ratio change and $\rho_{u,n}(t), \rho_{d,n}(t) < 0.90$ for the candidate η_j.
      if all constraints are satisfied **then**
        Set $\eta(t) = \eta_j$ and increment K_{used}.
      else
        Keep $\eta(t) = \eta(t-1)$.
      end if
    else
      Keep $\eta(t) = \eta(t-1)$.
    end if
    Calculate $T_{\text{act}}(t)$, $T_{\text{keep}}(t)$, and reward r_t.
    Obtain s_{t+1} and store the interaction sample.
    if the trajectory length reaches H **then**
      Calculate GAE advantages and target returns; update the actor and critic.
    end if
  end for
end for
Obtain the trained policy π_ϕ^*.
Online stage: dynamic slot-ratio decisions
Fix the actor parameters ϕ.
Initialize the evaluation ratio with the nearest-safe or minimum-load rule; set $K_{\text{used}} = 1$.
for $t = 1, \ldots, T$ **do**
  Evaluate $\rho_{\max}$ at the current ratio and construct $\mathcal{F}(t)$.
  if $\rho_{\max} \geq 0.95$ and $K_{\text{used}} < K_{\max}$ **then**
    if $\mathcal{F}(t) \neq \varnothing$ **then**
      Select η_{rec} nearest to the current ratio in $\mathcal{F}(t)$.
    else
      Select $\eta_{\text{rec}} \in \arg\min_{\eta_j \in \mathcal{A}_\eta} \rho_{\max}(\eta_j, t)$; select the minimum-load fallback.
    end if
    if $|\eta_{\text{rec}} - \eta_{\text{cur}}| \geq 0.015$ **then**
      Execute recovery, increment K_{used}, and set the last-update step to t.
    end if
  end if
  Construct s_t after recovery and obtain the policy action.
  Execute a candidate only if budget remains, the update gap exceeds 10, the change is at least 0.015, and $\rho_{\max} < 0.90$.
  Otherwise retain the current ratio; no change is allowed after budget exhaustion.
  Evaluate delay with the common unstable-queue penalty; record K_{used}.
end for
Output the dynamic slot-ratio sequence and average closed-loop communication delay.

6. Simulation Results

MATLAB R2022b simulations are conducted to evaluate the task-aware delay model and the proposed dynamic TDD optimization methods. Five schemes are compared: offline-optimized fixed slot-ratio allocation (Optimization-Fixed) [31], Periodic slot-ratio updates [32], Event-Triggered slot-ratio updates, PPO-based joint slot-ratio optimization, and an advantage actor–critic (A2C) reinforcement learning baseline, motivated by established asynchronous actor–critic deep reinforcement learning methods [33]. For each operating setting (N, B), Optimization-Fixed searches offline for the constant uplink slot ratio that minimizes the average closed-loop communication delay over the evaluated candidate ratios. This search is repeated for each AGV-count setting at $B = 25$ MHz and each bandwidth setting at $N = 15$, yielding a setting-dependent ratio $\eta^{*}_{\mathrm{fix}}(N, B)$. The selected ratio remains constant throughout the corresponding online evaluation trajectory. This setting-dependent offline-optimized fixed-ratio baseline assesses the benefit of dynamic adaptation relative to an optimized constant allocation. AGV task states are generated by a random-state process to emulate switching between assembly and non-assembly tasks as AGVs move through the workshop. Each generated task state determines the corresponding interaction period and packet sizes, after which the common communication and queueing models are applied to all five schemes. For each parameter setting, multiple Monte Carlo test trajectories are generated, and their averages are used as the final performance metrics. In the main comparison, Periodic and Event-Triggered parameters are selected using a separately generated offline trajectory before the test loop. Each test trajectory is shared across allocation methods to permit paired comparisons, while the PPO agent is trained beforehand or loaded from a saved model and is not updated during testing. The AGV-count and bandwidth sweeps use reproducible random seeds 12,026 and 22,026, respectively; these are evaluation seeds, not independently trained PPO seeds. The comparisons use the same evaluation trajectories for each allocation scheme, providing a consistent basis for reporting the mean delay across the Monte Carlo runs.

6.1. Simulation Parameters

The main system and simulation parameters are listed in Table 2. Closed-loop interaction periods are set to 5 ms for assembly task states and 120 ms for non-assembly task states. The respective uplink packet sizes are 4096 and 512 bits, and the downlink packet sizes are 768 and 384 bits. The average uplink and downlink signal-to-noise ratios (SNRs) are 15 and 18 dB, respectively, with a channel-fluctuation standard deviation of 1.2 dB. Both codeword blocklengths are 512 channel uses, and the target block error probability is set to 10^{-5} per codeword. The simulation horizon is $T = 2000$ decision steps, the maximum number of slot-ratio updates is $K_{\max} = 20$, and the uplink fraction lies in $[0.20, 0.85]$. Each setting in the AGV-count and system-bandwidth sweeps is evaluated using 1000 Monte Carlo runs. The average closed-loop communication delay is evaluated with the equivalent TDD service-rate model, while $K_{\max}$ sets the maximum number of counted resource configurations during the decision horizon.

The task generator uses a two-state macro-load process initialized with equal probabilities and switched with probability 0.005 per decision step. Its dwell length is geometrically distributed with a mean of 200 decision steps. Conditional on the current macro state, each AGV's assembly indicator is independently resampled at every step with probability 0.6 or 0.4; its non-assembly probability is the complement. This generator produces time-varying assembly and non-assembly task demands throughout the decision horizon, with the AGV interaction periods specified independently by T_{asm} and T_{non}. Uplink and downlink channel samples are independently generated across steps and AGVs in the dB domain, with

respective means 15 and 18 dB and standard deviation 1.2 dB; the channel realization is independently sampled at each decision step.

Table 2. System and simulation parameters.

Parameter	Value	Parameter	Value
T_{asm}	5 ms	T_{non}	120 ms
β_{asm}	4096 bits	γ_{asm}	768 bits
β_{non}	512 bits	γ_{non}	384 bits
$\overline{\text{SNR}}_u$	15 dB	$\overline{\text{SNR}}_d$	18 dB
σ_{SNR}	1.2 dB	ε	10^{-5}
S_u	512	S_d	512
T (decision steps)	2000	$K_{\max}$	20
Δ_{dec} (decision interval)	5 ms	Total simulation duration	10 s
$\eta_{\min}$	0.20	$\eta_{\max}$	0.85
N	1–21	B	20–30 MHz

S_u and S_d denote the uplink and downlink codeword blocklengths, respectively, measured in channel uses. T denotes the simulation horizon, measured in resource-allocation decision steps.

When training is invoked, the PPO environment samples N from $\{1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21\}$ at episode reset. Bandwidth is sampled uniformly from 22–30 MHz with probability 0.6 and from 20–30 MHz otherwise, using the same task and channel generators. A single trained or saved PPO agent is reused across the parameter sweeps without retraining at each setting. The training and testing procedures use the stated environment generator, and the trained policy is reused across the reported evaluation settings.

An A2C baseline is additionally evaluated to compare PPO against another actor–critic reinforcement learning method. A2C uses the same task-aware communication environment, 15-dimensional observations, and 28-action allocation space; both A2C and PPO are trained for 700 episodes to maintain the same training-episode budget.

The PPO-specific network architecture and training parameters are listed separately in Table 3. The 15-dimensional state is mapped to a discrete action space containing one keep action and 27 candidate slot ratios spaced by 0.025, resulting in 28 actions. Both the actor and critic use two fully connected hidden layers with 128 ReLU units per layer. The remaining parameters in Table 3 specify the trajectory length, mini-batch optimization, learning rates, discounting, GAE, clipping, entropy regularization, and training horizon used for offline policy learning.

Table 3. PPO architecture and training parameters.

Parameter	Value	Parameter	Value
State dimension	15	Action-space size	28
Candidate-ratio spacing	0.025	Update-gap parameter	10 decision steps
Actor hidden layers	128–128	Critic hidden layers	128–128
Experience horizon H	512	Mini-batch size	128
Epochs per update	5	Training episodes	700
Discount factor γ	0.999	GAE factor λ_{GAE}	0.95
Clipping factor δ_c	0.10	Entropy-loss weight	0.005
Actor learning rate	6×10^{-5}	Critic learning rate	1.1×10^{-4}
Gradient threshold	1	Maximum steps per episode	2000

For comparison, all five schemes are evaluated with the same generated task and channel trajectories, system parameters, per-device bandwidth rule, and communication delay model. Optimization-Fixed uses the offline-selected $\eta^*_{\text{fix}}(N, B)$ for the current setting and performs no online reconfiguration. The selected ratio can differ across AGV counts and bandwidths, while remaining unchanged within each evaluation trajectory. The

Periodic, Event-Triggered, A2C, and PPO schemes are each limited to at most $K_{\max}$ counted configurations, including their initial task-based configuration where applicable. The Periodic scheme updates the resource configuration every 100 decision steps. The Event-Triggered scheme uses the same minimum update interval as PPO and decides whether to reconfigure from the reference-to-current deviation using the best-performing threshold selected offline from the predefined candidate grid for the specified simulation setting and generated random trajectories. PPO jointly determines whether to update and which target ratio to select based on traffic demands, the current ratio, channel states, instantaneous load ratios, and the remaining budget; its execution layer additionally applies the $\rho < 0.90$ limit to candidate policy updates and the preventive recovery rule described above.

6.2. PPO Training Performance

Figure 3 shows the PPO training process. At the beginning of training, episode rewards are low and fluctuate substantially because the policy is still exploring. As training progresses, the average reward increases and stabilizes after approximately 230 episodes. During the later stages, both the average reward and the 50-episode moving mean remain within a stable range. This indicates that the training process converges under the considered simulation environment and parameter settings.

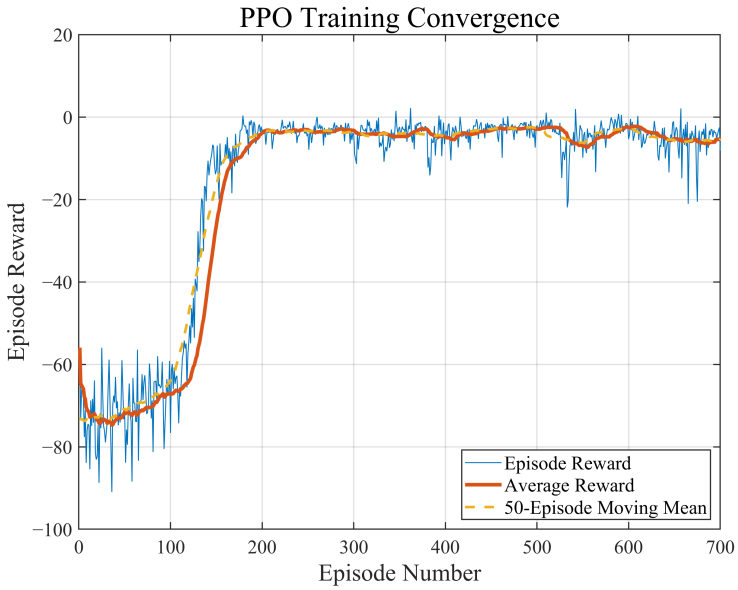


Figure 3. PPO training convergence.

Individual episode rewards continue to vary because the AGV task states and wireless channels are randomly generated. Nevertheless, the smoothed reward curves stabilize in the considered training environment, indicating consistent policy learning across the later training episodes.

6.3. Average Closed-Loop Communication Delay Versus the Number of AGVs

Figure 4 compares five schemes for different numbers of AGVs at a fixed bandwidth of $B = 25$ MHz. The average delay increases with the number of AGVs for all methods as the available bandwidth per device decreases. PPO achieves a lower average delay than A2C at every tested AGV count, as well as the lowest delay among the five compared methods.

At $N = 21$, the average delays of Optimization-Fixed, Periodic, Event-Triggered, PPO, and A2C are 1.304, 1.172, 1.084, 1.04770, and 1.112 ms, respectively. PPO reduces the delay by approximately 19.65%, 10.61%, 3.35%, and 5.78% relative to these four baselines, respec-

tively. The comparison with A2C supports the effectiveness of PPO under the evaluated settings.

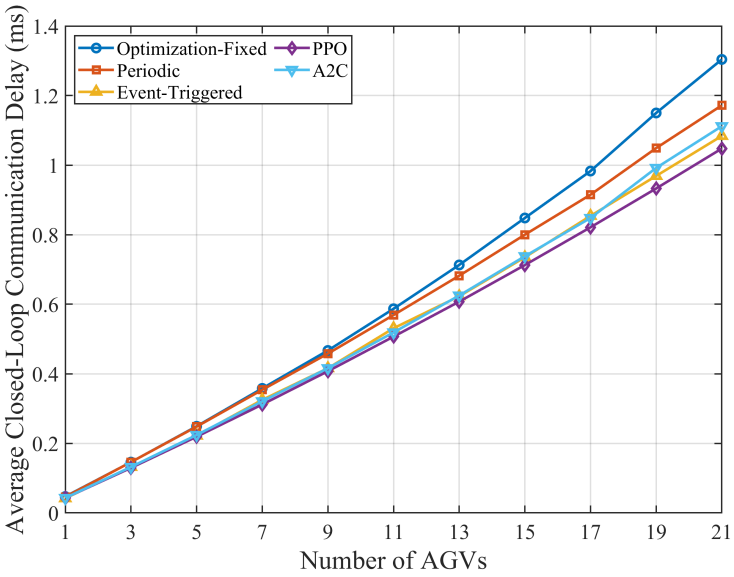


Figure 4. Average closed-loop communication delay versus the number of AGVs, comparing Optimization-Fixed, Periodic, Event-Triggered, PPO, and A2C allocation.

6.4. Average Closed-Loop Communication Delay Versus System Bandwidth

Figure 5 compares the same five methods across bandwidths from 20 to 30 MHz at $N = 15$. Average closed-loop communication delay decreases for all methods as the bandwidth increases. PPO consistently achieves lower delay than A2C across the tested bandwidth range, in addition to outperforming the Optimization-Fixed, Periodic, and Event-Triggered baselines in this sweep.

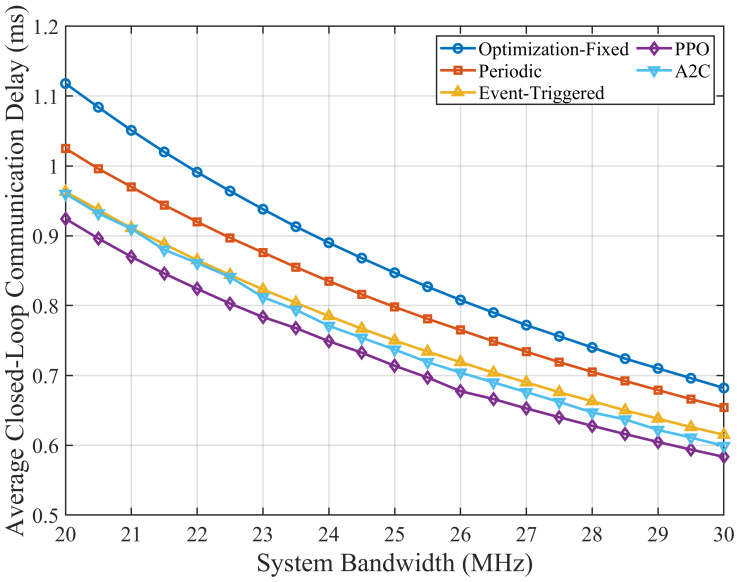


Figure 5. Average closed-loop communication delay versus system bandwidth, comparing Optimization-Fixed, Periodic, Event-Triggered, PPO, and A2C allocation.

At $B = 25$ MHz, PPO achieves an average delay of approximately 0.714 ms, compared with 0.737 ms for A2C, corresponding to a reduction of approximately 3.16%. These comparisons indicate that PPO attains more favorable delay performance than A2C under the evaluated settings, with both methods trained for 700 episodes.

6.5. Impact of the Event-Triggering Threshold

Figure 6 presents the threshold-dependent delay performance for $N = 15$, $B = 25$ MHz, and $K_{\max} = 20$. The average number of counted configurations, including the initialization, is also evaluated. All candidate thresholds are evaluated on the same Monte Carlo trajectories to reduce variability across comparisons, and the best-performing candidate is identified from the resulting delay measurements for the specified operating setting.

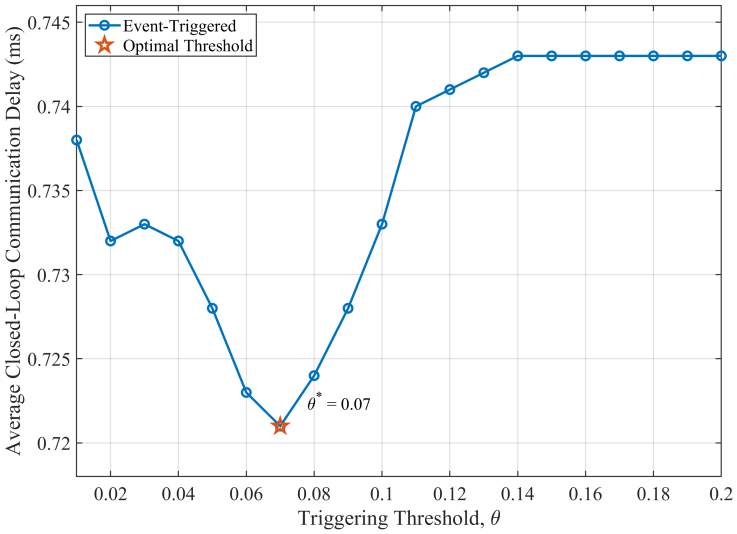


Figure 6. Performance under different event-triggering thresholds. The asterisk (*) indicates the selected optimal triggering threshold.

With a small threshold, the system responds even to slight changes in the reference ratio, causing frequent reconfigurations. As the threshold increases, small traffic fluctuations are filtered out, the counted configuration number decreases substantially, and the average closed-loop communication delay initially decreases.

Within the predefined candidate grid and under the specified setting ($N = 15$, $B = 25$ MHz, and $K_{\max} = 20$) using the generated random trajectories, $\theta = 0.07$ yields the minimum observed average delay of 0.721 ms, with 2.26 counted configurations on average. Compared with $\theta = 0.01$, the average configuration count is reduced by approximately 88.7%, while the average delay is further reduced by approximately 2.30%. Thus, more frequent updates do not necessarily produce a lower closed-loop communication delay in this experiment. The result directly shows that the additional configurations observed at $\theta = 0.01$ do not reduce the reported average delay under this setting.

For candidate thresholds larger than 0.07 in this experiment, stricter triggering conditions are accompanied by an increase in the average delay. At $\theta \geq 0.14$, the average counted configuration number approaches one, meaning that almost no reconfiguration occurs after initialization. The average delay then stabilizes at approximately 0.743 ms. These results identify a scenario-dependent operating point within the evaluated candidate grid.

7. Discussion

The AGV-count and bandwidth experiments show that adapting the uplink/downlink slot ratio reduces average closed-loop communication delay under the evaluated task-dependent traffic conditions. As the number of AGVs increases, shared bandwidth becomes more constrained and the advantages of adaptive allocation become more pronounced. Increasing the system bandwidth reduces delay across all compared methods, while preserving their observed performance ordering.

The event-triggered results highlight the importance of balancing responsiveness against the reconfiguration budget. The threshold selected by offline search, $\theta = 0.07$, achieves the lowest observed delay within the tested candidate grid for $N = 15$, $B = 25$ MHz, and $K_{\max} = 20$. Very frequent updates are not necessarily beneficial, whereas excessively restrictive thresholds approach static allocation. The selected threshold is therefore specific to the evaluated conditions rather than a universal optimum.

PPO further reduces delay by jointly choosing whether to update and which candidate ratio to use based on traffic demand, channel conditions, queue-load indicators, and the remaining budget.

The newly added A2C baseline achieves higher average delay than PPO throughout the AGV-count and bandwidth sweeps. Both A2C and PPO are trained for 700 episodes, and PPO achieves lower average delay under the evaluated settings. Its measured improvement over Event-Triggered allocation is smaller than the improvement over the Optimization-Fixed and Periodic baselines. These comparisons characterize the complete evaluated allocation schemes, including PPO's execution safeguards, rather than isolating the learned policy or individual framework components.

The aggregated 15-dimensional PPO observation provides a compact representation of the system conditions, but it may not fully preserve the heterogeneous traffic and channel information of individual AGVs. Consequently, different underlying system states may produce similar observations, potentially affecting allocation optimality and policy robustness.

The locally stationary M/M/1 approximation provides a tractable characterization of average queueing delay, although its accuracy may be affected when the actual packet arrival and service processes deviate from the model assumptions.

The results are based on averaged TDD service capacity, locally stationary M/M/1 queueing approximations, and simulated task/channel trajectories. Explicit physical TDD scheduling, persistent packet backlogs, and reconfiguration overhead are not included. Future work will investigate packet-level validation, robustness under more realistic traffic and channel dynamics, and deployment-oriented computational evaluation.

8. Conclusions and Future Work

This paper developed a task-aware closed-loop communication delay analysis and dynamic TDD optimization framework for multi-AGV networks in a flexible automotive final-assembly workshop. The proposed model maps assembly and non-assembly task states to interaction periods and uplink/downlink arrival rates and combines finite-blocklength transmission with locally stationary M/M/1 queueing to characterize closed-loop communication delay. Based on this model, an event-triggered method was designed to selectively update the TDD slot ratio under a finite reconfiguration budget, and a PPO-based method was further developed to jointly determine whether to reconfigure and which slot ratio to adopt. At $N = 21$, Optimization-Fixed, Event-Triggered, PPO, and A2C achieve average delays of 1.304, 1.084, 1.04770, and 1.112 ms, respectively. PPO reduces delay by 19.65% relative to Optimization-Fixed and by 5.78% relative to A2C in this scenario. These results show that the two adaptive schemes reduce multi-AGV closed-loop communication delay relative to the evaluated Optimization-Fixed and Periodic baselines under the stated simulation assumptions and reconfiguration limits.

Future work will investigate joint optimization of the TDD slot ratio, per-device bandwidth, transmit power, and traffic priorities, together with more realistic interference-aware channel models and heterogeneous traffic processes. More complex AGV mobility, bursty packet arrivals, multiple concurrent service classes, and experimental or hardware-in-the-loop

validation will also be considered to further evaluate the robustness, generalization capability, and scalability of the proposed framework in practical industrial wireless networks.

Author Contributions: Conceptualization, Y.W., Y.Z. and Z.W.; methodology, Y.W.; writing—original draft preparation, Y.W.; writing—review and editing, H.L., X.C. and Y.F.; supervision, Z.W. and T.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Beijing University of Posts and Telecommunications–China Unicom Joint Innovation Center under Grant 2025-STHZ-BJYDDX-005, and in part by the National Natural Science Foundation of China (NSFC) under Grant 62271081.

Data Availability Statement: The simulation data generated and analyzed during the current study are available from the corresponding author upon reasonable request.

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

References

1. Meng, Z.; Ma, D.; Wang, S.; Wei, Z.; Feng, Z. Modeling and design of the communication sensing and control coupled closed-loop industrial system. In *Proceedings of the GLOBECOM 2023—2023 IEEE Global Communications Conference*; IEEE: New York, NY, USA, 2023; pp. 2626–2631.
2. Liu, H.; Wei, Z.; Li, F.; Lin, Y.; Qu, H.; Wu, H.; Feng, Z. Integrated sensing and communication signal processing based on compressed sensing over unlicensed spectrum bands. *IEEE Trans. Cogn. Commun. Netw.* **2024**, *10*, 1801–1816. [CrossRef]
3. Liu, H.; Wei, Z.; Du, Y.; Lu, B.; Yang, F.; Li, X.; Feng, Z. WCTNet-ISAC: A two-stage scheme with CSI enhancement and high-precision synchronization for uplink ISAC signal processing. *IEEE Trans. Mob. Comput.* **2026**, *25*, 17634–17652. [CrossRef]
4. Shi, W.; Xu, W.; You, X.; Zhao, C.; Wei, K. Intelligent reflection enabling technologies for integrated and green Internet-of-Everything beyond 5G: Communication, sensing, and security. *IEEE Wirel. Commun.* **2022**, *30*, 147–154. [CrossRef]
5. Qiao, Y.; Fu, Y.; Yuan, M. Communication–control co-design in wireless networks: A cloud control AGV example. *IEEE Internet Things J.* **2022**, *10*, 2346–2359. [CrossRef]
6. Gong, Y.; Yao, H.; Wang, J.; Li, M.; Guo, S. Edge intelligence-driven joint offloading and resource allocation for future 6G industrial Internet of Things. *IEEE Trans. Netw. Sci. Eng.* **2022**, *11*, 5644–5655. [CrossRef]
7. Xu, W.; Wei, Z.; Liu, L.; Zhang, Y.; Liu, H.; Zhou, Y.; Pang, Y.; Feng, Z. Delay performance analysis with short packets in intelligent machine networks. In *Proceedings of the 2025 IEEE 102nd Vehicular Technology Conference (VTC2025-Fall)*; IEEE: New York, NY, USA, 2025; pp. 1–5.
8. Accurso, N.; Mastronarde, N.; Malandra, F. A comprehensive analytical model of end-to-end uplink delay in 5G NB-IoT networks. In *Proceedings of the 2025 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*; IEEE: New York, NY, USA, 2025; pp. 1–6.
9. Li, C.P.; Jiang, J.; Chen, W.; Ji, T.; Smee, J. 5G ultra-reliable and low-latency systems design. In *Proceedings of the 2017 European Conference on Networks and Communications (EuCNC)*; IEEE: New York, NY, USA, 2017; pp. 1–5.
10. Chen, J.; Wei, Z.; Liu, L.; Meng, Z.; Pang, Y.; Feng, Z. Analysis of closed-loop delay for multi-device industrial system. In *Proceedings of the 2025 5th International Conference on Intelligent Communications and Computing (ICICC)*; IEEE: New York, NY, USA, 2025; pp. 241–247.
11. Esswie, A.A.; Pedersen, K.I. On the ultra-reliable and low-latency communications in flexible TDD/FDD 5G networks. In *Proceedings of the 2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*; IEEE: New York, NY, USA, 2020; pp. 1–6.
12. Adamuz-Hinojosa, O.; Delgado-Ferro, F.; Domènech, N.; Navarro-Ortiz, J.; Muñoz, P.; Mahdi Darroudi, S. Empirical analysis of 5G TDD patterns configurations for industrial automation traffic. In *Proceedings of the 2025 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*; IEEE: New York, NY, USA, 2025; pp. 488–493.
13. Raffeck, S.; Grøsvik, S.G.; Lange, S.; Hossfeld, T.; Zinner, T.; Geissler, S. Parameterizing 5G new radio: A comparative measurement study on throughput and delay. In *Proceedings of the 2024 20th International Conference on Network and Service Management (CNSM)*; IEEE: New York, NY, USA, 2024; pp. 1–7.
14. Shi, W.; Xu, W.; Huang, Y.; Yao, J.; Hu, W.; Wang, D. Flexible-duplex cell-free architecture for secure uplink communications in low-altitude wireless networks. *IEEE J. Sel. Top. Signal Process.* **2026**, 1–15. [CrossRef]
15. Kerttula, J.; Marttinen, A.; Ruttik, K.; Jäntti, R.; Alam, M.N. Dynamic TDD in LTE small cells. *EURASIP J. Wirel. Commun. Netw.* **2016**, *2016*, 194. [CrossRef]
16. Alwarafy, A.; Abdallah, M.; Ciftler, B.S.; Al-Fuqaha, A.; Hamdi, M. Deep reinforcement learning for radio resource allocation and management in next generation heterogeneous wireless networks: A survey. *arXiv* **2021**, arXiv:2106.00574.

17. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347.
18. Kay, S.M. *Fundamentals of Statistical Signal Processing: Estimation Theory*; Prentice-Hall, Inc.: Hoboken, NJ, USA, 1993.
19. Huang, K.; Liu, W.; Li, Y.; Vucetic, B.; Savkin, A.V. Optimal downlink–uplink scheduling of wireless networked control for industrial IoT. *IEEE Internet Things J.* **2019**, *7*, 1756–1772. [CrossRef]
20. Park, P.; Ergen, S.C.; Fischione, C.; Lu, C.; Johansson, K.H. Wireless network design for control systems: A survey. *IEEE Commun. Surv. Tutor.* **2017**, *20*, 978–1013. [CrossRef]
21. Shi, H.; Zheng, M.; Liang, W.; Zhang, J.; Wang, K.; Liu, S. Transmission scheduling with order constraints in WIA-FA-based AGV systems. *IEEE Internet Things J.* **2020**, *8*, 381–392. [CrossRef]
22. Nasir, A.A.; Tuan, H.D.; Nguyen, H.H.; Debbah, M.; Poor, H.V. Resource allocation and beamforming design in the short blocklength regime for URLLC. *IEEE Trans. Wirel. Commun.* **2020**, *20*, 1321–1335. [CrossRef]
23. Durisi, G.; Koch, T.; Popovski, P. Toward massive, ultrareliable, and low-latency wireless communication with short packets. *Proc. IEEE* **2016**, *104*, 1711–1726. [CrossRef]
24. Polyanskiy, Y.; Poor, H.V.; Verdú, S. Channel coding rate in the finite blocklength regime. *IEEE Trans. Inf. Theory* **2010**, *56*, 2307–2359. [CrossRef]
25. Bertsekas, D.P.; Gallager, R.G.; Humblet, P. *Data Networks*; Prentice Hall: Englewood Cliffs, NJ, USA, 1992.
26. Kobayashi, H.; Konheim, A. Queueing models for computer communications system analysis. *IEEE Trans. Commun.* **1977**, *25*, 2–29. [CrossRef]
27. Tang, F.; Zhou, Y.; Kato, N. Deep reinforcement learning for dynamic uplink/downlink resource allocation in high mobility 5G HetNet. *IEEE J. Sel. Areas Commun.* **2020**, *38*, 2773–2782. [CrossRef]
28. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, UK, 1998.
29. Ciosek, K.; Whiteson, S. Expected policy gradients for reinforcement learning. *J. Mach. Learn. Res.* **2020**, *21*, 1–51. [CrossRef]
30. Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.I.; Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *arXiv* **2015**, arXiv:1506.02438.
31. NGMN Alliance. *5G TDD Uplink*; White Paper; NGMN Alliance: Düsseldorf, Germany, 2022.
32. Esswie, A.A.; Pedersen, K.I.; Mogensen, P.E. Quasi-dynamic frame coordination for ultra-reliability and low-latency in 5G TDD systems. In *Proceedings of the 2019 IEEE International Conference on Communications Workshops (ICC Workshops)*; IEEE: New York, NY, USA, 2019; pp. 1–6.
33. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. *Proc. Mach. Learn. Res.* **2016**, *48*, 1928–1937.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.