Available online at https://jcupt.bupt.edu.cn

The Journal of China Universities of Posts and Telecommunications

April 2026, 33(2): 86 – 101

CSQF-based hybrid traffic scheduling method for wide area networks

Liu Xiaokai[1], Ding Tianju[1], Xu Fangmin[2](✉), Zong Chuyu[2], Zhao Chenglin[2]

1. School of Mechanical and Electrical Engineering, Beijing Information Science and Technology University, Beijing 102206, China
2. School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China

Abstract

This study addresses hybrid traffic scheduling challenges in time-sensitive networking (TSN) over wide area networks (WANs) by proposing a dynamic collaborative scheduling algorithm based on cyclic specified queuing and forwarding (CSQF). Fixed resource blocks (FBs) and elastic resource blocks (EBs) are partitioned to isolate time-triggered (TT) and audio-video bridging (AVB) flows. Dynamic cycle scaling and sliding window preemption further optimize resource efficiency. Simulations show the algorithm achieves 75.6% AVB flow scheduling success under 3 000 total flows, reduces resource fragmentation by 28% compared to static strategies, and ensures 100% TT flow success. Results demonstrate its effectiveness in balancing strict TT delay constraints with enhanced hybrid traffic performance, offering a practical solution for deterministic WAN transmission.

Keywords time-sensitive networking (TSN), hybrid traffic scheduling, cyclic specified queuing and forwarding (CSQF), dynamic collaboration

1 Introduction

In emerging domains such as industrial Internet, smart manufacturing, and autonomous driving, TSN has emerged as a pivotal technology underpinning real-time communication, leveraging its high-precision latency control and large-scale data transmission capabilities[1]. However, the deployment of TSN within WANs encounters significant challenges. Traditional scheduling mechanisms (e.g., time-aware shaper (TAS)[2] and cyclic queuing and forwarding (CQF)[3]) inherently rely on the assumption of negligible propagation latency, typical of local area networks (LANs). Furthermore, these mechanisms are constrained by the seven-hop limitation imposed by the IEEE 802.1D protocol[4], rendering them ill-suited for the inherent latency characteristics of long-distance networks. Additionally, issues such as insufficient time synchronization precision, high traffic dynamism, and intense resource contention within WANs further impede the large-scale adoption of TSN.

To extend TSN to wide-area environments, the Internet engineering task force (IETF) deterministic networking (DetNet) Working Group proposed the CSQF mechanism. It achieves deterministic end-to-end latency control (jitter less than or equal to 2 cycles) by partitioning time into fixed-length cycles and annotating packet headers with cycle labels[5]. Nevertheless, CSQF exhibits the following limitations. 1) Its cyclic label stack primarily supports single-flow linear path scheduling, presenting difficulties in multi-flow hybrid traffic scenarios; 2) the scheduling problem exhibits nondeterministic polynomial-time hard

Received date: 18-08-2025
Corresponding author: Xu Fangmin, E-mail: xufm@bupt.edu.cn
DOI: 10.19682/j.cnki.1005-8885.2026.1007

(NP-hard) characteristics, resulting in high modeling complexity; 3) The diverse quality of service (QoS) requirements inherent to hybrid traffic exacerbate resource allocation conflicts. For instance, TT flows demand strict periodicity and low-latency guarantees, whereas AVB flows place greater emphasis on bandwidth utilization. Traditional static scheduling algorithms struggle to concurrently satisfy these divergent requirements.

Hybrid traffic encompasses three distinct flow types: TT flows, audio AVB flows, and best-effort (BE) flows. TT flows mandate stringent periodicity and timeliness for transmission, while AVB flows possess specific requirements for bandwidth and transmission latency[6]. The existing scheduling algorithms (such as greedy randomized adaptive search procedure (GRASP) and particle swarm-optimization orthogonal multiple access (PSO-OMA)) exhibit computational complexity that grows super-linearly with network scale ($O(n^3)$), making them unable to meet the real-time scheduling demands for large-scale nodes and traffic in WANs. For instance, as mentioned in Ref. [7], dynamic topology changes and spatio-temporal constraints of cross-domain traffic in scenarios like the fifth generation mobile network (5G) core networks or cloud-edge collaboration further exacerbate scheduling difficulties. Ref. [8] proposed an integer linear programming (ILP) formulation for the scheduling problem and introduced a GRASP-based heuristic for routing. Ref. [9] presented a hybrid traffic scheduling mechanism combining TAS and CQF mechanisms; this mechanism dynamically arranges the overall scheduling sequence of mixed traffic based on flow laxity while ensuring the schedulability of TT flows, thereby guaranteeing efficient scheduling of multiple types of mixed traffic in TSN. Ref. [10] proposed dependency-aware priority scheduling (DAPS) adjustment algorithms tailored for different link overlap conditions. Refs. [11-13] designed a probabilistic coloring algorithm, an optimized hybrid deterministic scheduling and routing (OHDSR) method, and an improved PSO-OMA, respectively, to reduce latency and conflicts. However, these methods predominantly focus on LAN scenarios and suffer from high algorithmic complexity, making them challenging to adapt to the scalability demands of WANs.

Consequently, to address the challenges of hybrid traffic scheduling in WAN environments, this paper proposes a dynamic cycle-cooperative scheduling algorithm based on the CSQF mechanism. This algorithm rationally partitions periodic resource blocks to jointly accommodate the scheduling requirements of both TT and AVB flows, ensuring the timeliness constraints of TT flows are met while concurrently providing flexible bandwidth allocation strategies for AVB flows. Furthermore, the algorithm incorporates a dynamic cycle scaling mechanism to adapt to fluctuating network loads. Additionally, it employs a sliding window preemption strategy to mitigate conflicts among different traffic types, thereby optimizing the overall scheduling efficacy of the hybrid traffic.

The main contributions of this paper can be summarized in three points.

1) A separation mechanism for fixed resource blocks (FB) and elastic resource blocks (EB) based on CSQF is proposed, achieving physical isolation between TT flows and AVB flows; compared with traditional CSQF's single-process scheduling, this effectively resolves QoS conflicts arising from mixed traffic.

2) A dynamic period-scaling and sliding-window pre-allocation coordination strategy is designed; relative to static-period scheduling methods, it reduces resource fragmentation by 28% and improves the scheduling success rate of AVB flows by more than 15%.

3) A GPU-accelerated parallel ILP solving framework is constructed for large-scale WAN scenarios; compared with existing ILP-based scheduling algorithms, it achieves the 3 - 5 times increase in computational efficiency and markedly enhances scalability. Compared with representative approaches, the proposed method overcomes the limitations of conventional CSQF in mixed-traffic scheduling, balances the determinism of TT flows with the elasticity of AVB flows, and is better suited to WAN environments characterized by long latency and massive deployment.

2 System model

2.1 Network model

A TSN is abstracted as a directed graph $G=(V,E)$, where V denotes the set of network nodes comprising TSN switches and end stations. The set E represents physical links. Since TSN operates over full-duplex Ethernet, a physical link between nodes V_a and V_b corresponds to two unidirectional links: $[V_a,V_b]$ and $[V_b,V_a]$[14].

Network traffic is classified into three categories: TT flows, AVB flows, and best-effort (BE) flows. This paper exclusively considers the scheduling of TT and AVB flows, excluding BE flows from consideration. Thus, the schedulable flow set F is defined as $F=\{F_{\mathrm{TT}},F_{\mathrm{AVB}}\}$.

If a flow $f_i\in F_{\mathrm{TT}}$, it is characterized by the seven-tuple,

$$f_i=\{S_i^{\mathrm{T}},D_i^{\mathrm{T}},T_{i_\mathrm{cycle}}^{\mathrm{T}},L_i^{\mathrm{T}},D_i^{\mathrm{T}},P_i^{\mathrm{T}},T_{i_\mathrm{offset}}^{\mathrm{T}}\} \tag{1}$$

where S_i^{T} denotes the source node of the TT flows, D_i^{T} denotes the destination node, $T_{i_\mathrm{cycle}}^{\mathrm{T}}$ denotes the transmission period, L_i^{T} denotes the frame size (in B), D_i^{T} denotes the deadline of the flow, P_i^{T} denotes the path information of the flows, $T_{i_\mathrm{offset}}^{\mathrm{T}}$ denotes the offset time of the flows.

If a flow $f_i\in F_{\mathrm{AVB}}$, it is characterized by the five-tuple,

$$f_j=\{S_j^{\mathrm{A}},D_j^{\mathrm{A}},B_j^{\mathrm{A}},D_j^{\mathrm{A}},P_j^{\mathrm{A}}\} \tag{2}$$

where S_j^{A} denotes the source node of the AVB flows, D_j^{A} denotes the destination node, B_j^{A} denotes the bandwidth requirement, D_j^{A} denotes the deadline of the flow, P_j^{A} denotes the path information of the flows.

2.2 Hybrid traffic scheduling model

The CSQF mechanism divides the transmission time at an output port into a series of equal intervals, each called a cycle T_{slot}. Among the multiple queues participating in scheduling, only one queue serves as the transmission queue (Tx) per cycle, while the others function as reception queues (Rx). Reception queues exhibit differential waiting times based on their positional offset from the queue (Tx)[15]. The queue resource blocks at each port across different cycles can be defined as $R_{(V_a,V_b)}^{T(t)}$, where $T(t)$ denotes the current cycle within the sequence of time slices, and (V_a,V_b) represents the network link between nodes V_a and V_b.

A hybrid traffic scheduling model designed using the CSQF mechanism must accommodate the QoS requirements of diverse traffic types[16]. This necessitates partitioning resources into two categories: FB and EB, allocated to distinct traffic classes[17]. Specifically, FB guarantee deterministic transmission for TT flows. EB are allocated to bandwidth-sensitive traffic, namely AVB flows. As illustrated in Fig. 1, assume the CSQF scheduling model utilizes N queue resources. The first K queues are allocated to TT flows, while the remaining $N-K$ queues are assigned to AVB flows. Upon arrival, TT flows are prioritized to enter the reception queues closer to the Tx queue. This minimizes their waiting time (requiring fewer cycles), aligning with their stringent low-latency requirements. Conversely, AVB flows, possessing lower priority and more relaxed latency constraints, are directed to rearward reception queues. Each queue has a finite capacity L. When a queue accumulates sufficient packets and exhausts its capacity, it becomes unable to receive further data. In this scenario, the system must select a rearward queue with available capacity for incoming traffic. Failure to locate such a queue result in a buffer overflow, ultimately causing packet loss.

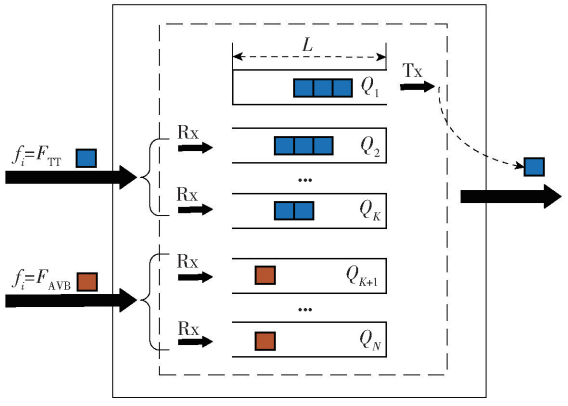

Fig. 1 Hybrid traffic scheduling model based on CSQF

As illustrated in Fig. 2, an example demonstrates the scheduling scenario for TT flows and AVB flows within a two-hop network when $N=3$. Dashed lines in the figure denote waiting for transmission, while solid

lines indicate the ongoing transmission of current traffic. Assuming both flows (one TT flow and one AVB flow) traverse from Node A through Node B to Node C, the TT flow, possessing higher priority, is transmitted preferentially during the T_0 cycle, while the AVB flow waits in the subsequent queue.

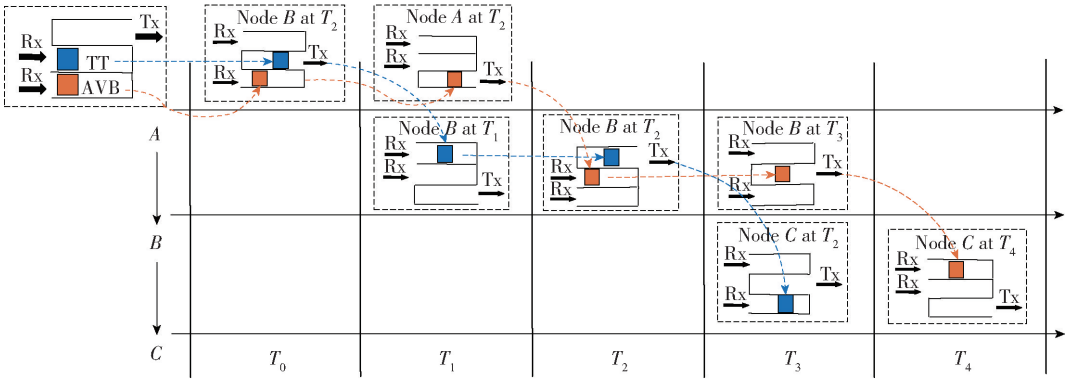

Fig. 2 Illustration of hybrid traffic scheduling with CSQF

According to the operational principle of the CSQF mechanism, the queue states cycle periodically, with each queue taking turns to have its gating mechanism opened to become the transmission queue. Furthermore, given the WAN scenario where propagation delay is non-negligible, it is accounted for that traffic incurs a propagation delay equivalent to one cycle duration before arriving at the next node. Consequently, at T_1, the queue containing the AVB flow at Node A becomes the transmission queue and commences transmission; simultaneously, at Node B, the TT flow enters the foremost receiving queue (which is scheduled to become the transmission queue in the next cycle). During T_2, the AVB flow arrives at Node B and enters a receiving queue; if no TT flow is currently pending scheduling at Node B, it may enter the foremost receiving queue for priority handling. Meanwhile, the TT flow departs from Node B. The scheduling process during T_3 mirrors that of T_1, at which point the TT flow arrives at Node C and completes its scheduling. At T_4, the AVB flow also arrives at Node C and completes its scheduling.

From this sequence, the cycle tag for the TT flow is derived as $\langle 0, 2 \rangle$. In this specific instance, both types of traffic enter their respective queues simultaneously at initialization. Due to the AVB flow occupying a lower-priority queue, it ultimately experiences a scheduling delay greater than that of the TT flow by Δt, equivalent to 1 T_{slot} duration. Under this CSQF hybrid traffic scheduling mechanism, independent queues for each traffic type are guaranteed, and resource allocation is dynamically adjusted based on traffic demands and network load conditions to maximize network resource utilization efficiency.

2.3 Model assumptions

The scheduling model in this paper is based on the following assumptions.

1) BE traffic scheduling is not considered for now: BE traffic has no strict QoS requirements and can be transmitted opportunistically using residual bandwidth in actual deployments. This paper focuses on the hybrid scheduling optimization of TT and AVB flows. Support for priority scheduling mechanisms for BE traffic can be extended in the future.

2) The network has sub-microsecond clock synchronization: this relies on the IEEE 1588 PTPv2 precise clock synchronization protocol. In WAN scenarios, sub-microsecond synchronization accuracy has been achieved through optical fiber transmission and clock correction techniques.

3) A centralized control architecture is adopted: global resource scheduling is implemented based on a centralized software-defined networking (SDN) controller, which aligns with the WAN deterministic

transmission control architecture recommended by the DetNet working group.

A reasonable deployment scenario for the above assumptions should meet the following characteristics. First, it requires high real-time and deterministic transmission performance, with priority given to guaranteeing the QoS characteristics of TT/AVB flows; second, it should feature controllable network infrastructure that can deploy the PTPv2 synchronization protocol and an SDN controller; additionally, it should have a low proportion of BE traffic or accept opportunistic transmission delays. Specific deployment scenarios include: industrial Internet backbone networks, wide-area measurement systems in smart grids, and cloud-edge collaborative manufacturing networks.

3 Algorithm design

3.1 Problem modeling

In WAN environments, the scheduling demands for mixed traffic primarily manifest in balancing the transmission efficiency and resource allocation between TT flows and AVB flows, which exhibit NP-hard characteristics[18]. TT flows typically require stringent delay control, while AVB flows prioritize bandwidth utilization[19]. Driven by these divergent requirements, conventional scheduling methods struggle to simultaneously satisfy the heterogeneous characteristics of both traffic types, particularly under resource-constrained conditions. To address the needs of both traffic types and enable efficient mixed-traffic transmission, this chapter proposes a scheduling scheme based on the CSQF mechanism. This scheme ensures high-efficiency cooperative transmission through precise resource allocation and dynamic scheduling strategies, as depicted in Fig. 3. Specifically, the CSQF mechanism allocates independent queue resources for each traffic type, via dynamic adjustments, guarantees the timeliness of TT flows while maximizing the bandwidth utilization of AVB flows.

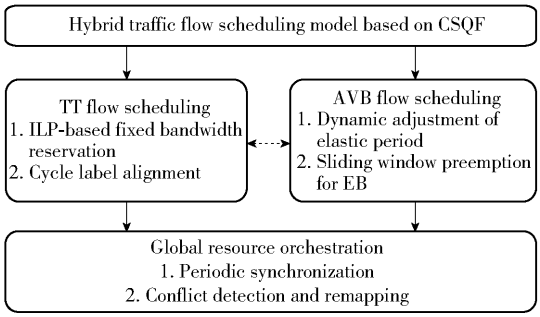

Fig. 3 Collaborative transmission process of TT and AVB flows

To achieve this objective, the following optimization goals are defined.

1) Ensure all TT flows complete end-to-end transmission within their deadlines. That is, TT flows must transmit all packets on time within the specified cycle without exceeding the predefined delay constraints.

2) Maximize the number of successfully scheduled AVB flows, scheduling as many AVB flows as possible to meet their bandwidth requirements. AVB flows are generally more delay-tolerant; therefore, without compromising TT flow transmission, AVB flow scheduling should fully leverage the remaining bandwidth resources. Define S_n as an indicator for whether the AVB flows are successfully scheduled,

$$S_n = \begin{cases} 1; & \text{if } f_j \in F_{\text{AVB}} \text{ schedulable} \\ 0; & \text{Otherwise} \end{cases} \tag{3}$$

Further, the optimization objective is formalized as

$$\max \sum S_n \tag{4}$$

To ensure that the scheduling process satisfies dual requirements of real-time constraints and bandwidth utilization, it is necessary to fulfill a series of constraint conditions. The following outlines primary constraints for different traffic types.

3.1.1 Constraints for TT flows

1) Strict deadline constraint

All packets of every TT flow must complete transmission within their specified delay limits. For a TT flow f_i, its end-to-end delay from the source to the destination node, denoted as f_{i_delay}, must be strictly less than or equal to its deadline D_i^{T}. Thus, all TT

flows must satisfy the following delay constraint,

$$f_{i_\text{delay}} \leqslant D_i^{\text{T}}; \quad \forall f_i \in F_{\text{TT}} \tag{5}$$

where f_{i_delay} represents the end-to-end delay of the flow, and D_i^{T} denotes the maximum allowable delay, i. e. , the deadline.

2) Period lower bound constraint

Each TT flow's packets must complete transmission within its predetermined period to meet latency requirements. The lower bound must guarantee that all packets stored in the queue can be scheduled within this duration. This duration includes the processing delay f_{i_proc}, queuing delay f_{i_queue}, and transmission delay f_{i_trans}. The lower bound constraint is expressed as

$$\min T_{\text{slot}} = \lceil f_{i_\text{proc}} + (f_{i_\text{queue}} + f_{i_\text{trans}}) \rceil \leqslant \left\lceil f_{i_\text{prop}} + \frac{LM}{BW} \right\rceil \tag{6}$$

where L denotes the maximum queue capacity, M denotes the maximum packet size, and BW denotes the link capacity.

3) Period upper bound constraint

To ensure system stability and enable effective scheduling of all TT flows, the upper bound of the period length must guarantee that the periods of all TT flows in the system are divisible by this bound. This upper bound equals the greatest common divisor (GCD) of all flow transmission periods. The constraint is formalized as

$$\max T_{\text{slot}} = G(T_{1_\text{cycle}}^{\text{T}}, T_{2_\text{cycle}}^{\text{T}}, \cdots, T_{i_\text{cycle}}^{\text{T}}) \tag{7}$$

where $T_{i_\text{cycle}}^{\text{T}}$ denotes the period of a TT flow. Typically, T_{slot} exceeds its minimum value and must be divisible by the hypercycle:

$$T_{\text{Hypercycle}} = L(T_{1_\text{cycle}}^{\text{T}}, T_{2_\text{cycle}}^{\text{T}}, \cdots, T_{i_\text{cycle}}^{\text{T}}) \tag{8}$$

$$T_{\text{Hypercycle}} \bmod T_{\text{slot}} = 0 \tag{9}$$

4) Offset time constraint

The bias determines the initial transmission cycle time for each stream, defined as the time when the stream begins transmission from the ingress switch (first switch) at the source node to the terminal node[20]. This bias must not exceed the stream's cycle duration. Furthermore, TT flow must be fully forwarded within the current transmission cycle to prevent flow interference between adjacent cycles,

$$0 \leqslant T_{i_\text{offset}}^{\text{T}} < \frac{L_i^{\text{T}}}{T_{\text{slot}}} \tag{10}$$

5) Bounded queue length constraint

The number of TT flow data packets cached in a single queue must not exceed the total queue capacity L, thereby preventing overflow-induced packet loss. As TT flows utilize FBs, their queue capacity requires independent management. This constraint ensures that each FB queue remains non-overloaded during any cycle

$$Q_{\text{TT}}(t,m) \leqslant L; \quad \forall t, m < K \tag{11}$$

3.1.2 AVB flow constraints

1) Bandwidth constraint

During transmission, the minimum bandwidth B_j^{A} for the AVB flow must be guaranteed. This requires the allocation of continuous or discrete periodic blocks within elastic resource blocks. The quantity of these blocks is determined by converting the bandwidth requirements,

$$N = \left\lceil \frac{B_j^{\text{A}} T_{\text{slot}}}{8} \right\rceil \tag{12}$$

2) Elastic resource block constraints

To prioritize the transmission of TT flows, the dynamic resource acquisition for AVB flows must not occupy the fixed resource blocks allocated to TT flows, thereby avoiding interference with the deterministic transmission of time-sensitive flows. Define the resource allocation matrix $\boldsymbol{R}(t)$, where $r(t,m) = 1$ indicates that queue m is occupied at cycle, and $r(t, m) = 0$ otherwise. Let M_{TT} denote the fixed resource block set dedicated to TT flows. The constraint is formalized:

$$r(t,m) = 0; \quad \forall t, m \in M_{\text{TT}} \tag{13}$$

This constraint ensures strict isolation between EB and FB.

3) Preemption window constraints

AVB flows are permitted to preempt elastic resources only within a finite future horizon via a sliding window strategy, preventing long-term occupation that could affect future reservations for TT flows. Let W denote the preemption window size (in units of cycles). If the current cycle is t, AVB flows may preempt EB resource

blocks solely within the interval $[t, t+W]$. Define $N_{\mathrm{EB}}^{j}(t)$ as the number of EB resource blocks allocated to the flow at a cycle t. The constraint is given as

$$N_{\mathrm{EB}}^{j}(t')=0; \quad \forall t'>t+W, \forall fj \in F_{\mathrm{AVB}} \tag{14}$$

4) Delay constraint

The maximum latency for AVB flows from queue entry to scheduling must satisfy an upper bound to prevent excessive queuing of low-priority traffic. Let D_j^{A} denote the maximum allowable delay for an AVB flow. This constraint is formalized as

$$f_{j_\mathrm{delay}} \leqslant D_j^{\mathrm{A}}; \quad \forall f_j \in F_{\mathrm{AVB}} \tag{15}$$

Combined with queue mapping rules, AVB flows are inserted into lower-priority queues, resulting in significant queuing delay expressed as $N_{\mathrm{nwc}} T_{\mathrm{slot}}$.

3.2 TT flow scheduling based on fixed resource blocks

The ILP-based TT flow scheduling algorithm is designed to achieve high-precision end-to-end transmission for TT flows in WANs, while balancing low computational complexity and high reliability. This algorithm integrates shortest weighted path routing with a periodic scheduling mechanism. Through link delay compensation, periodic label assignment, and queue capacity verification, it resolves latency fluctuations and resource contention issues during TT flow transmission.

To explicitly formalize the scheduling problem, the proposed ILP model is defined by the following key elements. The primary decision variable is denoted as $x_{i,n,c}$, a binary variable that equals 1 if the packet of TT flow f_i is transmitted by node n in cycle c, and 0 otherwise. The objective function aims to minimize the total scheduling time slots used, thereby compacting the schedule to reserve maximum contiguous space for AVB flows. The model is subject to three critical sets of constraints: uniqueness and consistency constraints, which ensure that each flow is scheduled exactly once per period on its path and strictly follows the cycle sequence defined by link delays (Eq. (10)); capacity and conflict-free constraints, ensuring that the number of packets assigned to a specific queue q does not exceed its physical buffer capacity C_q (Eq. (11)) and that fixed resource blocks are not oversubscribed; and hard real-time constraints, which enforce that the cumulative end-to-end latency, including processing and propagation delays, strictly adheres to the flow's deadline D_i (Eq. (5)).

The detailed algorithmic framework of the ILP-TT flow scheduling scheme based on shortest weighted path routing is presented in pseudocode as Algorithm 1.

Algorithm 1 ILP-TT flow scheduling scheme based on shortest weighted path routing

Input: Network topology $G=(V,E)$, link weight, flow set F_{TT} to be scheduled and flow parameters.

Output: Path selection and time slot allocation for $f_i \in F_{\mathrm{TT}}$, i.e., successful scheduling of the TT flow set $F_{\mathrm{suc}}^{\mathrm{T}}$.

For each TT flow $f_i \in F_{\mathrm{TT}}$, from source node S_i^{T} to destination node D_i^{T} do
  Use the shortest weighted path algorithm to select the optimal transmission P_i^{T} considering link delay and bandwidth
  For each node V_i in path P_i^{T} **do**
    Calculate the link propagation delay τ between current node V_a and next node V_b
    Compute the required period offset ΔT_i based on delay τ, and strictly verify if the current queue capacity C_q is sufficient to accommodate the frame size of f_i.
    if verification fails **then**
      Mark as "No feasible route, scheduling failed" and **break**
    end if
  end for
  Construct the ILP model by applying constraints for strict deadline adherence, periodic flow isolation, and maximum queue buffer limits, referencing Eqs. (5) – (11).
  Execute the ILP solver to determine the precise time slot allocation and queue mapping for flow f_i.
  if a feasible solution S_i is found **then**
    Update the set of successfully scheduled flows $S_{\mathrm{TT}} \leftarrow S_{\mathrm{TT}} \cup \{f_i\}$
    Update the occupied link time slot resource blocks in the global reservation matrix
  else

16 "No available slots, scheduling failed"
17 **end if**
18 **end for**
19 **return** S_{TT}

The algorithm initially selects the optimal path for each TT stream using the shortest path algorithm with weighted links. This path selection comprehensively considers both the physical topology of the network and the weights of the links (e.g., link delay), thereby ensuring that the chosen path provides low latency and high bandwidth utilization during transmission.

Subsequently, slot allocation is performed through an ILP formulation[21]. Periodic label assignment is executed for each node V along the current path. The link propagation delay $L_{(V_a,V_{a+1})}$ between the current node V_a and the next node V_{a+1} is computed, followed by the calculation of the period offset $\Delta = \left\lceil \frac{L(Va,V_{a+1})}{T_{slot}} \right\rceil$ based on this delay.

If the current node V_a is the source node, an initial periodic label $\langle T_i, Q_i \rangle$ is assigned, where T_i denotes the period identifier and Q_i represents the queue identifier. For non-source nodes, the periodic label is inherited from the previous hop as $T_a = T_a - 1 + \Delta a - 1$, and the queue position $Q_a = (Q_a - 1 + 1) \bmod M$ is allocated based on the queue identifier of the preceding node.

Upon completing the periodic label assignment, the algorithm verifies whether the queue capacity $C_{TT}(Q_a)$ satisfies the requirements of the TT flow. Should the queue capacity prove insufficient to accommodate the data volume $Q_{TT}(T_a, Q_a) + L(V_a, V_{a+1}) > C_{TT}(Q_a)$ of the current stream, the period offset and periodic label are recalculated to ensure scheduling validity.

Finally, the algorithm updates the resource reservation matrix R_{TT} with the computed periodic labels and queue identifiers. This stores the information of flow f_i in the corresponding queue position, guaranteeing scheduled transmission in future cycles. Successfully scheduled flows are recorded in the set of successfully scheduled TT flows F_{suc}^{T}.

The runtime of the ILP solver for TT flow planning in this algorithm depends on the number of network nodes, the number of TT flows, and the complexity of the constraints. In the experimental environment (Intel (R) Core(TM) i5-8300H CPU + NVIDIA RTX-2080-Ti GPU), when there are 200 TT flows and 16 network nodes, the average runtime of a single ILP solve is 0.87 s; when the number of TT flows increases to 500 while the node count remains at 16, the average runtime rises to 2.31 s.

To ensure practicality for large-scale problems, the algorithm incorporates the following optimizations. 1) It employs shortest weighted path routing to pre-select candidate paths, thereby reducing the number of decision variables in the ILP model. 2) It introduces a period label alignment and queue capacity pre-validation mechanism to discard infeasible solutions in advance, decreasing unnecessary iterations. 3) It leverages GPU parallelism to accelerate the ILP solving process, processing multiple TT flow ILP models concurrently with multi-threading, which yields a 3 – 5 times efficiency gain compared with serial CPU computation. Experimental results demonstrate that, even when the number of network nodes increases to 50 and the number of TT flows reaches 1 000, the algorithm can complete the scheduling plan within 10 s, satisfying the real-time scheduling requirements of WAN scenarios.

3.3 AVB flow scheduling optimization based on dynamic elastic cycles

To maximize the number of successfully scheduled AVB flows in hybrid traffic scheduling, this study proposes an AVB flow scheduling algorithm based on dynamic period adjustment and elastic resource block allocation. This algorithm dynamically adjusts cycle times according to network load, rationally allocates resource blocks, and effectively addresses resource fragmentation issues, thereby enhancing the transmission performance of AVB flows in mixed-traffic environments. The AVB flow scheduling scheme based on dynamic cycle adjustment and elastic resource block allocation is detailed in pseudocode as Algorithm 2.

Algorithm 2 AVB flow scheduling scheme based on dynamic cycle adjustment and elastic resource block allocation.

Input: Network topology $G=(V,E)$, link bandwidth BW, network load factor L_{load}, flow set F_{AVB} to be scheduled and flow parameters, cycle T_{slot}, sliding window W, successfully scheduled the TT flow set $F_{\text{suc}}^{\text{T}}$.

Output: Bandwidth and period allocation for $f_j \in F_{\text{AVB}}$, i. e., successful scheduling of the AVB flow set $F_{\text{suc}}^{\text{A}}$ with dynamic cycle length T_{elastic}.

Calculate the current network load factor based on the usage of fixed and elastic resource blocks. Compare the load against congestion thresholds (0.7/0.3) to dynamically expand or compress the elastic cycle length T_{elastic}.
for each AVB flow $f_j \in F_{\text{AVB}}$ **do**
  Determine the required number of elastic resource blocks N_j by dividing the flow's bandwidth requirement by the bandwidth capacity of a single block.
  for cycle t from T_{current} to $T_{\text{current}}+W$ **do**
    if sufficient available elastic resource blocks exist in cycle t **then**
      Tentatively allocate the resource blocks to flow f_i
      Update the global resource reservation matrix M
      break (proceed to conflict check)
    else (Insufficient resources)
      Trigger the token bucket rate-limiting mechanism to constrain the bandwidth demand of f_i, recalculate the required block count N_j, and re-attempt allocation.
    end if
    Conflict detection: verify if the current allocation conflicts with fixed resource blocks reserved for TT flows (S_{TT}).
    if a temporal conflict is detected **then**
      Shift the allocation to the next valid slot within the sliding window to avoid the conflict
    else
      Confirm the reservation and maintain current allocation
    end if
  end for
end for
return S_{AVB} with updated T_{elastic}

The algorithm initially computes the network load factor according to

$$L_{\text{load}}=\frac{N_{\text{FB}}+N_{\text{EB}}}{N_{\text{total}}} \tag{16}$$

where N_{FB} denotes the number of used FB resource blocks, N_{EB} denotes the number of used EB resource blocks, N_{total} denotes the total number of resource blocks. Subsequently, it dynamically adjusts the parameter T_{elastic} based on

$$T_{\text{elastic}}=\begin{cases}T_{\text{slot}}(1+\alpha); & L_{\text{load}}>a\\ T_{\text{slot}}(1-\alpha); & L_{\text{load}}<b\end{cases} \tag{17}$$

to adapt to current network conditions. When the network experiences heavy-load conditions ($L_{\text{load}}>a$), the cycle is prolonged to ensure adequate bandwidth allocation for AVB flows. Conversely, under light-load scenarios ($L_{\text{load}}<b$), the cycle is shortened to enhance resource utilization. Referencing the congestion threshold design principles of transmission control protocol (TCP) vegas[22] and data center transmission control protocol (DCTCP)[23], link utilization below 30% is classified as a light-load state, while utilization exceeding 70% triggers congestion avoidance mechanisms. Consequently, the experimental design in this chapter adopts the thresholds $a=0.7$ and $b=0.3$.

For AVB flows, the number of required resource blocks N_j is calculated according to

$$N_j \leftarrow \left\lceil \frac{B_j^{\text{A}} T_{\text{elastic}}}{S_{\text{block}}} \right\rceil \tag{18}$$

where S_{block} represents the bandwidth of a single resource block. The sliding window mechanism is then employed to search for available elastic resource blocks over T future cycles. If sufficient available resource blocks are found within the sliding window, these resources are allocated to the AVB flow, and the resource allocation matrix R_{AVB} is updated. If insufficient resource blocks are available, the algorithm triggers the token bucket rate-limiting mechanism. This adjusts the bandwidth demand of the AVB flow through token bucket quotas and recalculates the required number of resource blocks N_j.

Additionally, the algorithm checks for resource conflicts with successfully scheduled TT flows. Should any resource conflict exist, the AVB flow is deferred to the next safe time window.

Upon completing the scheduling process, the algorithm broadcasts the updated elastic cycle T_{elastic}

and the resource allocation matrix R_{AVB} across all network nodes via the centralized network architecture. This ensures coordinated consistency throughout the entire network.

The complete algorithm flowchart is illustrated in Fig. 4. The time complexity primarily depends on the dynamic cycle scaling and elastic resource allocation processes. Within a single network node, dynamic cycle scaling traverses the resource block states within the current cycle window, resulting in a time complexity of $O(N)$. Sliding window preemption only requires querying the states of K queues, exhibiting a complexity of $O(K)$. Elastic resource allocation utilizes a proportional-integral controller for real-time bandwidth adjustment, incurring a constant computational overhead of $O(1)$. Consequently, the per-node algorithm complexity is $O(N+K)$, significantly lower than traditional centralized ILP scheduling methods ($O(2^N)$). For large-scale scenarios involving M nodes, the overall complexity scales to $O(M(N+K))$, demonstrating linear growth with the number of nodes. This aligns with the requirements for large-scale design.

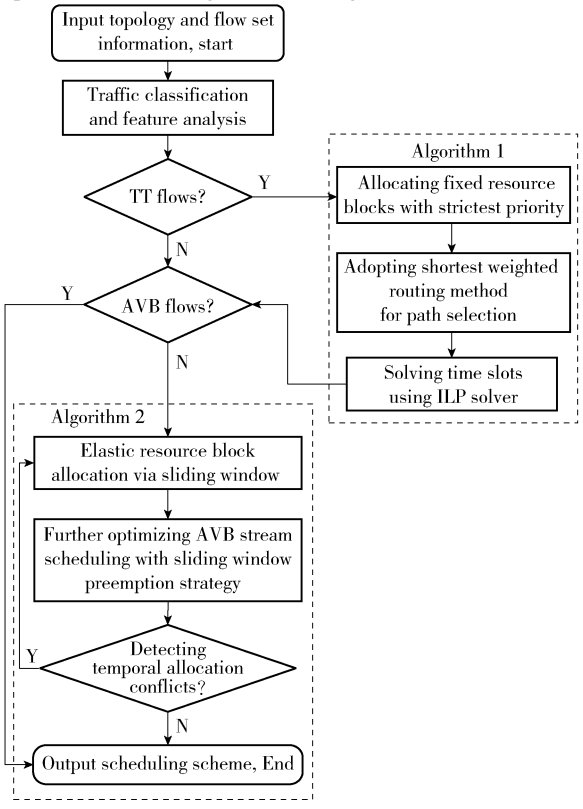

Fig. 4 Hybrid traffic scheduling method flowchart

4 Simulation experiments and results analysis

4.1 Simulation environment design

The simulation environment was implemented using Python programming, with computations executed on a local computer enhanced by graphics processing unit (GPU) acceleration. The hardware configuration comprised an Intel(R) Core (TM) i5-8300H CPU @ 2.30 GHz (8-core processor) and 16GB random access memory (RAM). Computational tasks were offloaded to an NVIDIA RTX 2080 Ti GPU to leverage parallel processing capabilities, thereby accelerating algorithm performance and expediting model convergence.

The acceleration mechanism of the RTX 2080 Ti GPU is primarily realized through the compute unified device architecture (CUDA) architecture, with parallelization strategies covering four aspects: task partitioning, data parallelism, algorithmic optimization, and communication optimization. Task partitioning separates the ILP solving tasks for TT-flow scheduling from the resource-block allocation tasks for AVB flows into independent parallel task sets. The ILP solving tasks are grouped by TT flow, with each GPU core handling the ILP model of 1 – 2 TT flows, while the AVB-flow allocation tasks are grouped by link and processed in parallel. Data parallelism stores global information such as the network topology and the resource reservation matrix in GPU shared memory. Parallel tasks access the data through a memory pool, enabling high-efficiency reads and eliminating redundant data-transfer overhead. Algorithmic parallel optimization uses the single-instruction-multiple-thread (SIMT) architecture during ILP solving. The constraint-validation module evaluates multiple

Data and Code Availability Statement

The datasets generated and analyzed during the current study, along with the complete source code implementing the proposed CSQF-based hybrid traffic scheduling method and GPU-acceleration components, are available in the GitHub repository: https://github.com/xzcvvbxzfd/CSQF_Hybrid_Scheduler.

candidate solutions concurrently, and the dynamic period-adjustment module computes each link's load status in parallel-speeding up period parameter optimization. Communication optimization employs asynchronous data transfer, allowing the CPU and GPU to handle different stages of the workflow simultaneously and reducing waiting time associated with data movement.

The network topology was modeled after the Internet2 backbone network architecture[24], a widely adopted framework for simulating wide-area network scenarios. This topology consists of 16 nodes: 8 edge host nodes (denoted h0 – h7) representing industrial terminal equipment (e. g., sensors, controllers) and 8 core switch nodes (labeled 0 – 7). These nodes interconnect via full-duplex links exhibiting propagation delays ranging from 0. 15 ms to 6. 77 ms. The 0. 15 ms delay corresponds to short-range links typical of intra-factory device communication, while the 6. 77 ms delay emulates long-distance transmission characteristics, analogous to the approximate 5 ms unidirectional fiber delay between Beijing and Shanghai. The topology was constructed as a directed weighted graph using the NetworkX library, with edge weights defined by propagation delays (Fig. 5). Force-directed algorithm (spring layout) visualization ensured a clear and interpretable topological structure for analysis.

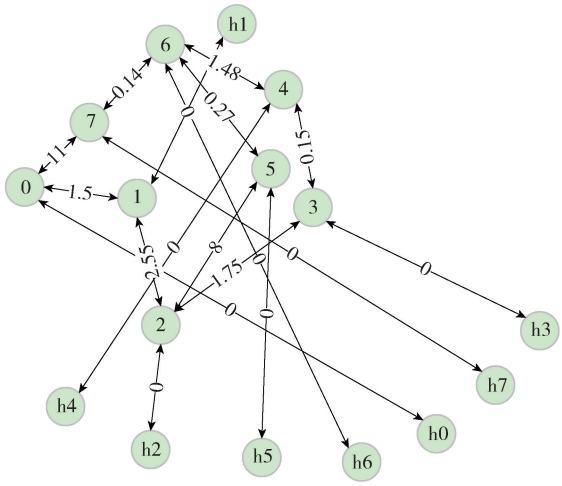

Fig. 5 The wide-area network experimental topology

The traffic generation is categorized into TT flows and AVB flows. TT flows constitute strictly periodic traffic with periods randomly selected from {4 ms, 8 ms, 16 ms, 32 ms}, featuring a fixed packet size of 100 B. The end-to-end deadline is calculated as twice the propagation delay plus three times the period T_{slot}, ensuring jitter remains below the theoretical upper bound. Path selection employs Dijkstra's algorithm, prioritizing low-latency routes using link delay as the weighting metric[25]. AVB flows exhibit randomized bandwidth demands of {10, 20, 50} Mbit/s, with a relaxed deadline of 50 ms, while adhering to the same path selection logic as TT flows. During traffic injection, TT flows transmit strictly according to their periods, with initial offsets uniformly distributed. AVB flows utilize the Poisson distribution to simulate bursty traffic, capturing realistic network fluctuation characteristics. Experiments inject 500 to 3 000 large-scale traffic flows with varying TT/AVB ratios, significantly exceeding typical industrial IoT scenarios.

Algorithmic parameters are designed around dual objectives: deterministic transmission and elastic resource allocation. The global scheduling base period is set at 0. 125 ms, with FB reserved for TT flows. The hypercycle is unified at 32 ms to cover the least common multiple of all stream periods, ensuring scheduling consistency. Each port configures N queues: the first K queues (capacity: 10 packets) are dedicated to TT flows, while the remaining $N - K$ queues serve as EB for AVB flows. The sliding window size of an AVB flow is set to W periods, which limits it from seizing only short-term resources in the future. The dynamic period adjustment range is constrained to 0. 7 to 1. 3 times the base period, the expansion factor is set to 0. 5, and the load factor threshold is divided into light load (< 0.3) and heavy load ($\geqslant 0.7$), triggering cycle compression or expansion.

4. 2 Analysis of experimental results

This experiment compares the performance of three algorithms.

1) The dynamic elastic cycle scheduling algorithm proposed in this paper optimizes resource fragmentation

through dynamic adjustment of cycle length. It introduces a sliding window preemption strategy to limit AVB flows' occupation of future resources and incorporates a token bucket rate-limiting mechanism to ensure bandwidth fairness.

2) Static cycle scheduling: this approach allocates resources using fixed cycles and lacks dynamic adjustment capabilities. Elastic resource block allocation relies solely on sliding window preemption.

3) Preemption strategy without dynamic adjustment: operating within a fixed cycle framework, this strategy allows AVB flows unrestricted preemption of resources in any future cycle but imposes no sliding window constraints.

4.2.1 Impact of workload scale and traffic proportion on scheduling performance

Initially, an experiment was designed to validate the scheduling effectiveness of the proposed algorithm as the flow set size increases. With the TT flow ratio fixed at 10%, the total traffic volume was incrementally increased starting from 500 flows, using a step size of 500, until reaching a total of 3 000 flows. The number of successfully scheduled AVB flows and scheduling efficiency were evaluated.

The experimental results, depicted in Fig. 6, indicate that the AVB flow scheduling success rate decreases as the total flow volume increases.

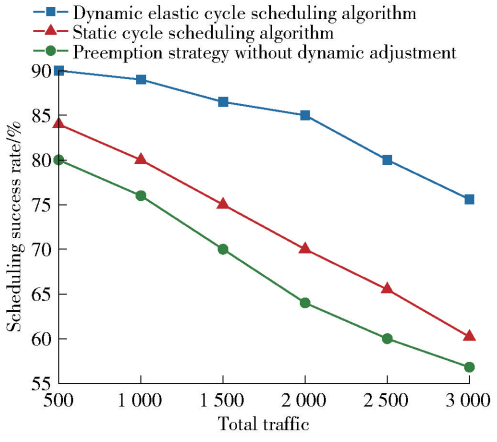

Fig. 6 Scheduling effectiveness of AVB flows under different workload scales

However, the dynamic elastic cycle scheduling algorithm consistently outperforms the other two algorithms. At a total flow volume of 500, the scheduling success rate reaches approximately 90%, showing an improvement of 6% and 10% over static cycle scheduling and the dynamic adjustment-free preemption strategy, respectively.

When the flow volume increases to 3 000, the proposed algorithm maintains a scheduling rate of 75.6%, while the rates for static cycle scheduling and the dynamic adjustment-free preemption strategy drop to 60.2% and 56.8%, respectively. As the flow volume increases from 500 to 3 000, linear regression was used to calculate the average percentage decline in scheduling rate per additional flow. The results show a decline rate of 0.06%/flow for the proposed algorithm, 0.12%/flow for the static cycle scheduling algorithm, and 0.14%/flow for the strategy without dynamic adjustment. This demonstrates the superior load resistance capability of the proposed dynamic elastic cycle scheduling algorithm. Furthermore, TT flows achieved 100% successful scheduling in all scenarios, validating the effectiveness of the fixed resource block isolation mechanism.

To verify the adaptability of the proposed dynamic elastic cycle scheduling algorithm to different traffic proportions within mixed flows, this experiment was designed for five traffic proportion scenarios. Each scenario involves a mixed flow set containing a total of 1 000 flows. The TT flow proportion was adjusted (ranging from 5% to 25%), and the changes in scheduling success rates for both TT flows and AVB flows were analyzed as the proportion of TT flows gradually increased.

Results presented in Fig. 7 indicate an overall decrease in AVB flow scheduling success rates with increasing TT flow proportion. However, the dynamic elastic scheduling algorithm remains significantly superior to comparative schemes throughout all tested ratios. As the TT flows proportion rises from 5% to 25%, the AVB scheduling rate for the proposed algorithm decreases from 95.6% to 63.5%, corresponding to an average decline of approximately 3.2% per 5% TT proportion increment. In contrast, static cycle and non-dynamic strategies exhibit faster

deterioration rates of 4.0% and 4.3%, respectively. Critically, the decline trajectory of the dynamic algorithm notably moderates when TT proportions exceed 15%, demonstrating its effectiveness in mitigating resource fragmentation and long-term conflicts through cycle-length adjustments and sliding window constraints.

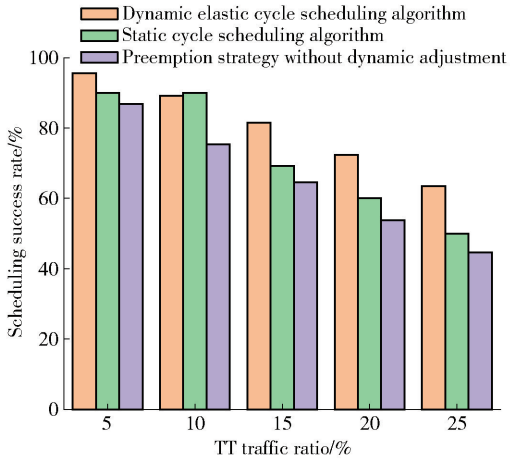

Fig. 7 AVB flow scheduling effectiveness at different TT flow proportions

4.2.2 The impact of dynamic cycle adjustment on resource fragmentation

The resource fragmentation rate (RFR) reflects the proportion of EB bandwidth unused by AVB flows due to discrete distribution. More and scattered fragments make RFR closer to 1, indicating lower resource utilization.

$$R = \frac{B_{\text{frag}}}{C_{\text{total_EB}}} \times 100\% \tag{19}$$

where $C_{\text{total_EB}}$ denotes the total EB block bandwidth capacity, B_{frag} denotes the total bandwidth of fragmented EB blocks.

Experiments were conducted by adjusting the network load factor ($L_{\text{load}} = 0.2, 0.5, 0.8$) to compare the performance differences between the dynamic elastic period algorithm and the static period algorithm. $L_{\text{load}} = 0.2$ simulates a light-load scenario, $L_{\text{load}} = 0.5$ simulates a medium-load scenario, and $L_{\text{load}} = 0.8$ simulates a high-load scenario. The experiment configured a fixed total traffic of 2 000 flows, with TT flows accounting for 10% (i.e., 200 flows). The experimental results are illustrated in Fig. 8. In the light-load scenario, the dynamic period algorithm reduces the resource fragmentation rate to 5% by compressing periods to merge resource blocks, significantly lower than the 15% observed with the static algorithm. Concurrently, the scheduling rate of AVB flows reaches 92.3%, approaching the theoretical optimum of 95%. As the load increases, the dynamic algorithm extends periods under heavy loads to prevent excessive resource segmentation, maintaining a fragmentation rate of only 12%. In contrast, the static algorithm exhibits a fragmentation rate as high as 40% due to fixed periods causing resource dispersion and accumulation.

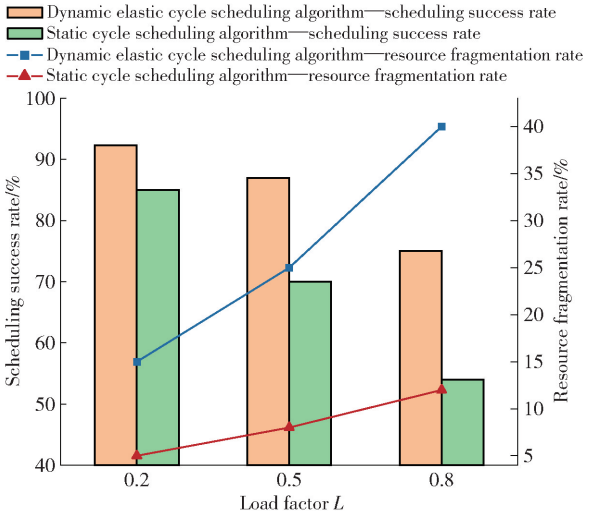

Fig. 8 Resource fragmentation rate and AVB flow scheduling effectiveness under varying loads

This demonstrates that the dynamic period mechanism consolidates discrete idle resource blocks into contiguous elastic resource intervals through period compression, thereby enhancing bandwidth utilization. Experimental data indicate that a 10% reduction in fragmentation rate correlates with an approximate 8.5% improvement in AVB flow scheduling rate.

The TT flows scheduling results under various load factor experimental conditions are presented in Table 1. The proposed algorithm maintains a 0% failure rate across all load factors, while the static strategy exhibits failure rates rising to 5.8% under heavy loads due to deadline violations.

This effectively validates the critical role of the

proposed algorithm in deterministic transmission.

Table 1 TT flows scheduling failure rates under different loads

Load factor	Dynamic cycle method failure rate/%	Static cycle method failure rate/%
0.2	0	0
0.5	0	2.2
0.8	0	5.8

4.2.3 Adaptability verification of the dynamic resilient cycle scheduling algorithm

To further verify the adaptability of the proposed dynamic resilient cycle scheduling algorithm, experimental scenarios with two typical network topologies were designed. Specific topology details are listed in Table 2. Both topologies incorporate core and edge zones, with varying link delays simulating real-world networks. By conducting experiments under these topologies, the performance of three algorithms across different networks was observed to validate the adaptability of the proposed method.

Table 2 Topology parameter information

Network topology	Number of switches	Number of links	link delay/ms
Hierarchical star topology	Core switch: 5 Edge switch: 10	40	Core-to-core: 5 – 10 Periphery-core: 0.5 – 2 Terminal access: 0.1
Ring-based backbone network topology	Core switch: 8 Edge switch: 8	32	Core-to-core: 5 Periphery-core: 1 Terminal access: 0.1

The experimental parameters set the traffic scale to 2 000, with TT flows accounting for 15%. Results shown in Fig. 9(a) present the scheduling success rate of AVB flows, results in Fig. 9(b) display the resource fragmentation rate of EBs. The proposed algorithm demonstrates significant advantages. In the hierarchical star topology, it achieves an 84.3% scheduling success rate, 14.5% higher than static strategies. This improvement stems from dynamically consolidating fragmented resources through cycle adjustment, resulting in a fragmentation rate of only 7.5%. In contrast, non-dynamic preemptive strategies exhibit notable flaws: unrestricted preemption intensifies resource conflicts, yielding the lowest scheduling rate (60.7%) and highest fragmentation rate (35.2%) in the ring topology.

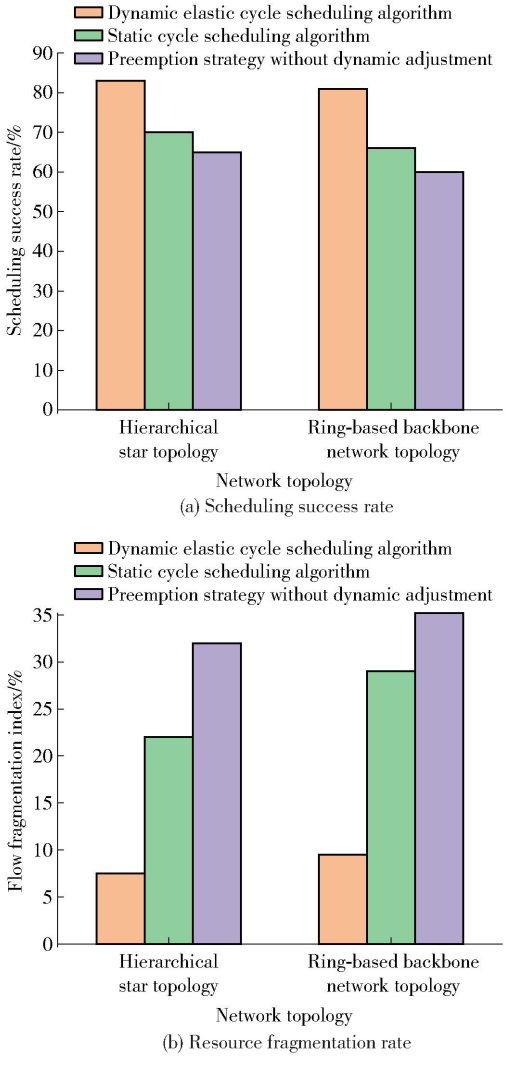

Fig. 9 Performance of three algorithms in scheduling AVB flows under two network topologies

The standard deviation of TT flow jitter quantifies fluctuations in temporal deviations, directly reflecting an algorithm's capability to ensure deterministic transmission. A smaller standard deviation indicates more stable TT flow transmission and higher determinism. Experimental data in Table 3 show that the proposed algorithm, leveraging dynamic resource allocation, maintains jitter standard deviations within

1.5 μs in the star topology and 1.9 μs in the ring topology. Its deterministic guarantee significantly outperforms the other two algorithms, validating the efficacy of the fixed resource block isolation mechanism.

Table 3 Performance of three algorithms in scheduling TT flows under two network topologies

Jitter standard deviation	Dynamic elastic cycle scheduling algorithm/μs	Static cycle scheduling/μs	Preemptive strategy without dynamic adjustment/μs
Hierarchical star topology	1.2±0.3	2.1±0.6	3.5±1.2
Ring-based backbone network topology	1.5±0.4	2.4±0.7	4.0±1.5

4.2.4 Comparison with the latest state-of-the-art algorithms

To further validate the superiority of the algorithm proposed in this paper, two representative state-of-the-art algorithms recently proposed are introduced as comparison baselines. One is OHDSR[12], a TSN mixed-traffic scheduling algorithm proposed in 2023, which employs constraint programming to achieve joint routing and scheduling. The other is DAPS[10], a dependency-aware priority scheduling algorithm proposed in 2024, which is optimized for mixed-criticality traffic. The experimental parameters are set with a traffic scale of 2 000 and a TT-flow proportion of 10%. The comparative experimental results are shown in Table 4.

Table 4 Comparison results between the proposed algorithm and the latest state-of-the-art algorithms

Algorithm	AVB scheduling success rate/%	Resource fragmentation rate/%	TT flow jitter standard deviation/μs	Running time/s
Ours	89.7	8.2	1.3	4.2
OHDSR	78.5	18.6	2.7	7.5
DAPS	82.3	14.3	2.1	5.8

As can be seen from the experimental results in Table 4, the proposed algorithm achieves an AVB scheduling success rate of 89.7%, which is 11.2% higher than that of OHDSR and 7.4% higher than that of DAPS. In terms of resource fragmentation rate, the proposed algorithm reduces it to 8.2%, which is 10.4 and 6.1 percentage points lower than those of OHDSR and DAPS, respectively. Meanwhile, the proposed algorithm exhibits superior TT flow jitter control performance and the shortest running time among the three algorithms.

The advantages of the proposed algorithm stem from its dynamic cycle scaling mechanism and sliding-window preemption strategy. The dynamic cycle scaling mechanism adaptively adjusts the cycle length according to the network load, effectively reducing resource fragmentation; the sliding-window preemption strategy restricts the range of AVB flows' resource occupation, avoiding long-term resource conflicts with TT flows. These two mechanisms work synergistically to better balance the deterministic transmission requirements of TT flows and the bandwidth utilization needs of AVB flows, thus achieving comprehensive performance superiority over the comparative state-of-the-art algorithms.

5 Conclusions

This paper addresses the hybrid traffic scheduling problem in WAN by proposing a dynamic collaborative scheduling scheme based on the CSQF mechanism. Through the design of a deterministic scheduling algorithm for TT flows utilizing ILP, combined with a fixed resource block isolation mechanism, it ensures strict delay constraints for time-sensitive flows. Simultaneously, the introduction of a dynamic elastic period adjustment strategy and a sliding window preemption mechanism optimizes bandwidth utilization and mitigates resource fragmentation issues for AVB flows. Simulation experiments demonstrate that the proposed algorithm achieves a high scheduling success rate for AVB flows across diverse complex scenarios, significantly reducing the resource fragmentation rate, thereby validating its robustness under high-load and stringent real-time requirements. This scheme offers theoretical support and practical reference for efficient collaborative transmission of hybrid traffic in WANs.

Future work will extend the model to support BE

traffic scheduling and design a three-flow priority scheduling mechanism for TT/AVB/BE flows. At the same time, research will be conducted on clock-synchronization error-tolerant algorithms and distributed control architectures to enhance the algorithm's robustness in non-ideal synchronization and distributed deployment scenarios.

References

[1] XU C I, YU H B, JIN X, et al. Industrial Internet for intelligent manufacturing: Past, present, and future. Frontiers of Information Technology & Electronic Engineering, 2024, 25(9): 1173 – 1192.

[2] STÜBER T, OSSWALD L, LINDNER S, et al. A survey of scheduling algorithms for the time-aware shaper in time-sensitive networking (TSN). IEEE Access, 2023, 11: 61192 – 61233.

[3] WANG X L, YAO H P, MAI T L, et al. Joint routing and scheduling with cyclic queuing and forwarding for time-sensitive networks. IEEE Transactions on Vehicular Technology, 2023, 72(3): 3793 – 3804.

[4] NAYAK N G, DÜRR F, ROTHERMEL K. Incremental flow scheduling and routing in time-sensitive software-defined networks. IEEE Transactions on Industrial Informatics, 2018, 14(5): 2066 – 2075.

[5] CHEN S, LEGUAY J, MARTIN S, et al. Load balancing for deterministic networks. Proceedings of the 2020 IFIP Networking Conference, 2020, Jun 22 – 26, Paris, France. Piscataway, NJ, USA: IEEE, 2020: 785 – 790.

[6] EVELEENS J. Ethernet AVB overview and status. Proceedings of the SMPTE 2014 Annual Technical Conference & Exhibition, 2014, Oct 20 – 23, Hollywood, CA, USA. SMPTE (Society of Motion Picture and Television Engineers), 2014: 11p.

[7] KRISHNAN A, LEE H J, EMONTSBOTZ J, et al. Bridging depths and data in mines: 5G-based point cloud mapping in unstructured environments. Construction Robotics, 2024, 8(1): Article 9.

[8] POP P, LANDER RAAGAARD M, CRACIUNAS S S, et al. Design optimization of cyber-physical distributed systems using IEEE time-sensitive networks (TSN). IET Cyber-Physical Systems: Theory and Applications, 2016, 1(1): 86 – 94.

[9] PEI J C, HU Y X, TIAN L, et al. A hybrid traffic scheduling strategy for time-sensitive networking. Electronics, 2022, 11(22): Article 3762.

[10] GUO M, SUN Y F, GU C J, et al. Exploiting dependency-aware priority adjustment for mixed-criticality TSN flow scheduling. Proceedings of the IEEE/ACM 32nd International Symposium on Quality of Service (IWQoS'24), 2024, Jun 19 – 21, Guangzhou, China. Piscataway, NJ, USA: IEEE, 2024: 6p.

[11] CAO Z P, LIU Q R, LIU D P, et al. Enhanced system design and scheduling strategy for switches in time-sensitive networking. IEEE Access, 2021, 9: 42621 – 42634 .

[12] AKRAM B O, NOORDIN N K, HASHIM F, et al. Joint scheduling and routing optimization for deterministic hybrid traffic in time-sensitive networks using constraint programming. IEEE Access, 2023, 11: 142764 – 142779.

[13] BUJOK P, TVRDÍK J, POLÁKOVÁ R. Adaptive differential evolution vs. nature-inspired algorithms: An experimental comparison. Proceedings of the 2017 IEEE Symposium Series on Computational Intelligence (SSCI'17), 2017, Nov 27 – Dec 1, Honolulu, HI, USA. Piscataway, NJ, USA: IEEE, 2017: 8p.

[14] BARZEGARAN M, REUSCH N, ZHAO L, et al. Real-time guarantees for critical traffic in IEEE 802.1 Qbv TSN networks with unscheduled and unsynchronized end-systems. arXiv Preprint, arXiv: 2105.01641, 2021.

[15] RATHEE A. Performance evaluation of THT deployment with different scheduling methods in WSN. International Journal of Computer Applications, 2011, 23(9): 44 – 47.

[16] ZHAI H T, ZHANG Q R, TANG R, et al. RETRACTED ARTICLE: Hybrid traffic security shaping scheme combining TAS and CQSF of time-sensitive networks in smart grid. EURASIP Journal on Wireless Communications and Networking, 2023: Article 106.

[17] WANG J X, YANG S J, ZHUANG L, et al. Multi-objective online hybrid traffic scheduling algorithm in time-sensitive networks. Computer Science, 2023, 50(7): 286 – 292 (in Chinese)

[18] KROLIKOWSKI J, MARTIN S, MEDAGLIANI P, et al. Joint routing and scheduling for large-scale deterministic IP networks. Computer Communications, 2021, 165: 33 – 42.

[19] ZHENG L, ZHANG K Y, WEI G D, et al. Mixed-criticality traffic scheduling in time-sensitive networking using multiple combinatorial packing based on free time domain. Electronics, 2024,13(13): Article 2644.

[20] HUANG J Y, HSU M H, SHEN C A. A novel routing algorithm for the acceleration of flow scheduling in time-sensitive networks. Sensors, 2020, 20(21): Article 6400.

[21] LIU H, MUKERJEE M K, LI C L, et al. Scheduling techniques for hybrid circuit/packet networks. Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies (CoNEXT'15), 2015, Dec 1 – 4, Heidelberg, Germany. New York, NY, USA: ACM, 2015: 13p.

[22] BRAKMO L S, PETERSON L L. TCP vegas: End to end congestion avoidance on a global Internet. IEEE Journal on Selected Areas in Communications, 1995, 13(8): 1465 – 1480.

[23] ALIZADEH M, GREENBERG A, MALTZ D A, et al. Data center TCP (DCTCP). Proceedings of the ACM SIGCOMM 2010 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'10), 2010, Aug 30 – Sept 3, New Delhi, India. New York, NY, USA: ACM, 2010: 63 – 74.

[24] CASTILLO-VELAZQUEZ J I, SERRANO-MARTINEZ D J, MORALES A. Emulation of the connectivity of backbone and management for the layer 3 service of INTERNET2: 2016 topology. Proceedings of the IEEE 37th Central America and Panama Convention (CONCAPAN'17), 2017, Nov 15 – 17, Managua, Nicaragua. Piscataway, NJ, USA: IEEE, 2017: 4p.

[25] ZHENG G, GUO Y X. A routing strategy with link disruption tolerance for multilayered satellite networks. International Journal of Communications, Network and System Sciences, 2010, 3(11): 835 – 842.

(Editor: Luo Lang)