DARS: Dynamic Reliability-Aware Routing and Scheduling for Fault-Tolerant TSN

Wenbin Tian, *Student Member, IEEE*, Changqing Long, *Student Member, IEEE*, Rui Tan, *Senior Member, IEEE*, Shibo He, *Senior Member, IEEE*, and Chaojie Gu, *Senior Member, IEEE*

Abstract—Frame Replication and Elimination for Reliability (FRER) provides fault tolerance by replicating frames and transmitting them along disjoint paths, thereby ensuring ultra-reliable and low-latency communication in TSN networks. However, existing FRER methods with static orchestration strategies, fixed reliability assumptions, and predefined redundancy levels cannot adapt to permanent, transient, or multi-concurrent failures in dynamic network service conditions. To overcome this, we propose DARS, a dynamic reliability-aware routing and scheduling method that incrementally adjusts flow paths and schedules to guarantee reliability and timeliness under evolving faults and network fluctuations. First, we introduce Dynamic Reliability Score (DRS), a new metric that quantifies device reliability and updates it according to flow arrival status, which continuously reflects the service status of traversed devices. Second, we design an incremental adaptive orchestration framework with offline and online strategies, where the former pre-configures all critical flows, while the latter continuously adjusts non-compliant flows. We further develop an FRER-based joint routing and scheduling method that leverages the state-aware DRS to generate adaptive transmission schemes. Experiments on four TSN topologies show that DARS satisfies all flow transmission demands within five iterations under single-fault cases and about 92% of flows under five-fault scenarios, outperforms baseline methods under heavy-load and multi-fault conditions, and reduces redundant transmissions by 5–12.5% through adaptive redundancy.

Index Terms—TSN, FRER, Flow routing and scheduling, Fault tolerance

1 Introduction

Reliable and timely communication are essential for real-time and mission-critical systems [1]–[3]. For instance, smart transportation requires up to 99.9% reliability and 100 ms latency for traffic movement control [4]–[7]. Industrial Internet of Things (IIoT) systems impose 99.99% reliability and 50 ms latency constraints for real-time process control [8], [9]. 3GPP Release 16 sets even stricter standards, an extraordinary 99.9999% reliability and 1 ms latency, for emerging ultra-reliable low-latency communication (URLLC) applications such as drones, telemedicine, and autonomous vehicles [10], [11]. To meet these stringent requirements for time- and reliability-sensitive (TRS) flows, Time-Sensitive Networking (TSN), a subset of IEEE 802.1 standards, has been developed [12]–[14]. TSN augments Ethernet-based network capabilities by providing high Quality of Service (QoS) guarantees, including low latency, bounded jitter, zero packet loss, and ultra-high reliability [15]–[17]. By implementing explicit routing, traffic shaping and scheduling, and resource reservation mechanisms, TSN imposes precise per-hop routing and scheduling, thereby facilitating timely delivery of TRS flows [18]–[20]. Meanwhile, TSN leverages IEEE 802.1CB Frame Replication and Elimination for Reliability (FRER) [21], [22] to enhance transmission reliability. By sending multiple replicated frames (i.e., Ethernet frames) from the talker to the listener via multiple disjoint paths, this spatial redundancy can enable zero-delay recovery from network failures without requiring backup-path switching or retransmissions [23].

However, such end-to-end (E2E) timeliness and reliability guarantees rely on proper routing and scheduling planning. The routing task identifies disjoint paths for replicated flows to provide spatial diversity against failures, while the scheduling task allocates transmission timeslots along these paths to prevent conflicts and satisfy strict latency constraints. Existing studies predominantly adopt offline approaches that derive static transmission schemes and preconfigure them on devices to guarantee E2E delivery in TSN networks [24]–[26]. While these methods offer significant fault tolerance, our observations from practical TSN systems reveal several limitations:

1) **Preconfigured transmission schemes cannot cope with evolving network failures.** Network failures are sporadic and unpredictable, including permanent, transient, and multi-occurring types that static FRER configurations often fail to handle [27]–[29].
2) **Offline strategies lack responsiveness to transient faults.** Network service state (NSS) varies with traffic patterns (e.g., bursty flows), load conditions (e.g., temporary congestion), and device status (e.g., CPU utilization, clock drift, or buffer overflow). Such fluctuations frequently lead to transient faults, which occur more often than hardware-induced permanent failures. However, static schemes cannot adapt promptly to these soft failures [30], [31].

Wenbin Tian, Changqing Long, Shibo He, and Chaojie Gu are with the College of Control Science and Engineering, Zhejiang University, Hangzhou, China (E-mail: {wbtian, lcqzju, s18he, gucj}@zju.edu.cn). Rui Tan is with the College of Computing and Data Science, Nanyang Technological University, Singapore (E-mail: tanrui@ntu.edu.sg).

This work was supported in part by the National Science Fund of China under Grant No. (U23A20296, 62302439). Chaojie Gu is the corresponding author.

3) **Redundancy needs vary at runtime.** During long-term operation under specific NSS conditions, some flows transmit reliably with minimal redundancy, while others require additional redundant paths to maintain service quality. In this case, static strategies with fixed redundancy can lead to excessive bandwidth usage for certain flows and insufficient fault tolerance for others [32], [33].

These limitations arise from three underlying issues: unpredictable fault patterns, rigid transmission schemes, and fixed redundancy levels. These issues pose the following challenges, which remain unaddressed by existing methods.

Challenge 1: How to determine switch reliability (i.e., failure model)? Existing works typically treat switch reliability as a constant representing a device's mature operational state. However, a TSN switch requires not only basic connectivity but also consistently high performance to meet strict QoS guarantees. A static reliability value is insufficient for fine-grained time scales and cannot capture reliability changes caused by NSS variations [34]–[36].

Challenge 2: How to determine the degree of redundancy? FRER entails a reliability–bandwidth trade-off, as redundancy improves fault tolerance but also proportionally increases bandwidth consumption. While prior studies compare network performance under varying redundancy levels [24], [28], none has explored how to determine or adjust redundancy at runtime. Conventional models suggest using k disjoint paths to tolerate up to $k-1$ failures, but unpredictable faults often invalidate this assumption, leaving resources either over-provisioned or insufficient.

Challenge 3: How to dynamically select preferred transmission schemes for redundant frames? Although some FRER works introduce runtime adjustments [37]–[39], they still rely on static assumptions of constant reliability and fixed redundancy. As a result, the generated transmission schemes are often unsuitable, incomplete, or even bandwidth-wasteful when facing unforeseeable, variable, and evolving faults in TSN.

To address these challenges, this paper proposes DARS, a dynamic reliability-aware adaptive routing and scheduling method for fault-tolerant TSN. First, we introduce the Dynamic Reliability Score (DRS), a novel metric that captures time-varying switch service reliability. DRS is iteratively updated based on flow arrival status (FAS), thereby reflecting reliability evolution under network failures and NSS fluctuations. Second, we develop an adaptive orchestration framework consisting of offline and online phases, where the former pre-configures all TRS flows, while the latter iteratively detects and re-orchestrates non-compliant flows based on updated network states. Finally, we propose an FRER-based joint routing and scheduling method that leverages NSS-aware DRS and adjustable redundancy to derive transmission schemes meeting reliability and timeliness constraints. Experiments on four representative TSN topologies demonstrate that DARS satisfies all flow transmission demands within five iterations under single-fault cases and 92% of flows under five-fault scenarios. With dynamic DRS updates, DARS outperforms other methods in heavy load and multi-fault scenarios while achieving a more balanced load distribution. Moreover, adaptive redundancy reduces redundant transmissions by 5–12.5% compared with fixed-redundancy schemes. To summarize, this paper makes the following contributions:

1) We propose *DRS*, a new metric that dynamically quantifies switch reliability, and iteratively updates it based on FAS.
2) We design an incremental orchestration strategy that combines offline preconfiguration with online flow re-orchestration to enhance network adaptability.
3) We develop an FRER-based joint routing and scheduling algorithm that leverages NSS-aware DRS and changeable redundancy to ensure reliable and timely transmission of replicated flows in dynamic TSN environments.

The remainder of this paper is organized as follows: Section 2 introduces background and motivation. Section 3 presents the DRS reliability model. Section 4 proposes an incremental adaptive orchestration framework. Section 5 discusses the joint routing and scheduling algorithm. Section 6 evaluates the performance of the proposed method. Section 7 concludes the paper.

2 Background and Motivation

This section reviews fault tolerance methods, FRER-based routing and scheduling strategies, and device reliability modeling approaches in TSN networks, and motivates the need for a dynamic reliability-aware orchestration framework.

2.1 Fault Tolerance in TSN Networks

Unlike best-effort networking services, TSN requires strict E2E QoS guarantees, which renders fault tolerance a critical design concern [40]. TSN relies on redundancy mechanisms to achieve seamless transmission switchover, thereby maintaining continuous and deterministic service under various failures [41], [42]. Temporal redundancy sends multiple frame copies along the same path to tolerate transient faults. Álvarez et al. [34] develop a time-redundancy mechanism for critical TSN frames and later propose the Proactive Transmission of Replicated Frames method [35] to provide proactive protection against transient failures, which demonstrates substantial reliability gains in OMNeT++ simulations. Feng et al. [36] introduce an offline reservation-based fault-tolerant scheduler for IEEE 802.1Qbv that employs an SMT-based optimizer to allocate time slices for retransmission. However, these approaches realize temporal redundancy through timeslot scheduling, which primarily mitigates transient disturbances (e.g., EMI-induced errors or short-term congestion). Such approaches cannot recover from structural or persistent failures (e.g., device outages or link losses), which undermine the long-term reliability of TSN.

Spatial redundancy, as defined by the IEEE 802.1CB FRER standard, replicates and transmits packets along disjoint paths to enhance delivery reliability. Ergenç et al. [43] implement 802.1CB in OMNeT++ and integrate IS-IS and Shortest Path Bridging into the control plane to support large-scale studies of reliable TSN routing. Maile et al. [44] identify configuration limitations in the 802.1CB standard and provide formal solutions for safe parameter tuning, including the selection between match and vector recovery algorithms, history length, reset timers, and burst-size prediction. Thomas et al. [45] discuss delay penalties caused by packet misordering under FRER and develop

TABLE 1: Comparison of Representative FRER-Based Routing and Scheduling Methods

Method	**Core Mechanism**	**Dynamic Orchestration**	**Dynamic Reliability**	**Dynamic Redundancy**
Lin et al. [51]	GARNet-based routing and fault-tolerant scheduling	×	×	×
Li et al. [52]	ILS-based optimization for throughput/delay/reliability	×	×	×
Ji et al. [24]	FRER orchestration balancing reliability and real-time performance	×	×	×
Feng et al. [25]	Path minimization and AVB degradation under congestion	×	×	×
Zhou et al. [26]	SMT-based heuristic for large-scale configuration	×	×	×
Syed et al. [37]	Heuristic-based flow updates in dynamic TSN in-vehicle networks	✓	×	×
Feng et al. [38]	Hybrid spatial–temporal redundancy with heuristic reorchestration	✓	×	×
Kong et al. [39]	Three-mode failure recovery with SDN controller	✓	×	×
DARS (ours)	**Dynamic reliability-aware adaptive reorchestration**	✓	✓	✓

a network-calculus method for worst-case timing analysis. Overall, these studies elucidate the design, configuration, and implementation of FRER, demonstrating its effectiveness in handling both transient and permanent failures. However, the spatial redundancy paradigm inherently complicates network orchestration, as replicated frames traverse multiple paths and each must conform to specific scheduling mechanisms (e.g., IEEE 802.1Qbv [46]) and resource reservation mechanisms (e.g., IEEE 802.1Qcc [47]) to satisfy temporal constraints and configuration consistency. As a result, beyond reliability mechanisms themselves, orchestration emerges as a critical challenge for FRER, since routing and scheduling decisions must be continuously adjusted to preserve reliability and timeliness when NSS fluctuates or faults evolve [48]–[50].

2.2 Routing and Scheduling for FRER

TSN performs flow routing and scheduling to meet strict transmission constraints, including low latency, bounded jitter, extremely low packet loss, and high transmission reliability [19], [34], [53]–[56]. FRER therefore also requires dedicated routing and scheduling for replicated frames. Multipath routing, a well-studied research topic in traditional networks [57]–[59], has also been explored in combination with FRER. Ergenç et al. [28] discuss the packet loss issue caused by path overlap and propose a new metric, "reassurance," to guide reliability-aware FRER path selection. Hu et al. [60] present an edge-disjoint path selection method to identify independent routes for replicated frames. To satisfy both reliability and timing constraints, researchers also investigate joint routing and scheduling approaches for FRER in TSN. Lin et al. [51] propose a Graph Attention Residual Network (GARNet)-based routing and fault-tolerant scheduling method to improve the reliability and scheduling success rate of TSN flows in power communication networks under permanent failure scenarios. Li et al. [52] develop heuristic algorithms for both routing and scheduling and optimize orchestration schemes through an Iterative Local Search (ILS) framework, which enhances network throughput and link-load balance while maintaining deterministic delay and reliability. Ji et al. [24] integrate the FRER mechanism into Time-Sensitive Software-Defined Networking and balance reliability and real-time transmission performance in industrial IoT systems. Feng et al. [25] propose a joint routing and scheduling algorithm that determines the minimum number of disjoint redundant paths required to meet reliability targets and introduce a service-degradation mechanism for AVB streams under heavy load to balance network reliability and schedulability. Zhou et al. [26] introduce a Satisfiability Modulo Theory (SMT)-based routing and scheduling algorithm that employs incremental synthesis heuristics to improve scalability for large-scale TSN configurations. However, these methods are primarily designed for offline or semi-static planning and therefore cannot directly adapt transmission schemes to evolving network conditions.

To overcome the rigidity of offline methods, several online routing and scheduling algorithms have been proposed for FRER. Syed et al. [37] analyze four dynamic scheduling and routing heuristic methods that incrementally add or remove traffic configurations based on the original setup to support FRER functionality in TSN-based in-vehicle networks. Feng et al. [38] combine spatial and temporal redundancy to address both permanent and transient faults, applying heuristic algorithms for online incremental rerouting and rescheduling of failed traffic. Kong et al. [39] propose a runtime recovery framework that recomputes routing and scheduling upon failures, achieving dynamic redundancy via SDN-based orchestration with three recovery modes: full-functionality, reduced-functionality, and emergency. Although these online approaches support runtime updates, they remain dependent on fixed reliability assumptions and preset redundancy, which leads to homogenized or suboptimal decisions, inefficient resource utilization, and even infeasible schedules under highly dynamic NSS conditions.

2.3 Device Reliability Modeling

Device performance degrades over time as hardware components undergo gradual wear. Device reliability is thus commonly modeled as the probability of fault-free operation over time, typically expressed as $R(t) = e^{-\lambda t}$. In traditional best-effort networks, device reliability is often treated as a constant to characterize devices in a mature operational period, where reliability essentially corresponds to physical availability, manifested as network connectivity. In TSN networks, however, device reliability depends not only on connectivity but also on the ability to sustain QoS and stable high-performance. Consequently, TSN requires fine-grained observation of device reliability to capture device operational conditions. More concretely, as switch core modules (e.g., ports, forwarding fabrics, processing units, and clock synchronization circuits) exhibit performance variations over time due to transient disturbances and permanent failures, their component-level and switch-level reliability also fluctuate accordingly [61]–[63].

Transient faults, typically triggered by non-structural disturbances such as electromagnetic interference (EMI) [64], buffer overflows [65], link jitter, or scheduling anomalies, are usually short-lived and self-recoverable [66], [67]. Although these faults do not physically damage hardware, they can still introduce additional delays, synchronization drift, or timing jitter, which ultimately violate TSN's strict timing guarantees. In contrast, permanent failures, caused by structural problems such as component aging [68], overheating, or functional degradation, result in partial or complete loss of functionality in key modules [69]. These failures not only affect the node itself but also disrupt the global network topology, thereby rendering preconfigured FRER routing and scheduling schemes invalid.

In traditional wired networks, to simplify analysis, many studies treat devices as stable entities, and their reliability is characterized as a constant, measured by the mean time to failure (MTTF), mean time between failures (MTBF), and mean time to repair (MTTR) [70]–[72]. Based on these assumptions, network-level reliability is modeled by Reliability Block Diagrams [73], Fault Trees [74], and Markov Chains [75], [76]. These methods are effective in best-effort networking scenarios in which device availability or network connectivity is the primary concern, and minor performance variations are generally tolerable. However, TSN imposes strict E2E QoS constraints. In such cases, even without physical device failures, runtime NSS fluctuations — including changes in traffic patterns, load conditions, service statuses, and configurations — can directly affect switch behavior and compromise QoS assurance. Therefore, traditional static reliability models are insufficient for TSN, as they fail to capture time-varying and state-dependent service states. Device reliability in TSN should instead be regarded as a dynamic property that evolves with network conditions. To capture such dynamic NSS variations, researchers have explored dynamic reliability modeling approaches, such as stochastic dynamic models [77], Bayesian Networks [78], Stochastic Petri Nets [79], and Dynamic Markov Models (DMMs). However, these models primarily describe system-level failure propagation or probabilistic reliability behavior. Overall, existing device-level static reliability modeling approaches have several limitations in TSN contexts:

1) Their modeling granularity is too coarse to reflect device service states across time scales, hardware components, and flows;
2) They rely heavily on offline statistical data and lack mechanisms for real-time feedback or online adaptation;
3) They are not directly linked to network performance indicators such as latency, jitter, or utilization, which limits their applicability in guiding TSN orchestration decisions.

Consequently, TSN requires a runtime-capable device reliability modeling mechanism that dynamically reflects a device's operational state and evolves in response to NSS variations.

2.4 Limitations and Motivation

As summarized in Table 1, existing FRER-based routing and scheduling approaches still face several critical limitations when dealing with evolving faults and NSS fluctuations in TSN environments.

1) Most TSN systems rely on static configurations with predefined routes and schedules that cannot adapt to changing network conditions [80]–[83]. Although some online orchestration methods exist, they are largely reactive and lack proactive regulation [26], [37].
2) In real-world TSN deployments, networks often encounter unpredictable faults that may recover or persist. However, existing methods adopt fixed reliability assumptions that lack fine-grained modeling and real-time updating, which deviates from the actual runtime states of devices and complicates fault localization and orchestration decisions.
3) Current orchestration algorithms are tightly bound to static parameter assumptions. Without a feedback loop linked to NSS fluctuations and fault dynamics, orchestration strategies tend to produce patch-like adjustments rather than globally optimized, NSS-aware reconfigurations. Moreover, preset redundancy levels often lead to resource imbalance: reliable flows may consume excessive bandwidth, while vulnerable flows remain under-protected.

The above limitations invoke three technical challenges outlined in Section 1: the absence of runtime reliability modeling (Challenge 1), the lack of adaptive redundancy mechanisms (Challenge 2), and the inability to dynamically select appropriate transmission schemes under NSS fluctuations (Challenge 3). To address these challenges, DARS introduces three tightly integrated mechanisms. First, to address Challenge 1, we devise a dynamic device-reliability modeling approach, where port- and switch-level reliability scores continuously update based on runtime FAS, supplanting static reliability assumptions with fine-grained, feedback-driven network state awareness (Section 3). Second, to address Challenge 2, we incorporate an adaptive redundancy mechanism that modulates the replication degree according to aggregated path reliability, yielding bandwidth-efficient fault tolerance under evolving network conditions (Section 5). Third, to address Challenge 3, DARS employs an incremental offline–online orchestration framework that re-orchestrates failed flows on top of an initial static configuration, enabling iterative routing and scheduling adaptation with minimal disruption (Sections 4 and 5). Collectively, these mechanisms

TABLE 2: Main Notations

Symbol	Definition
$G = (V, E)$	Network topology with switches V and links E
$I_v^{(i)}$	Ingress port i of switch v
$O_v^{(j)}$	Egress port j of switch v
f_i	TRS flow i with demand tuple $f_{i,D}$
$f_{i,j}$	j-th replicated instance of flow f_i
$\hat{f}_{i,j}$	Observed j-th replica at listener
R_e	Link reliability
R_v^s	Static hardware reliability of switch v
$R_{v,I/O}^d$	Dynamic port reliability of switch v
$\hat{R}_{v,I/O}^d$	Updated dynamic port reliability of switch v
R_v^d	Switch DRS
$\hat{R}_v^d$	Updated switch DRS
$\alpha, \beta, \gamma, \theta$	Reward weights for the four FAS categories
v_{imp}	Topological importance index of switch v
$\mathcal{H}_{v,I/O}$	Historical port reliability records
L_h	Reliability history window length
C	Capacity set
C^+	Occupied capacity set
C^-	Available capacity set
$\tilde{P}_G$	Precomputed disjoint path set
$\delta_{f_{i,j},v}$	Transmission time of replica $f_{i,j}$ at switch v
sl	Duration of one timeslot
τ	Number of occupied timeslots (ceiled)
$S_{f_{i,j},v}$	Timeslot interval assigned at switch v

elevate FRER-based orchestration from static, parameter-bound schemes to a dynamic, NSS-aware orchestration paradigm for TSN networks.

3 Reliability Modeling with DRS

This section introduces DRS, an iteratively updated reliability score for switches based on FAS, thereby informing reliability-aware path selection and traffic scheduling.

3.1 DRS Definition

To accurately characterize switch service variations and locate network faults, we propose DRS to quantify the time-evolving reliability of each network switch. Network topology is modeled as a graph $G = (V, E)$, where V is the set of TSN switches and E is the set of physical links. Each switch $v \in V$ is equipped with m_v input ports $\{I_v^{(1)}, \dots, I_v^{(m_v)}\}$ and n_v output ports $\{O_v^{(1)}, \dots, O_v^{(n_v)}\}$. Each link $e = (u, v) \in E$ connects two distinct switches u and v, where $u \neq v$. All network components are assigned a reliability value. Each link reliability R_e is treated as a time-invariant constant that reflects long-term hardware availability. TSN switches, which handle flow reception, processing, and forwarding, are particularly sensitive to NSS fluctuations, including clock drift, queueing and forwarding dynamics, processing variability, and hardware degradation. Therefore, the DRS R_v^d of each switch v is defined to integrate both static and dynamic components. The static part R_v^s is also modeled as a time-invariant constant, reflecting the intrinsic functional reliability of the switch hardware. The dynamic parts $R_{v,I}^d$ and $R_{v,O}^d$ capture the runtime reliability variations on the ingress and egress sides of switch v induced by NSS fluctuations. Notably, because each port directly interfaces with its incoming or outgoing link, these aggregated port-side dynamic components $R_{v,I}^d$ and $R_{v,O}^d$ also implicitly account for short-term link-side variations (e.g., bit flips and transient loss), thereby avoiding an additional dynamic link-reliability model. This ingress–device–egress chain follows the classical series configuration in reliability engineering [84], [85], and the overall switch DRS is formulated as

$$R_v^d = R_v^s \times R_{v,I}^d \times R_{v,O}^d. \tag{1}$$

The dynamic reliability values are initialized to 1, corresponding to an error-free operating state, and are subsequently updated based on FAS feedback before being reflected in the aggregated switch-level DRS. The detailed mechanism that reconstructs device-level reliability updates from flow-level FAS observations is presented in Sections 3.3 and 3.4.

3.2 E2E Reliability of TRS Flow

We calculate the E2E reliability of TRS flows as the basis for selecting redundant paths and verifying whether a given transmission scheme satisfies reliability requirements. Each TRS flow f_i is defined by a demand tuple $f_{i,D} = (f_{i,\text{src}}, f_{i,\text{dst}}, f_{i,\text{size}}, f_{i,\text{period}}, f_{i,\text{DDL}}, f_{i,\text{jit}}, f_{i,\text{loss}}, f_{i,\text{rel}})$, which specifies the flow's source, destination, packet size, sending interval, deadline, jitter, loss tolerance, and reliability requirement. Under the FRER mechanism, flow f_i is replicated into K redundant instances $f_i^{\text{repl}} = \{f_{i,1}, \dots, f_{i,K}\}$.

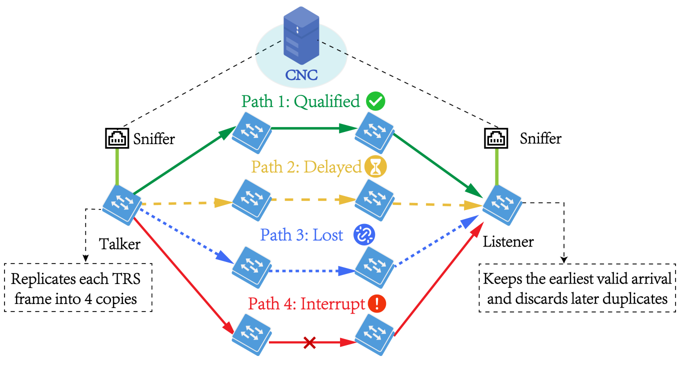

CNC
Path 1: Qualified
Sniffer
Sniffer
Path 2: Delayed
Talker
Listener
Path 3: Lost
Replicates each TRS frame into 4 copies
Keeps the earliest valid arrival and discards later duplicates
Path 4: Interrupt

Fig. 1: FRER process with four-path transmission. The talker replicates each TRS frame into four copies that traverse four disjoint paths. The listener accepts the earliest valid arrival (Path 1) and discards all later or corrupted duplicates. Sniffers at both the talker and listener forward transmission and reception statistics to CNC, which aggregates them into per-replica FAS outcomes: *Qualified*, *Delayed*, *Lost*, or *Interrupted*.

Each replica inherits the demand tuple of the original flow and is transmitted via disjoint paths $P_{f_i} = \{p_{f_{i,1}}, \dots, p_{f_{i,K}}\}$. Each path $p_{f_{i,j}}$ consists of a node set $V_{p_{f_{i,j}}} = \{v_1, \dots, v_n\}$ and a link set $E_{p_{f_{i,j}}} = \{e_1, \dots, e_{n-1}\}$, where $v_1 = f_{i,\text{src}}$, $v_n = f_{i,\text{dst}}$. The E2E reliability of each replica $f_{i,j}$ is the product of the switch- and link-level reliabilities along its path $p_{f_{i,j}}$, expressed as

$$R_{f_{i,j}} = \prod_{v \in V_{p_{f_{i,j}}}} R_v^d \cdot \prod_{e \in E_{p_{f_{i,j}}}} R_e. \tag{2}$$

The aggregated reliability of flow f_i across all replicas must meet the required threshold $f_{i,\text{rel}}$, computed as

$$R_{f_i} = 1 - \prod_{j=1}^{K} (1 - R_{f_{i,j}}) \geq f_{i,\text{rel}}. \tag{3}$$

3.3 FAS and Failure Models

We update the switch DRS based on FAS, which captures the delivery outcome of replicated flows $\hat{f}_{i,j}$. Specifically, as shown in Fig. 1, without requiring any additional protocol support, the centralized controller (i.e., Centralized Network Configuration, CNC) periodically snapshots talker-side transmissions and listener-side receptions of TRS flows over dedicated monitoring channels that are separate from the TSN data plane, and derives the corresponding FAS information from these flow-level observations. As a result, the signaling overhead of FAS feedback scales with TRS flow volume and the monitoring period, rather than overall data-plane traffic. In large-scale deployments, FAS extraction can also be delegated to local monitoring agents (e.g., edge sniffers or domain controllers), which perform flow-level analysis locally and report only aggregated FAS results to CNC, further containing signaling overhead. By aggregating these native transmission statistics, each replica is classified into one of four FAS categories: *Qualified*, *Interrupted*, *Lost*, and *Delayed*. Here, *Qualified* represents full, on-time delivery, while the other three can be associated with permanent, transient, or multi-concurrent failure conditions, as shown in Fig. 2. These FAS labels reflect how devices behave under

NSS fluctuations and therefore serve as feedback signals for updating DRS. When multiple abnormal symptoms co-occur for a replica (e.g., excessive delay and packet loss under severe congestion), FAS assigns a single dominant label according to a predefined severity priority, i.e., *Interrupted* > *Lost* > *Delayed* > *Qualified*, to avoid ambiguous classification and ensure consistent reliability feedback.

Interrupted. Severe service disruptions, such as switch hardware malfunctions, physical link cuts, or critical configuration errors, may prevent flows from being delivered or cause their arrival to far exceed the deadline. In this case, a replica is classified as *Interrupted* if its E2E delay not only exceeds the deadline $f_{i,\text{DDL}}$ but also surpasses the flow's jitter tolerance threshold.

$$t^{\text{e2e}}_{\hat{f}_{i,j}} - f_{i,\text{DDL}} > f_{i,\text{jit}}, \tag{4}$$

where $f_{i,\text{jit}}$ is the jitter tolerance threshold, which is also used to distinguish *Interrupted* from *Delayed*.

Lost. Packet losses may occur due to buffer overflow, transient congestion, or link errors, causing only a portion of the packets in replica $f_{i,j}$ to be successfully delivered. When the observed packet-loss ratio exceeds the flow's loss-tolerance requirement $f_{i,\text{loss}}$, the flow status is classified as *Lost*. The effectively received flow $\hat{f}_{i,j}$ is conservatively abstracted as

$$\hat{f}_{i,j} \le f_{i,j} \cdot (1 - f_{i,\text{loss}}). \tag{5}$$

Delayed. Traffic delay may occur due to transient congestion, scheduling imbalance, or temporary resource contention, where flows are delivered but exceed the deadline. In such cases, flow status is classified as *Delayed*, where the flow E2E latency marginally exceeds its deadline $f_{i,\text{DDL}}$ but remains within the jitter threshold $f_{i,\text{jit}}$.

$$0 < t^{\text{e2e}}_{\hat{f}_{i,j}} - f_{i,\text{DDL}} < f_{i,\text{jit}}. \tag{6}$$

The above categories of FAS *Delayed*, *Lost*, or *Interrupted* can belong to different failure models. Permanent failure refers to unrecoverable malfunctions caused by hardware defects, physical damage, or fatal misconfigurations. Transient failure represents temporary degradations caused by congestion or scheduling imbalance, which may recover with or without external intervention. Multi-concurrent failure is characterized by the simultaneous failure of multiple devices in the network, often leading to complex combined degradations. These mappings bridge flow-level observations with device-level reliability modeling, enabling FAS to serve as a runtime indicator for adaptive orchestration under NSS fluctuations.

3.4 FAS-based DRS Update Algorithm

We design an FAS-based DRS update algorithm that iteratively adjusts DRS values based on FAS feedback, as detailed in Algorithm 1. In each update round, all transmitted replicas $\hat{\mathcal{F}}$ are first classified by their FAS labels. Then, every replica is processed following the inherent severity order of four categories: *Interrupted*, *Lost*, *Delayed*, and *Qualified*. First, each replica $\hat{f}_{i,j} \in \hat{\mathcal{F}}$ is assigned an initial reward w_1 based on its FAS label, i.e., α for *Interrupted*, β for *Lost*, γ for *Delayed*, and θ for *Qualified*, where $-1 < \alpha < \beta < \gamma < 0 < \theta < 1$. The initial reward assignment follows the inherent severity ordering of four

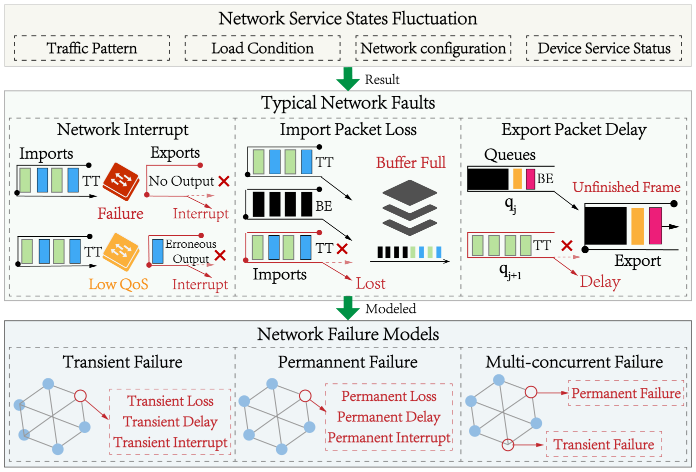


Fig. 2: NSS fluctuations lead to *Interrupted*, *Lost*, and *Delayed* faults, which are modeled as permanent, transient, and multi-concurrent failures.

Algorithm 1: DRS Update

```
Input: $\hat{\mathcal{F}}$, $\mathcal{P}_{\hat{\mathcal{F}}}$, $\alpha, \beta, \gamma, \theta$, $R^s_v$, $R^d_{v,I}$, $R^d_{v,O}$, $v_{\text{imp}}(v)$,
       $N_{v,I}$, $N_{v,O}$, $\mathcal{H}_{v,I}$, $\mathcal{H}_{v,O}$, $r$, $L_h$
Output: $\hat{R}^d_v$
1  Sort $\hat{\mathcal{F}}$ by FAS severity;
2  foreach $\hat{f}_{i,j} \in \hat{\mathcal{F}}$ do
3      $w_1 \leftarrow \begin{cases} \alpha, & \text{if FAS}(\hat{f}_{i,j}) = \textit{Interrupted} \\ \beta, & \text{if FAS}(\hat{f}_{i,j}) = \textit{Lost} \\ \gamma, & \text{if FAS}(\hat{f}_{i,j}) = \textit{Delayed} \\ \theta, & \text{if FAS}(\hat{f}_{i,j}) = \textit{Qualified} \end{cases}$;
4      $p_{\hat{f}_{i,j}} \leftarrow \mathcal{P}_{\hat{\mathcal{F}}}(\hat{f}_{i,j}) = \{v_1, \dots, v_n\}$;
5      $w_2 \leftarrow w_1 / n$;
6      foreach $v \in p_{\hat{f}_{i,j}}$ do
7          $w_3 \leftarrow w_2 / v_{\text{imp}}(v)$;
8          $\{I_v, O_v\} \leftarrow v$;
9          foreach $x \in \{I_v, O_v\}$ do
10             $w_4 \leftarrow w_3 / N_{v,x}$;
11             $\hat{R}^d_{v,x} \leftarrow R^d_{v,x} + w_4$;
12             $H^{\min} \leftarrow (1 - r)\min(\mathcal{H}_{v,x})$;
13             $H^{\max} \leftarrow (1 + r)\max(\mathcal{H}_{v,x})$;
14             $\hat{R}^d_{v,x} \leftarrow \min(\max(\hat{R}^d_{v,x}, H^{\min}), H^{\max})$;
15             $\hat{R}^d_{v,x} \leftarrow \min(\max(\hat{R}^d_{v,x}, 0), 1)$;
16             Update $\mathcal{H}_{v,x}$ and keep last $L_h$ records;
17         $\hat{R}^d_v \leftarrow R^s_v \times \hat{R}^d_{v,I} \times \hat{R}^d_{v,O}$;
```

FAS categories, so failures always receive negative feedback while successful delivery provides positive reinforcement. These weighting parameters are tunable rather than fixed and jointly determine DRS update strength: smaller magnitudes lead to smoother DRS evolution, whereas larger values increase sensitivity to NSS fluctuations. They also control how DRS reacts to different FAS types, since increasing any FAS weight amplifies its impact on DRS update. Since flows traverse different numbers of switches, applying the same reward to long and short paths would bias the update. To avoid this, the reward w_1 is evenly distributed over the traversed switches, i.e., $w_2 = w_1/n$, where n is the number of switches traversed by $\hat{f}_{i,j}$.

Switches also differ in their topological significance, and

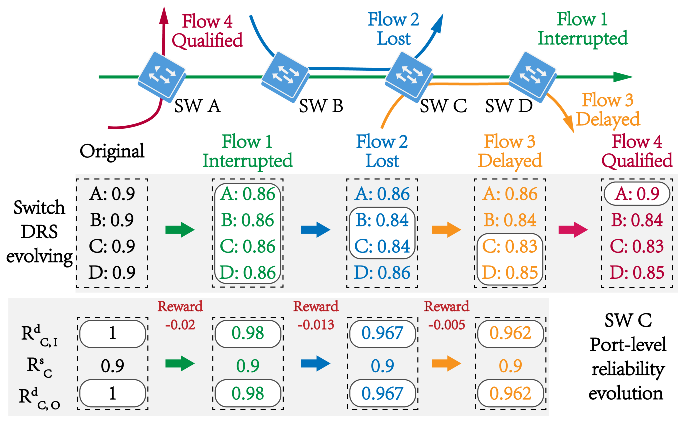


Fig. 3: Stepwise evolution of switch-level DRS and port-level dynamic reliability under different FAS outcomes.

therefore should respond to reliability feedback at different sensitivity levels. Core switches, for example, are expected to exhibit smaller reliability fluctuations in order to preserve network stability. To capture this property, we quantify switch relative importance by a weighted combination of classical graph-theoretic metrics.

$$v_{\text{imp}} = \lambda_{\text{dg}} v_{\text{dg}} + \lambda_{\text{bt}} v_{\text{bt}} + \lambda_{\text{PR}} v_{\text{PR}} + \lambda_{\text{cc}} v_{\text{cc}}, \tag{7}$$

where degree centrality v_{dg}, betweenness v_{bt}, PageRank v_{PR}, and core number v_{cc}, summarized in Appendix A, are linearly normalized into the range $(0, 1]$. The weights $\lambda_{\text{dg}}, \lambda_{\text{bt}}, \lambda_{\text{PR}}, \lambda_{\text{cc}} \geq 0$ tune their relative contributions with $\sum_{\cdot} \lambda_{\cdot} = 1$. The importance index v_{imp} attenuates the reward assigned to the switch, i.e., $w_3 = w_2 / v_{\text{imp}}$, so that topologically critical switches are updated more conservatively, preserving stability while still reflecting reliability variations.

When a flow traverses a switch v, the reward w_3 assigned to that switch is further distributed to its ingress and egress ports ($R^d_{v,I/O}$). However, to prevent heavily loaded ports from being disproportionately penalized, the port reward is further scaled by the number of traversing flows, i.e., $w_4 = w_3 / N_{v,I/O}$, where $N_{v,I/O}$ denotes the number of flows traversing the ingress or egress port. The dynamic port reliability is then updated in an additive manner, i.e., $\hat{R}^d_{v,I/O} \leftarrow R^d_{v,I/O} + w_4$.

To further limit extreme fluctuations in port reliability, each port maintains historical reliability records $\mathcal{H}_{v,I/O}$ over the most recent L_h update rounds. The updated port reliability $\hat{R}^d_{v,I/O}$ is bounded within a range derived from these historical records, defined as

$$(H^{\min}_{v,I/O}, H^{\max}_{v,I/O}) = ((1-r)\min(\mathcal{H}_{v,I/O}), (1+r)\max(\mathcal{H}_{v,I/O})), \tag{8}$$

and the clipped reliability is obtained as

$$\hat{R}^d_{v,I/O} = \min\big(\max(\hat{R}^d_{v,I/O}, H^{\min}_{v,I/O}),\ H^{\max}_{v,I/O}\big), \tag{9}$$

where $r \in (0, 1)$ controls the tolerance margin. Smaller values of r enforce stricter stability, while larger values allow faster adaptation. After each update round, the clipped port reliability values are appended to the historical record $\mathcal{H}_{v,I/O}$, while the oldest entry is discarded to maintain a fixed window of size L_h. All reliability values are additionally bounded within $[0, 1]$ to preserve consistency with reliability modeling principles.

Finally, the updated switch reliability $\hat{R}^d_v$ is obtained by combining static with dynamic port reliability

$$\hat{R}^d_v = R^s_v \times \hat{R}^d_{v,I} \times \hat{R}^d_{v,O}, \tag{10}$$

This multiplicative formulation accounts for ingress and egress port status $\hat{R}^d_{v,I/O}$ while remaining anchored by the static baseline R^s_v, so that both long-term stability and runtime flow behavior contribute to DRS updates.

Complexity analysis. In each update round, the DRS update algorithm processes all monitored replicas and updates ingress and egress reliability along their traversed paths. Since FAS labels fall into four finite categories, replica classification incurs linear time complexity, i.e., $O(|\hat{\mathcal{F}}|)$. For each replica, the algorithm iterates over the switches along its path of length L. At each switch, path-length normalization, topology-aware attenuation, and load-aware port scaling incur constant-time complexity. Historical bounding maintains a bounded reliability history window of length L_h, leading to an additional $O(L_h)$ cost per switch. As a result, the time complexity of one DRS update round is $O(|\hat{\mathcal{F}}| \cdot \bar{L} \cdot L_h)$, where $\bar{L}$ denotes the average path length. Since L_h is a small constant, the update cost scales linearly with the number of monitored replicas and the average path length.

Example. As shown in Fig. 3, switches A, B, C, and D are initially assigned the same static reliability value of 0.9. As all ingress and egress ports start with a reliability value of 1, this results in an initial DRS of 0.9 for each switch. Four TRS flows traverse different segments of the network and are classified as *Interrupted*, *Lost*, *Delayed*, and *Qualified* according to their observed FAS outcomes. Based on these FAS labels, the corresponding rewards are assigned and distributed across the involved switches, leading to stepwise DRS updates on a per-flow basis. More specifically, taking switch C as an example, Fig. 3 illustrates how port-level dynamic reliability evolves under different FAS feedback. When *Interrupted*, *Lost*, and *Delayed* flows traverse switch C, negative rewards of -0.02, -0.013, and -0.005 are applied, resulting in ingress and egress port reliabilities decreasing to 0.98, 0.967, and 0.962. These updates are aggregated with the static reliability, yielding switch-level DRS values of 0.86, 0.84, and 0.83.

4 Incremental Adaptive Orchestration Framework

This section presents the incremental adaptive orchestration framework, which consists of an offline stage for generating initial transmission schemes and an online stage that incrementally adapts these schemes based on runtime FAS feedback.

4.1 Incremental Orchestration Framework Overview

Fig. 4 illustrates the overall orchestration loop, which consists of an offline initialization stage and an online adaptation stage. The offline stage derives baseline transmission schemes under nominal network conditions, while the online stage progressively refines these schemes in response to NSS variations. During the offline stage, the framework estimates initial device reliability, ranks candidate routing paths, and precomputes disjoint paths and corresponding timeslot

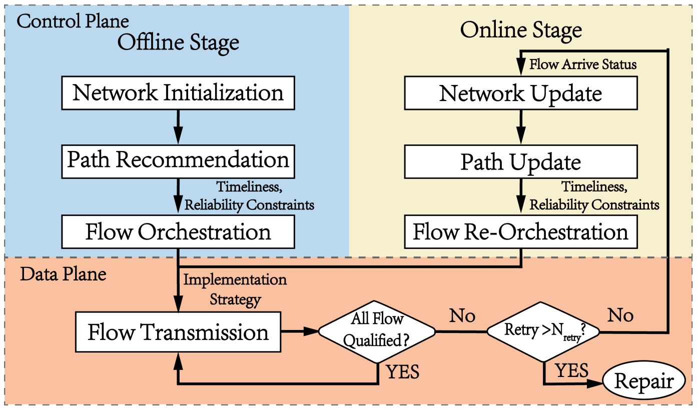


Fig. 4: Incremental orchestration loop with the offline and online stages.

Algorithm 2: Incremental Orchestration Process

```
Input: $F$, $G = (V, E)$, $K$, $C$, $N_{\text{retry}}$
Output: $P_F$, $S_F$
▷ Offline stage
1  $C^- \leftarrow C$; $P_F \leftarrow \emptyset$; $S_F \leftarrow \emptyset$;
2  initialize $R_e$, $R_v^s$, $R_{v,I/O}^d \leftarrow 1$;
3  $\tilde{P}_G \leftarrow$ K-disjoint($G$);
4  foreach $p \in \tilde{P}_G$ do
5      $R_p \leftarrow$ compute path reliability by Eq. (2);
6  Sort $\tilde{P}_G$ by $R_p$;
7  foreach $f_i \in F$ do
8      $P_{f_i}, S_{f_i} \leftarrow$ Joint($f_i, G, K, C^-, \tilde{P}_G$);
9      $P_F[f_i] \leftarrow P_{f_i}$;  $S_F[f_i] \leftarrow S_{f_i}$;
10     $C^- \leftarrow C^- \setminus (P_{f_i}, S_{f_i})$;
▷ Online stage
11 while network runtime do
12     $F_{\text{fail}} \leftarrow \emptyset$;
13     $\hat{F} \leftarrow$ NetworkTransmission($P_F, S_F$);
14     foreach $\hat{f}_i \in \hat{F}$ do
15         if not all copies of $\hat{f}_i$ are qualified then
16             $F_{\text{fail}} \leftarrow F_{\text{fail}} \cup \hat{f}_i$;
17             $C^- \leftarrow C^- \cup (P_F[\hat{f}_i], S_F[\hat{f}_i])$;
18         foreach $\hat{f}_{i,j} \in \hat{f}_i$ do
19             FAS, $p_{\hat{f}_{i,j}} \leftarrow \hat{f}_{i,j}$;
20             $R_{v,I/O}^d \leftarrow$ DRS Update(FAS, $p_{\hat{f}_{i,j}}$);
21     $\tilde{P}_G \leftarrow$ Path Update($R_{v,I/O}^d$);
22     cycle $\leftarrow 0$;
23     while $F_{fail} \neq \emptyset$ and cycle $< N_{retry}$ do
24         cycle $\leftarrow$ cycle + 1;
25         foreach $\hat{f}_i \in F_{fail}$ do
26             $P_{\hat{f}_i}, S_{\hat{f}_i} \leftarrow$ Joint($\hat{f}_i, G, K, C^-, \tilde{P}_G$);
27             $P_F[\hat{f}_i] \leftarrow P_{\hat{f}_i}$;  $S_F[\hat{f}_i] \leftarrow S_{\hat{f}_i}$;
28             $C^- \leftarrow C^- \cup (P_F[\hat{f}_i], S_F[\hat{f}_i])$
```

schedules for each TRS flow. The resulting orchestration schemes guarantee that all TRS flows satisfy their E2E delay and reliability constraints under idealized operating conditions. At runtime, the online stage periodically acquires FAS every T_{detect} to identify transmission anomalies. If all flows are fully delivered and within their deadlines, they are marked as *Qualified* and the current configuration is preserved. Otherwise, any replicated flow classified as *Interrupted*, *Delayed*, or *Lost* is marked as unqualified, thereby initiating the incremental re-orchestration process. During re-orchestration, the DRS values of the involved switches are first updated based on FAS feedback, as described in Section 3, yielding an updated representation of the current NSS. Candidate paths are then reassessed and reordered according to the revised DRS values. Based on the updated rankings, the joint routing and scheduling algorithm is invoked to re-orchestrate unqualified flows, while *Qualified* flows preserve their existing configurations. If more than N_{retry} re-orchestration attempts have failed to recover a flow's quality, this indicates a high likelihood of persistent degradation at specific core devices. In such cases, the corresponding device reliability values are deliberately attenuated in the DRS model, serving as an explicit fault indicator to inform maintenance decisions. The complete orchestration loop, spanning both offline and online stages, is formalized in Algorithm 2, supporting continual adaptation under dynamic network conditions.

4.2 Offline Stage

The offline stage initializes the network environment and precomputes orchestration strategies for all TRS flows before runtime execution.

Network Initialization. We first model the network topology G and assign initial reliability values to each link R_e and switch R_v^s based on all reliability-relevant information available from device specifications and historical operational records. For all switches, the dynamic reliability values of ports $R_{v,I/O}^d$ are initially set to 1, reflecting ideal conditions. At this stage, resources are initialized with no flows deployed, so the overall capacity set C remains fully unoccupied, with the occupied capacity set $C^+ = \emptyset$ and the available capacity set $C^- = C$. These capacity sets compose the solution space for later flow allocation and adjustment.

Path Recommendation. We apply a conventional routing algorithm (e.g., K-disjoint path search) to precompute all disjoint paths $\tilde{P}_G$ for each source-destination pair. Each candidate path is evaluated for E2E reliability according to Eq. (2). These candidate paths are ordered in descending E2E reliability to favor reliable paths for FRER-replicated frames. This pre-ranking mechanism facilitates efficient path retrieval during both initial orchestration and subsequent reconfiguration.

Flow pre-orchestration. With available resource capacity C^- and pre-ranked paths $\tilde{P}_G$, the framework invokes the joint routing and scheduling algorithm (detailed in Section 5) to compute feasible orchestration strategies. Specifically, for each TRS flow $f_i \in F$, the algorithm selects K disjoint routing paths P_{f_i} and assigns a set of transmission slots S_{f_i} along the path to construct the complete E2E transmission scheme. These schemes guarantee that all K flow copies meet their E2E delay and reliability constraints under nominal conditions, thereby providing fault tolerance in the network. The output preconfigured strategies are deployed to the TSN network and serve as the baseline configuration for subsequent online monitoring and adaptation.

4.3 Online Stage

Once the network enters runtime, the orchestration framework transitions into the online stage, which comprises an iterative monitoring and reconfiguration loop.

Flow Arrival Status. During each iterative cycle, all scheduled flows F are transmitted according to the current orchestration scheme. Upon completion, the received flows $\hat{F}$ are evaluated and labeled based on their FAS. If all K replicas of a flow $\hat{f}_i$ are fully delivered within the delay bound, the flow is marked as *qualified* and its current transmission configuration is retained for the next cycle, continuing to occupy the corresponding device resources. If any replica is *Interrupted*, *Lost*, or *Delayed*, the flow is marked as *unqualified* and added to F_{fail}. The detection of any unqualified flows activates re-orchestration to accommodate runtime network failures.

Network Capacity Updates. As unqualified flows will be rescheduled, the network resources they occupy are released and collected into the available capacity set C^-, which serves as the feasible resource set for re-orchestrating failed flows. This update maintains an accurate and up-to-date view of network resources during re-orchestration, thereby reducing the search space and computational overhead of the orchestration process.

Path Re-Ranking. For each received replica, the DRS of devices along its transmission path is updated based on its FAS, thereby altering the E2E reliability of the precomputed paths $\tilde{P}_G$. We recalculate the path reliability with Eq. (2), and re-rank the paths in descending order of reliability. This dynamic ranking enables reliability-aware adaptive path selection, which reflects the latest network conditions and helps failed flows bypass low-reliability devices.

Flow Re-orchestration. If any unqualified flows are detected, the framework proceeds with incremental re-orchestration. For each unqualified flow $f_i \in F_{\text{fail}}$, the joint routing and scheduling algorithm is re-invoked within the updated available capacity C^- and the re-ranked path set $\tilde{P}_G$. Unlike offline pre-orchestration, which considers global capacity and all flows, the online stage focuses on a small set of failed flows and operates under partial resource availability, thereby improving responsiveness and minimizing disruption to already-qualified flows. This DRS-driven adaptation enables the framework to dynamically respond to runtime faults without requiring full network reconfiguration. To deploy the updated orchestration schemes on TSN devices, the network enters a controlled reconfiguration phase, during which data transmission is temporarily suspended, the updated configuration is applied, and the network is restarted after configuration completion. This execution prevents re-orchestration from interfering with in-flight packet forwarding.

5 FRER-based Joint Routing and Scheduling Algorithm

This section presents an FRER-based joint routing and scheduling algorithm that satisfies both the E2E timeliness and reliability requirements of TRS flows under NSS fluctuations.

5.1 E2E Time Model

To characterize the temporal behavior of TRS flows, this section defines their E2E time model and associated scheduling constraints. To alleviate scheduling complexity, the transmission timeline is discretized into integer-based timeslots, which serve as the fundamental scheduling granularity. Each timeslot has a fixed duration sl, and its occupancy is represented by a binary variable, set to 1 if occupied and 0 otherwise. The scheduling task is to assign a set of timeslots for each flow copy at every switch along its path. When a flow copy $f_{i,j}$ traverses a switch v, its transmission time $\delta_{f_{i,j},v}$ is expressed as

$$\delta_{f_{i,j},v} = \tau \cdot sl, \quad \tau = \left\lceil \frac{f_{\text{size}}}{bw_v \cdot sl} \right\rceil, \tag{11}$$

where the occupied timeslot set is $S_{f_{i,j},v} = [sl_a, sl_{a+\tau}]$, and τ denotes the number of occupied slots.

To maintain network-wide timing alignment and scheduling consistency, the concept of a hypercycle (HC) is introduced as the global scheduling period. The HC is defined as the least common multiple (LCM) of the sending intervals of all TRS flows and is synchronized with the GCL cycle length of TAS switches. This relationship is expressed as

$$\text{HC} = \text{LCM}(f_{\text{period}}) = N \cdot sl, \tag{12}$$

where N is the total number of timeslots in one hypercycle.

TSN switches employ the Time-Aware Shaper (TAS) mechanism, standardized in IEEE 802.1Qbv, to precisely regulate scheduled traffic. As illustrated in Fig. 5, incoming flows are mapped to priority queues, and queue transmission is governed by the Gate Control List (GCL). The GCL defines a sequence of time windows of length q_j within a cycle of total duration cl. Thus, the transmission window of flow $f_{i,j}$ at TAS switch v is constrained by

$$S_{f_{i,j},v} = [sl_a, sl_{a+\tau}] \subseteq \left[\sum_{j=1}^{x} q_j, \sum_{j=1}^{x+1} q_j \right), \tag{13}$$

where the occupied timeslot set $S_{f_{i,j},v}$ spans τ timeslots and must lie within the active interval of queue x, indicated by $[x, x+1]$, with $x \in \mathbb{Z}^+$. All switches remain synchronized to the hypercycle, i.e., $HC = cl$, thereby preserving consistent GCL execution and avoiding timing conflicts across the network.

5.2 E2E Latency of TRS Flows

As each TRS flow copy $f_{i,j}$ traverses the switches along its path $p_{f_{i,j}}$, it occupies a sequence of timeslots at each node along the path, forming an E2E transmission schedule

$$\mathbb{S}_{f_{i,j}} = \{S_{f_{i,j},v_1}, S_{f_{i,j},v_2}, \dots, S_{f_{i,j},v_n}\}, \tag{14}$$

where each $S_{f_{i,j},v_k}$ represents τ consecutive timeslots occupied at node v_k. The transmission latency $t^{v_k}_{f_{i,j}}$ of a replica at switch v_k is defined as the accumulated latency from the source node v_1 to v_k, measured as the time difference between the ending timeslot at v_k and the starting timeslot at the source node v_1.

$$t^{v_k}_{f_{i,j}} = sl_{v_k} - sl_{v_1}, \quad \forall v_k \in p_{f_{i,j}}, \text{ s.t. } t^{v_k}_{f_{i,j}} < f_{i,\text{DDL}}, \tag{15}$$

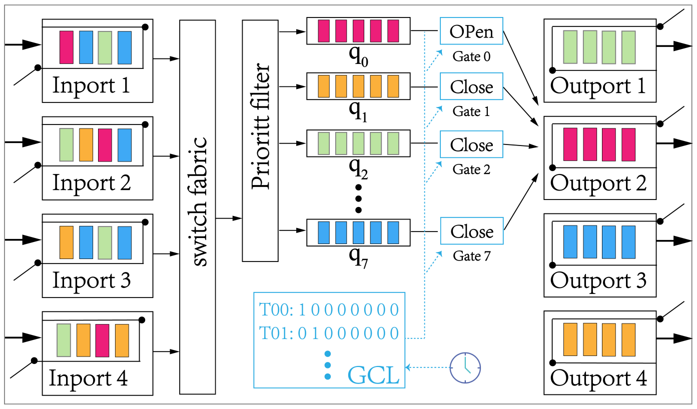


Fig. 5: Overview of TAS mechanisms in a TSN Switch. Traffic enters from four input ports and is distributed to output ports by the switch fabric. The priority filter assigns traffic to queues based on priority. These queues are managed by the GCL, which controls transmission time and order of each queue.

where sl_{v_1} denotes the start timeslot of $S_{f_{i,j},v_1}$ and sl_{v_k} denotes the ending timeslot of $S_{f_{i,j},v_k}$. To guarantee timely delivery, the accumulated latency of each replica must remain within the deadline $f_{i,\text{DDL}}$ specified by the flow f_i.

To achieve zero-delay switchover among redundant paths, the variation in E2E delays Δ_{f_i} across all K flow copies $f_{i,j}$ must remain within its jitter tolerance $f_{i,\text{jit}}$, that is

$$\Delta_{f_i} = \left| \max\left(t^{e2e}_{f_{i,a}}\right) - \min\left(t^{e2e}_{f_{i,b}}\right) \right| < f_{i,\text{jit}}, \quad \forall a, b \in \{1, \dots, K\},\ a \neq b. \tag{16}$$

5.3 Joint Routing and Scheduling Method

Algorithm 3 outlines the joint orchestration process, which plans routing paths and timeslot allocations for each TRS flow f_i. For each flow f_i, given its demand tuple $f_{i,D}$, the FRER mechanism replicates it into K redundant copies $f_i^{\text{repl}} = \{f_{i,1}, \dots, f_{i,K}\}$. These copies are transmitted along candidate paths $\tilde{P}_{f_{i,\text{src}}, f_{i,\text{dst}}}$ derived from the global path set $\tilde{P}_G$ by the source-destination pair $(f_{i,\text{src}}, f_{i,\text{dst}})$. While candidate paths are still available, each flow copy $f_{i,j}$ sequentially selects the candidate path $p_{f_{i,j}}$ from $\tilde{P}_{f_{i,\text{src}}, f_{i,\text{dst}}}$, preferring paths with higher reliability.

For each node v along the selected path $p_{f_{i,j}}$, the orchestration step enumerates all available timeslot segments $\mathbb{S}_{v_i} = \{S_1, S_2, \dots, S_n\}$, where each segment $S_j = [sl_a, sl_{a+\tau}]$ is a contiguous timeslot interval sufficient to accommodate the flow's transmission duration. These segments are sorted chronologically, and the earliest feasible segment S_v is selected and assigned to the current node v. If the delay constraint (Eq. 15) is violated at any node, the current path is skipped and the next candidate is tested.

Once valid timeslot segments are assigned at all nodes, the algorithm computes the flow E2E delay $t^{\text{e2e}}_{f_{i,j}}$ and reliability $R_{f_{i,j}}$. If the current copy is the first successful one, it is directly admitted to the transmission plan. Otherwise, if one or more flow copies have already been scheduled, a jitter check is carried out to verify temporal alignment among all redundant copies. The maximum delay difference Δ_{f_i} is calculated between the new copy and all existing copies of the same flow. If $\Delta_{f_i} < f_{i,\text{jit}}$, the deviation is acceptable and the copy is retained; otherwise, the current path is rejected and an alternative is explored. If all candidate paths fail, the copy is dropped and the orchestration proceeds to the next flow.

In the offline stage, the redundancy K is fixed, and the algorithm proceeds until all K redundant copies are successfully orchestrated. In contrast, during the online stage, the redundancy factor becomes dynamically adjustable according to the current network state and runtime fault conditions. After each copy is scheduled, the orchestration step evaluates the aggregated reliability R_{f_i} with Eq. 3 to determine whether the required threshold $f_{i,\text{rel}}$ has been reached. If the condition is satisfied, the process terminates early to conserve bandwidth; otherwise, additional copies

Algorithm 3: Joint routing and scheduling method

```
Input: $f_i$, $f_{i,D}$, $\tilde{P}_G$, cycle, $C_v^-$, $K$
Output: $P_{f_i}$, $S_{f_i}$
Initialize: $P_{f_i} \leftarrow \emptyset,\ S_{f_i} \leftarrow \emptyset,\ \mathbb{R} \leftarrow \emptyset,\ \mathbb{T} \leftarrow \emptyset,\ \mathbb{N} \leftarrow 0$
Replicate: $f_i^{\text{repl}} = \{f_{i,1}, \dots, f_{i,K}\}$
$\tilde{P}_{f_{i,\text{src}}, f_{i,\text{dst}}} \leftarrow \{p^1, \dots, p^M\} \subseteq \tilde{P}_G$ (sorted by reliability)
foreach $f_{i,j} \in f_i^{repl}$ do
    while $\tilde{P}_{f_{i,src}, f_{i,dst}} \neq \emptyset$ do
        $p_{f_{i,j}} \leftarrow \tilde{P}_{f_{i,\text{src}}, f_{i,\text{dst}}}$
        $\mathcal{S}_p \leftarrow \emptyset$, idx $\leftarrow$ true
        foreach $v \in p_{f_{i,j}}$ do
            $\mathbb{S}_v \leftarrow \{S_1, \dots, S_n\} \subseteq C_v^-$
            sort $\mathbb{S}_v$ by start time; pick earliest $S_v \in \mathbb{S}_v$
            $t^v_{f_{i,j}} \leftarrow$ end$(S_v)$ $-$ start$(S_{v_1})$ (Eq. 15)
            if $t^v_{f_{i,j}} > f_{i,DDL}$ then
                $\tilde{P}_{f_{i,\text{src}}, f_{i,\text{dst}}} \leftarrow \tilde{P}_{f_{i,\text{src}}, f_{i,\text{dst}}} \setminus \{p_{f_{i,j}}\}$
                idx $\leftarrow$ false
                break
            $\mathcal{S}_p \leftarrow \mathcal{S}_p \cup \{S_v\}$
        if $idx$=$true$ then
            $t^{\text{e2e}}_{f_{i,j}} \leftarrow$ end$(S_{v_n})$ $-$ start$(S_{v_1})$ (Eq. 15)
            $R_{f_{i,j}} \leftarrow$ PathRel$(p_{f_{i,j}})$(Eq. 2)
            if $\mathbb{T} \neq \emptyset$ then
                $\Delta_{f_i} \leftarrow \max(\mathbb{T} \cup \{t^{e2e}_{f_{i,j}}\}) - \min(\mathbb{T} \cup \{t^{e2e}_{f_{i,j}}\})$
                if $\Delta_{f_i} > f_{i,jit}$ then
                    discard $p_{f_{i,j}}$
                    continue
            $P_{f_i} \leftarrow P_{f_i} \cup \{p_{f_{i,j}}\}$, $S_{f_i} \leftarrow S_{f_i} \cup \mathcal{S}_p$
            $\mathbb{T} \leftarrow \mathbb{T} \cup \{t^{\text{e2e}}_{f_{i,j}}\}$, $\mathbb{R} \leftarrow \mathbb{R} \cup \{R_{f_{i,j}}\}$,
            foreach $v \in p_{f_{i,j}}$ do
                $C_v^- \leftarrow C_v^- \setminus \{S_v\}$
            $\mathbb{N} \leftarrow \mathbb{N} + 1$
            break
    if $cycle$ = $0$ $(stage$=$offline)$ then
        if $\mathbb{N} = K$ then
            break
    else
        $R_{f_i} \leftarrow$ AggRel$(\mathbb{R})$ (Eq. 3)
        if $R_{f_i} \geq f_{i,rel}$ then
            break
cycle $\leftarrow$ cycle+1
```

are scheduled incrementally until the target reliability is achieved or all candidate paths are exhausted.

5.4 Complexity Analysis

While both offline and online stages adopt the joint routing and scheduling method, they operate under distinct execution workflows and therefore exhibit different computational profiles.

Offline stage. *Network initialization* duplicates flows, assigns device reliability values, and clears resource capacities. The resulting cost grows linearly with the number of flows, nodes, and links, i.e., $O(|F| + |V| + |E|)$, and remains marginal compared with subsequent routing and scheduling operations. *Path recommendation* derives K disjoint candidate paths for each source–destination pair and incurs a complexity of $O\left(K \cdot (|E| + |V| \log |V|)\right)$, while reliability evaluation and ranking introduce a minor $O(K \log K)$ overhead. *Flow pre-orchestration* invokes the joint routing and scheduling algorithm for all TRS flows. For each flow, up to K candidate paths are examined, and per-hop timeslot discovery and constraint verification are performed along a path of length $|L|$ within the available capacity set C^-. The overall complexity of this stage is therefore upper-bounded by $O\left(|F| \cdot K \cdot |L| \cdot \log |C^-|\right)$. Overall, the offline-stage overhead is primarily driven by candidate path computation and per-flow orchestration, which scale with network topology and flow population but only applied in the offline phase.

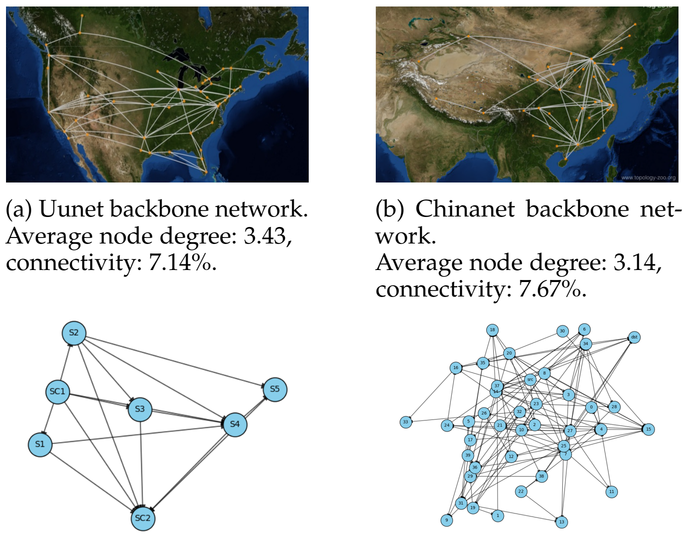
Online stage. *FAS assignment* labels received replicas based on their arrival status and incurs a linear cost of $O(|\hat{F}|)$. *DRS update*, as analyzed in Section 3.4, adjusts reliability scores along the paths traversed by monitored replicas, with a per-iteration cost of $O(|\hat{\mathcal{F}}| \cdot \bar{L} \cdot L_h)$. *Network capacity update* reclaims resources previously occupied by failed flows, which scales linearly with the number of affected resources and is bounded by $O(|V|)$. *Path re-ranking* recomputes and reorders candidate paths according to updated reliability values, incurring $O(K \log K)$ complexity. *Flow re-orchestration* re-invokes the joint routing and scheduling algorithm only for unqualified flows, yielding a complexity of $O\left(|F_{\text{fail}}| \cdot K \cdot |L| \cdot \log |C^-|\right)$, where $|F_{\text{fail}}| \ll |F|$. Overall, although the online stage introduces additional computation during network adaptation and DRS update, it operates only on failed flows. In practice, the number of failed flows depends on fault conditions and is typically limited. Consequently, the online orchestration overhead remains significantly lower than the offline global orchestration cost, rendering the proposed framework well suited for runtime adaptation in TSN networks.

6 Experiments

This section evaluates the adaptability, robustness, and efficiency of the proposed DARS framework under diverse network environments and fault scenarios, exploring DARS performance and underlying mechanisms.

6.1 Setup

Network Topology. We select four representative network topologies, including the public backbone networks Uunet and Chinanet [86], a small-scale industrial network, and a large-scale network. As illustrated in Fig. 6, these topologies span different network scales, connectivity densities, and deployment contexts, thereby covering a wide range of typical TSN scenarios.



(a) Uunet backbone network. Average node degree: 3.43, connectivity: 7.14%.

(b) Chinanet backbone network. Average node degree: 3.14, connectivity: 7.67%.

(c) Small-scale network. Average node degree: 4.29, connectivity: 71.43%.

(d) Large-scale network. Average node degree: 6.62, connectivity: 16.14%.

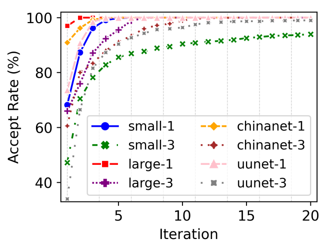
Fig. 6: Experimental network topologies.

Fault Types and Injection. To emulate network faults and NSS fluctuations, we consider *Interrupted*, *Lost*, and *Delayed* fault types, consistent with their FAS definitions in Section 3.3. In addition, *Multi-concurrent* failures are injected by randomly combining multiple fault types, representing complex and compound fault conditions.

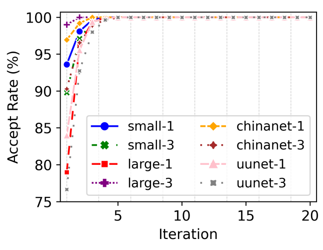
Traffic Configuration. We inject 100 TRS flows into each network. Each flow has a packet size uniformly selected from {32, 64, 128} bytes, a fixed period of 1 ms, and a deadline randomly chosen from {0.5, 1} ms. The reliability requirement of all flows is set to 99.99%, representing typical short-period, high-reliability control traffic in TSN systems.

Metrics. Four metrics are employed to comprehensively evaluate DARS. The accept rate measures the proportion of flows that satisfy both reliability and timeliness constraints, directly reflecting orchestration effectiveness under NSS variations. The E2E delay and jitter quantify the latency and its variation for successfully delivered flows, capturing flow-level timeliness after orchestration. Resource Utilization Standard Deviation (RU-STD) measures the dispersion of timeslot usage among switches per cycle, serving as an indicator of load balancing. DRS Standard Deviation (DRS-STD) characterizes the dispersion of reliability scores among switches per cycle, indicating the stability and consistency of switch reliability across the network.

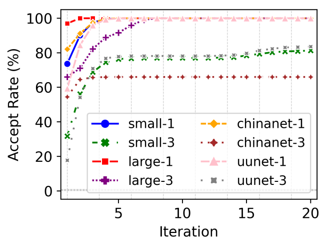
Comparison methods. We review FRER-based routing and scheduling methods in Table 1. Since dynamic orchestration incrementally refines an initial static configuration, the static method results correspond to the first execution round of the dynamic method. Although several studies [37]–[39] explore dynamic FRER orchestration, their design philosophies differ substantially from ours. In particular, [38] jointly consid-



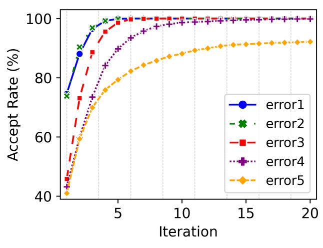
(a) *Delayed*



(b) *Lost*



(c) *Interrupted*



(d) *Multi-concurrent*

Fig. 7: Accept rate convergence under different fault types and intensities across representative network topologies.

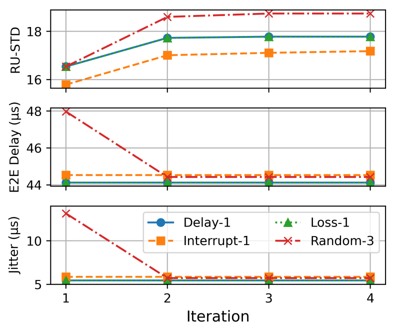
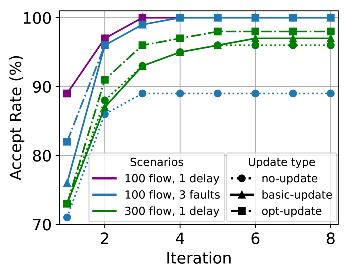
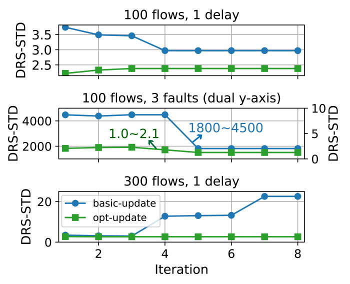


Fig. 8: E2E delay, jitter, and RU-STD under different injected faults in the large-scale network.



(a) Accept rate



(b) DRS-STD

Fig. 9: Comparison of no-update, basic-update, and DARS optimized update methods in the large-scale network.

Method	α	β	γ	θ
Basic	-0.10	-0.01	-0.005	0.001
Optimized	-0.50	-0.05	-0.01	0.01

TABLE 3: Weighting parameter configurations.

ers spatial and temporal redundancy, while [39] focuses on failure-triggered recovery modes. As a result, these approaches do not provide routing–scheduling strategies directly comparable to our work. To enable a fair and meaningful evaluation, we therefore select the first-round static solution and the heuristic-based FRER orchestration (HFO) method [37] as representative baselines.

Implementation. All experiments are implemented in Python, including topology construction, DRS update, path selection, and slot allocation. Each experiment runs for 10 hypercycles, and results are averaged over 20 random seeds to mitigate randomness.

6.2 Fault-Tolerance Evaluation

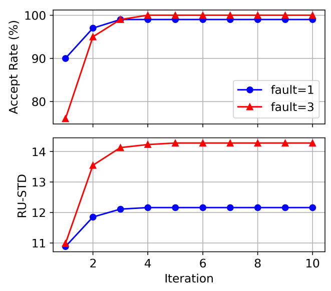
To evaluate the fault adaptation capability and convergence behavior of DARS, we perform iterative orchestration across four network topologies under *Interrupted*, *Lost*, *Delayed*, and *Multi-concurrent* fault scenarios. After each re-orchestration round, performance metrics are recorded to track the system's evolution. Figs. 7(a)–(c) show the accept rate convergence under single- and three-fault scenarios. Across all topologies and fault types, the accept rate increases rapidly and converges within approximately five iterations, indicating that DARS can effectively adapt routing, scheduling, and redundancy decisions in response to runtime faults. The final accept rate, however, strongly depends on fault severity. Under *Lost* faults, all topologies recover to a 100% accept rate, since packet loss can be intrinsically mitigated by FRER replication. Under *Delayed* faults, the accept rate stabilizes between 93% and 100%, reflecting timing violations that can be largely mitigated through schedule adjustment. In contrast, *Interrupted* faults result in lower acceptance levels (66%–100%), as persistent disruptions directly invalidate certain paths and reduce feasible routing options. The impact of fault intensity is further examined by increasing the number of injected faults. With a single fault, all networks eventually recover to full acceptance. When three faults are injected, the initial accept rate drops sharply under *Delayed* and *Interrupted* conditions and does not fully recover even with DARS. Fig. 7(d) further illustrates this trend in the small network. As the number of faults increases from one to five, the initial accept rate decreases from 74% to 41%, and convergence becomes slower. While DARS still converges to nearly 100% with up to four faults, acceptance drops to 92% under five faults. These results suggest that, as fault scale increases, maintaining both E2E timeliness and reliability becomes fundamentally constrained, even with adaptive orchestration. Fig. 8 presents the E2E delay, jitter, and RU-STD in the large-scale network. Under single-fault scenarios, both delay (around 45 μs) and jitter (around 6 μs) remain stable due to limited fault impact. Under *Multi-concurrent* faults, they are initially high but quickly decrease and stabilize at about 45 μs delay and 5μs jitter through DARS re-orchestration, underscoring DARS's network optimization capability. Meanwhile, RU-STD consistently converges across all cases, indicating that DARS not only restores service quality but also maintains balanced load distribution.

6.3 Update Methods Comparison

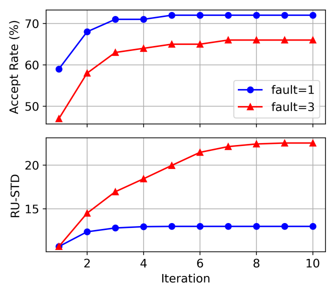
To isolate the contribution of the DRS update mechanism, we conduct ablation experiments comparing no-update, basic-update, and optimized-update strategies. The no-update method disables runtime reliability adaptation, while the basic-update method applies uniform FAS-based rewards w_1 to all traversed switches, where α, β, γ, and θ correspond to the FAS categories *Interrupted*, *Lost*, *Delayed*, and *Qualified*. The optimized-update method adopts the full DRS update algorithm described in Section 3.4. The weighting parameter configurations for the basic-update and optimized-update methods are summarized in Table 3. Fig. 9(a) compares the accept rate under different network settings. Under 100 flows (light load) and a single delay fault, all three methods exhibit similar convergence and reach a 100% accept rate within three iterations, indicating that simple fault conditions do not strongly differentiate update strategies. When the number of faults increases to three, the no-update method degrades to 89%, while both update-based methods maintain a 100% accept rate. With 300 flows (heavier load), none of the methods fully converge; however, the optimized-update method achieves the highest accept rate (98%), outperforming the basic-update (97%) and no-update (96%) baselines. This result demonstrates the benefit of incorporating reliability updates, with the optimized-update slightly outperforming the basic-update. Fig. 9(b) further illustrates the evolution of DRS-STD across iterations. Since DRS directly guides path selection, excessive reliability divergence may bias routing toward a small set of perceived "safe" devices, leading to load imbalance and potential bottlenecks. The optimized-update method consistently maintains lower reliability fluctuations (STD below 5 across switches) and converges faster than the basic-update method, whose STD ranges from approximately 1800 to 4500. This indicates that the proposed update mechanism promotes more balanced path selection. We also observe that increasing FAS initial reward slightly accelerates DRS convergence but has limited impact on DRS-STD, indicating that performance gains primarily stem from the structured DRS update mechanism rather than specific numerical parameter choices, as the update design inherently suppresses parameter-induced fluctuations. Overall, while the optimized-update method provides only modest gains in accept rate compared to the basic-update baseline, it significantly improves reliability stability and load balance, which are critical for long-term robustness and scalability in dynamic TSN environments.

6.4 Redundancy Strategy Comparison

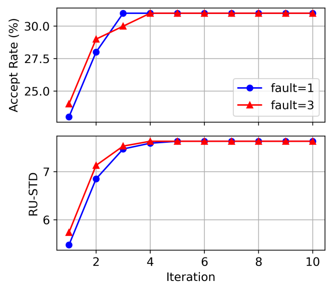
To evaluate the impact of the adaptive redundancy module, we conduct ablation experiments comparing fixed redundancy factors $K = 2, 3, 4$ with the adaptive redundancy strategy. Figs. 10 (a)–(d) show the accept rate and average resource utilization of $K = 2, 3, 4$ and Adaptive-K in the large-scale network under one delay fault and three random faults. With fixed redundancy, the accept rate decreases sharply as redundancy K increases, reaching nearly 100% for $K = 2$, 73% for $K = 3$, and 35% for $K = 4$. This is because, in large network topologies, it is often difficult to find multiple feasible paths to transmit all replicated flows simultaneously. Although resource usage would be expected to increase proportionally with redundancy, the declining acceptance rate alters this trend, leading to average resource utilization values of 12.98 and 22.53 for one and three faults with $K = 3$, which are higher than 12.16 and 14.28 for $K = 2$, and 7.63 and 7.63 for $K = 4$. Overall,



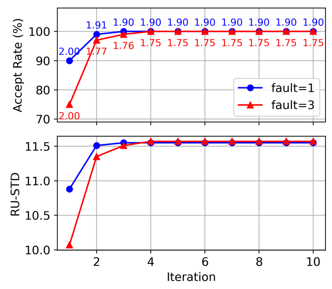
(a) K=2



(b) K=3

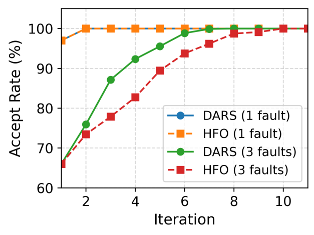


(c) K=4

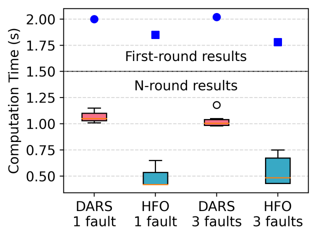


(d) Adaptive K

Fig. 10: Performance of different redundancy strategies.



(a)



(b)

Fig. 11: Comparison of accept rate and computation time under single- and three-delay fault with 100 flows.

$K = 2$ offers the best trade-off between acceptance rate and bandwidth overhead and is therefore the most effective fixed redundancy choice. In comparison, Adaptive-K consistently achieves near-100% acceptance rates with lower resource cost than $K = 2$. We annotate the average redundancy in each cycle in Fig. 10 (d), which illustrates its dynamic adjustment to the runtime network state, decreasing from $K = 2$ to about $K = 1.9$ under one fault and to $K = 1.75$ under three faults. This reduction corresponds to about 5–12.5% fewer redundant copies in low-risk segments, where the scheduler considers a single transmission sufficient for reliability. These results demonstrate that Adaptive-K not only guarantees high reliability but also reduces redundant bandwidth consumption, thereby offering greater flexibility and adaptability.

6.5 Comparison with Existing Methods

We further compare DARS with the static baseline and the HFO method to evaluate their computational efficiency. Under 100 flows (light load) and single- or three-delay fault scenarios, we evaluate the average accept rate and computation time of both offline and online orchestration rounds for HFO and DARS. The offline results represent static method performance, while the online rounds reflect dynamic adaptation capability. As shown in Fig. 11(a), both HFO and DARS improve the accept rate through iterative re-orchestration, indicating that dynamic adaptation outperforms the static baseline as the network configuration is continuously optimized. Under the single-fault scenario, HFO and DARS exhibit identical accept rate evolution and converge within two iterations, suggesting that simple fault conditions can be effectively handled by either method. In contrast, under the three-fault scenario, although both methods start from the same initial accept rate (66%), DARS converges significantly faster, requiring 8 rounds compared to 10 rounds for HFO. This faster convergence highlights the effectiveness of DARS's reliability-aware orchestration, which benefits from the DRS update mechanism that continuously prioritizes more reliable paths. The improved convergence behavior of DARS, however, comes with additional but acceptable computational overhead. As shown in Fig. 11(b), DARS incurs consistently higher computation overhead than HFO, particularly in later iterations due to the DRS update process. On average, DARS requires approximately 0.2 s more computation time than HFO in the first round, and about 0.55 s additional time per iteration in subsequent rounds. Despite this overhead, the computation cost remains stable across iterations and within a practical range, indicating that DARS achieves faster convergence and improved adaptability at the expense of moderate runtime overhead.

7 Conclusion

This paper has presented DARS, an adaptive routing and scheduling framework for TSN fault tolerance based on FRER. By introducing the DRS, DARS models switch reliability as a dynamic metric that reflects runtime service variations through FAS feedback. Built upon this reliability perception, the incremental orchestration process integrates offline pre-orchestration with online reconfiguration, enabling the system to proactively adapt to transient, permanent, and multi-occurrence failures. Furthermore, the proposed joint routing and scheduling algorithm leverages DRS-driven path re-ranking and adjustable redundancy to guarantee both E2E latency and reliability while optimizing bandwidth usage. Extensive experiments on diverse topologies and fault scenarios demonstrate that DARS achieves fast convergence, stable fault-tolerance, and efficient resource utilization, significantly outperforming static or feedback-limited approaches. DARS provides fine-grain reliability-aware orchestration, offering a promising pathway for reliable and timely communications in dynamic deterministic networks.

References

[1] Z. Ma, M. Xiao, Y. Xiao, Z. Pang, H. V. Poor, and B. Vucetic, "High-reliability and low-latency wireless communication for internet of things: Challenges, fundamentals, and enabling technologies," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7946–7970, 2019.

[2] A. Filali, Z. Mlika, S. Cherkaoui, and A. Kobbane, "Dynamic sdn-based radio access network slicing with deep reinforcement learning for urllc and embb services," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 4, pp. 2174–2187, 2022.

[3] M. Ke, Z. Gao, M. Zhou, D. Zheng, D. W. K. Ng, and H. V. Poor, "Next-generation urllc with massive devices: A unified semi-blind detection framework for sourced and unsourced random access," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 7, pp. 2223–2244, 2023.

[4] Y. Wu, H.-N. Dai, H. Wang, Z. Xiong, and S. Guo, "A survey of intelligent network slicing management for industrial iot: Integrated approaches for smart transportation, smart energy, and smart factory," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 2, pp. 1175–1211, 2022.

[5] H. Sun, H. Zhang, W. Guan, X. Liu, H. Ma, and V. C. Leung, "Enhancing low-latency and high-reliability data forwarding for in-vehicle time-sensitive networks," *IEEE Transactions on Vehicular Technology*, vol. 74, no. 8, pp. 11 868–11 881, 2025.

[6] X. Ge, "Ultra-reliable low-latency communications in autonomous vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 5005–5016, 2019.

[7] R. Chen, H. Luo, G. Sun, H. Yu, D. Niyato, and S. Dustdar, "Drdst: Low-latency dag consensus through robust dynamic sharding and tree-broadcasting for iov," *IEEE Transactions on Mobile Computing*, vol. 25, no. 1, pp. 1236–1253, 2026.

[8] A. Nasrallah, A. S. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, and H. ElBakoury, "Ultra-low latency (ull) networks: The ieee tsn and ietf detnet standards and related 5g ull research," *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 88–145, 2019.

[9] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial internet of things: Challenges, opportunities, and directions," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 11, 2018.

[10] 3GPP, "Service requirements for the 5g system (release 16)," *TS 22.261*.

[11] T. Liu, J. Li, F. Shu, and Z. Han, "Optimal task allocation in vehicular fog networks requiring urllc: An energy-aware perspective," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 3, pp. 1879–1890, 2020.

[12] A. Masaracchia, D. van Huynh, T. Q. Duong, O. A. Dobre, A. Nallanathan, and B. Canberk, "The role of digital twin in 6g-based urllcs: Current contributions, research challenges, and next directions," *IEEE Open Journal of the Communications Society*, vol. 6, pp. 1202–1215, 2025.

[13] J. Walrand, "A concise tutorial on traffic shaping and scheduling in time-sensitive networks," *IEEE Communications Surveys Tutorials*, vol. 25, no. 3, pp. 1941–1953, 2023.

[14] J. L. Messenger, "Time-sensitive networking: An introduction," *IEEE Communications Standards Magazine*, vol. 2, no. 2, pp. 29–33, 2018.

[15] W. Tian, C. Gu, M. Guo, S. He, J. Kang, D. Niyato, and J. Chen, "Large-scale deterministic networks: Architecture, enabling technologies, case study and future directions," *IEEE Network*, pp. 1–1, 2024.

[16] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0," *IEEE Industrial Electronics Magazine*, vol. 11, no. 1, pp. 17–27, 2017.

[17] L. Lo Bello and W. Steiner, "A perspective on ieee time-sensitive networking for industrial communication and automation systems," *Proceedings of the IEEE*, vol. 107, no. 6, pp. 1094–1120, 2019.

[18] "Ieee standard for local and metropolitan area networks–timing and synchronization for time-sensitive applications," *IEEE Std 802.1AS-2020 (Revision of IEEE Std 802.1AS-2011)*, pp. 1–421, 2020.

[19] "Ieee standard for local and metropolitan area networks – bridges and bridged networks - amendment 25: Enhancements for scheduled traffic," *IEEE Std 802.1Qbv-2015 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd-2015, and IEEE Std 802.1Q-2014/Cor 1-2015)*, pp. 1–57, 2016.

[20] "Ieee standard for local and metropolitan area networks–virtual bridged local area networks amendment 14: Stream reservation protocol (srp)," *IEEE Std 802.1Qat-2010 (Revision of IEEE Std 802.1Q-2005)*, pp. 1–119, 2010.

[21] "Ieee standard for local and metropolitan area networks–frame replication and elimination for reliability," *IEEE Std 802.1CB-2017*, pp. 1–102, 2017.

[22] M. Karimzadeh-Farshbafan, V. Shah-Mansouri, and D. Niyato, "Reliability aware service placement using a viterbi-based algorithm," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 622–636, 2020.

[23] R. Hofmann, B. Nikolić, and R. Ernst, "Challenges and limitations of ieee 802.1cb-2017," *IEEE Embedded Systems Letters*, vol. 12, no. 4, pp. 105–108, 2020.

[24] L. Ji, S. He, C. Gu, Z. Shi, and J. Chen, "Routing and scheduling for low latency and reliability in time-sensitive software-defined iiot," *IEEE Internet of Things Journal*, vol. 11, no. 7, pp. 12 929–12 940, 2024.

[25] Z. Feng, C. Wu, Q. Deng, Y. Lin, S. Gao, and Z. Gu, "On the scheduling of fault-tolerant time-sensitive networking with ieee 802.1cb," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 43, no. 6, pp. 1715–1728, 2024.

[26] Y. Zhou, S. Samii, P. Eles, and Z. Peng, "Reliability-aware scheduling and routing for messages in time-sensitive networking," *ACM Trans. Embed. Comput. Syst.*, vol. 20, no. 5, may 2021. [Online]. Available: https://doi.org/10.1145/3458768

[27] L. Lewis, "A case-based reasoning approach to the management of faults in communication networks," in *IEEE INFOCOM '93 The Conference on Computer Communications, Proceedings*, 1993, pp. 1422–1429 vol.3.

[28] D. Ergenç and M. Fischer, "On the reliability of ieee 802.1cb frer," in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, 2021, pp. 1–10.

[29] A. Dusia and A. S. Sethi, "Recent advances in fault localization in computer networks," *IEEE Communications Surveys Tutorials*, vol. 18, no. 4, pp. 3030–3051, 2016.

[30] C. Zhou, X. Huang, X. Naixue, Y. Qin, and S. Huang, "A class of general transient faults propagation analysis for networked control systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 4, pp. 647–661, 2015.

[31] B. Ossfeldt, "Maintaining permanent and temporary faults in a communications system," *IEEE Journal on Selected Areas in Communications*, vol. 4, no. 7, pp. 1047–1051, 1986.

[32] O. Chrysaphinou, S. Papastavridis, and P. Sypsas, "Relayed communication via parallel redundancy," *IEEE Transactions on Reliability*, vol. 38, no. 4, pp. 454–456, 1989.

[33] C. Kamyod, R. H. Nielsen, N. R. Prasad, R. Prasad, and N. Aunsri, "End-to-end reliability and optimization of intra and inter-domain ims-based communication networks," *Journal of Cyber Security and Mobility*, vol. 5, no. 3, pp. 233–256, 2016.

[34] I. Álvarez, J. Proenza, M. Barranco, and M. Knezic, "Towards a time redundancy mechanism for critical frames in time-sensitive networking," in *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2017, pp. 1–4.

[35] I. Álvarez, D. Čavka, J. Proenza, and M. Barranco, "Simulation of the proactive transmission of replicated frames mechanism over tsn," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2019, pp. 1375–1378.

[36] Z. Feng, Q. Deng, M. Cai, and J. Li, "Efficient reservation-based fault-tolerant scheduling for ieee 802.1qbv time-sensitive networking," *Journal of Systems Architecture*, vol. 123, p. 102381, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1383762121002629

[37] A. A. Syed, S. Ayaz, T. Leinmüller, and M. Chandra, "Fault-tolerant dynamic scheduling and routing for tsn based in-vehicle networks," in *2021 IEEE Vehicular Networking Conference (VNC)*, 2021, pp. 72–75.

[38] Z. Feng, Z. Gu, H. Yu, Q. Deng, and L. Niu, "Online rerouting and rescheduling of time-triggered flows for fault tolerance in time-sensitive networking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 11, pp. 4253–4264, 2022.

[39] W. Kong, M. Nabi, and K. Goossens, "Run-time recovery and failure analysis of time-triggered traffic in time sensitive networks," *IEEE Access*, vol. 9, pp. 91 710–91 722, 2021.

[40] Y. Seol, D. Hyeon, J. Min, M. Kim, and J. Paek, "Timely survey of time-sensitive networking: Past and future directions," *IEEE Access*, vol. 9, pp. 142 506–142 527, 2021.

[41] S. Kehrer, O. Kleineberg, and D. Heffernan, "A comparison of fault-tolerance concepts for ieee 802.1 time sensitive networks (tsn)," in *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, 2014, pp. 1–8.

[42] P. E. Kehl, J. Ansari, M. Lovrin, P. Mohanram, C.-C. E. Liu, J.-L. L. Yeh, and R. H. Schmitt, "5g-tsn integrated prototype for reliable industrial communication using frame replication and elimination for reliability," *Electronics*, vol. 14, no. 4, 2025. [Online]. Available: https://www.mdpi.com/2079-9292/14/4/758

[43] D. Ergenç and M. Fischer, "Implementation and orchestration of ieee 802.1cb frer in omnet++," in *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2021, pp. 1–6.

[44] L. Maile, D. Voitlein, K.-S. Hielscher, and R. German, "Ensuring reliable and predictable behavior of ieee 802.1cb frame replication and elimination," in *ICC 2022 - IEEE International Conference on Communications*, 2022, pp. 2706–2712.

[45] L. Thomas, A. Mifdaoui, and J.-Y. Le Boudec, "Worst-case delay bounds in time-sensitive networks with packet replication and elimination," *IEEE/ACM Transactions on Networking*, vol. 30, no. 6, pp. 2701–2715, 2022.

[46] "Ieee standard for local and metropolitan area networks–bridges and bridged networks–amendment 29: Cyclic queuing and forwarding," *IEEE 802.1Qch-2017 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd(TM)-2015, IEEE Std 802.1Q-2014/Cor 1-2015, IEEE Std 802.1Qbv-2015, IEEE Std 802.1Qbu-2016, IEEE Std 802.1Qbz-2016, and IEEE Std 802.1Qci-2017)*, pp. 1–30, 2017.

[47] "Ieee standard for local and metropolitan area networks–bridges and bridged networks – amendment 31: Stream reservation protocol (srp) enhancements and performance improvements," *IEEE Std 802.1Qcc-2018 (Amendment to IEEE Std 802.1Q-2018 as amended by IEEE Std 802.1Qcp-2018)*, pp. 1–208, 2018.

[48] A. Kostrzewa and R. Ernst, "Achieving safety and performance with reconfiguration protocol for ethernet tsn in automotive systems," *Journal of Systems Architecture*, vol. 118, p. 102208, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S138376212100148X

[49] Z. Yang, Y. Zhao, F. Dang, X. He, J. Wu, H. Cao, Z. Wang, and Y. Liu, "Caas: Enabling control-as-a-service for time-sensitive networking," in *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications*, 2023, pp. 1–10.

[50] G. Patti, L. L. Bello, and L. Leonardi, "Deadline-aware online scheduling of tsn flows for automotive applications," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 4, pp. 5774–5784, 2023.

[51] Z. Lin, Z. Zeng, Y. Yu, Y. Ren, X. Qiu, and J. Chen, "Graph attention residual network based routing and fault-tolerant scheduling mechanism for data flow in power communication network," *Computers, Materials and Continua*, vol. 81, no. 1, pp. 1641–1665, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S154622182400715X

[52] H. Li, H. Cheng, and L. Yang, "Reliable routing and scheduling in time-sensitive networks," in *2021 17th International Conference on Mobility, Sensing and Networking (MSN)*, 2021, pp. 806–811.

[53] T. Zhang, G. Wang, C. Xue, J. Wang, M. Nixon, and S. Han, "Time-sensitive networking (tsn) for industrial automation: Current advances and future directions," *ACM Comput. Surv.*, vol. 57, no. 2, Oct. 2024. [Online]. Available: https://doi.org/10.1145/3695248

[54] E. Li, F. He, Q. Li, and H. Xiong, "Bandwidth allocation of stream-reservation traffic in tsn," *IEEE Transactions on Network and Service Management*, vol. 19, no. 1, pp. 741–755, 2022.

[55] D. Bujosa, I. Álvarez, and J. Proenza, "Csrp: An enhanced protocol for consistent reservation of resources in avb/tsn," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 5, pp. 3640–3650, 2021.

[56] X. He, X. Zhuge, F. Dang, W. Xu, and Z. Yang, "Deepscheduler: Enabling flow-aware scheduling in time-sensitive networking," in *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications*, 2023, pp. 1–10.

[57] J. Tapolcai, G. Rétvári, P. Babarczi, and E. R. Bérczi-Kovács, "Scalable and efficient multipath routing via redundant trees," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 5, pp. 982–996, 2019.

[58] A. Gopalan and S. Ramasubramanian, "Ip fast rerouting and disjoint multipath routing with three edge-independent spanning trees," *IEEE/ACM Transactions on Networking*, vol. 24, no. 3, pp. 1336–1349, 2016.

[59] P. Kamboj, S. Pal, S. Bera, and S. Misra, "Qos-aware multipath routing in software-defined networks," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 2, pp. 723–732, 2023.

[60] S. Hu, Y. Cai, S. Wang, and X. Han, "Enhanced frer mechanism in time-sensitive networking for reliable edge computing," *Sensors*, vol. 24, no. 6, 2024. [Online]. Available: https://www.mdpi.com/1424-8220/24/6/1738

[61] D. Liberzon and A. Morse, "Basic problems in stability and design of switched systems," *IEEE Control Systems Magazine*, vol. 19, no. 5, pp. 59–70, 1999.

[62] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, 2004.

[63] H. S. Gunawi, R. O. Suminto, R. Sears, C. Golliher, S. Sundararaman, X. Lin, T. Emami, W. Sheng, N. Bidokhti, C. McCaffrey, D. Srinivasan, B. Panda, A. Baptist, G. Grider, P. M. Fields, K. Harms, R. B. Ross, A. Jacobson, R. Ricci, K. Webb, P. Alvaro, H. B. Runesha, M. Hao, and H. Li, "Fail-slow at scale: Evidence of hardware performance faults in large production systems," *ACM Trans. Storage*, vol. 14, no. 3, Oct. 2018. [Online]. Available: https://doi.org/10.1145/3242086

[64] I. Broster, A. Burns, and G. Rodriguez-Navas, "Comparing real-time communication under electromagnetic interference," in *Proceedings. 16th Euromicro Conference on Real-Time Systems, 2004. ECRTS 2004.*, 2004, pp. 45–52.

[65] G. Shen, Q. Li, W. Shi, F. Han, Y. Jiang, and L. Gu, "Poche: A priority-based flow-aware in-network caching scheme in data center networks," *IEEE Transactions on Network and Service Management*, vol. 19, no. 4, pp. 4491–4504, 2022.

[66] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, Y. Ganjali, and C. Diot, "Characterization of failures in an operational ip backbone network," *IEEE/ACM Transactions on Networking*, vol. 16, no. 4, pp. 749–762, 2008.

[67] S. Fang, C. H. Foh, and K. M. M. Aung, "Differentiated congestion management of data traffic for data center ethernet," *IEEE Transactions on Network and Service Management*, vol. 8, no. 4, pp. 322–333, 2011.

[68] D. Kraak, M. Taouil, S. Hamdioui, P. Weckx, F. Catthoor, A. Chatterjee, A. Singh, H.-J. Wunderlich, and N. Karimi, "Device aging: A reliability and security concern," in *2018 IEEE 23rd European Test Symposium (ETS)*, 2018, pp. 1–10.

[69] D. Gessner, J. Proenza, M. Barranco, and A. Ballesteros, "A fault-tolerant ethernet for hard real-time adaptive systems," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 5, pp. 2980–2991, 2019.

[70] A. Villemeur, *Reliability, Availability, Maintainability, and Safety Assessment: Assessment, hardware, software, and human factors*. J. Wiley, 1992, vol. 2.

[71] W. Ahmad, O. Hasan, U. Pervez, and J. Qadir, "Reliability modeling and analysis of communication networks," *Journal of Network and Computer Applications*, vol. 78, pp. 191–215, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804516302776

[72] H. Seddiqi and S. Babaie, "A new protection-based approach for link failure management of software-defined networks," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 4, pp. 3303–3312, 2021.

[73] A. P. Wood, "Multistate block diagrams and fault trees," *IEEE Transactions on Reliability*, vol. R-34, no. 3, pp. 236–240, 1985.

[74] L. Xing, M. Tannous, V. M. Vokkarane, H. Wang, and J. Guo, "Reliability modeling of mesh storage area networks for internet of things," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 2047–2057, 2017.

[75] C. Feng, H. Zhang, S. Yan, Y. Fu, and X. Bao, "Reliability evaluation for distribution system based on probabilistic model checking," in *2017 Second International Conference on Reliability Systems Engineering (ICRSE)*, 2017, pp. 1–6.

[76] S. Dharmaraja, V. Jindal, and U. Varshney, "Reliability and survivability analysis for umts networks: An analytical approach," *IEEE Transactions on Network and Service Management*, vol. 5, no. 3, pp. 132–142, 2008.

[77] W. Ren, J. Wu, X. Zhang, R. Lai, and L. Chen, "A stochastic model of cascading failure dynamics in communication networks," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 5, pp. 632–636, 2018.

[78] C. Zhao, X. Li, Y. Liu, T. Liu, S. Qi, D. Wang, and S. Huang, "Dynamic bayesian network-driven reliability evaluation model for optical networks," in *2025 23rd International Conference on Optical Communications and Networks (ICOCN)*, 2025, pp. 1–3.

[79] Y. Wang, X. Hu, L. Wu, Y. Cai, and X. Gao, "A gspn-based method for dynamic system reliability modelling and evaluation," in *12th International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (QR2MSE 2022)*, vol. 2022, 2022, pp. 338–343.

[80] J. Xu, J. Wang, S. Gao, C. Liu, J. Wang, M. Wei, L. Zhou, and L. Tian, "Research on path selection and flow scheduling of time-sensitive networks for power substations," *Energy Reports*, vol. 9, pp. 1042–1048, 2023, 2022 The 3rd International Conference on Power and Electrical Engineering. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352484723008442

[81] Y. Zhang, Q. Xu, C. Chen, S. Wang, L. Xu, S. Duan, and X. Guan, "Scalable scheduling in industrial time-sensitive networking: A flow graphic distributed scheme," *IEEE Transactions on Industrial Informatics*, vol. 21, no. 2, pp. 1068–1077, 2025.

[82] H. Yu, T. Taleb, and J. Zhang, "Deep reinforcement learning-based deterministic routing and scheduling for mixed-criticality flows," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 8, pp. 8806–8816, 2023.

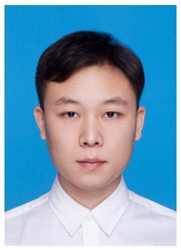
[83] J. Krolikowski, S. Martin, P. Medagliani, J. Leguay, S. Chen, X. Chang, and X. Geng, "Joint routing and scheduling for large-scale deterministic ip networks," *Computer Communications*, vol. 165, pp. 33–42, 2021.

[84] K. Trivedi, *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*. Wiley, 2016. [Online]. Available: https://books.google.com/books?id=OJ6tDAAAQBAJ

[85] C. Ebeling, *An Introduction to Reliability and Maintainability Engineering: Third Edition*. Waveland Press, 2019. [Online]. Available: https://books.google.com/books?id=rh2WDwAAQBAJ

[86] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.

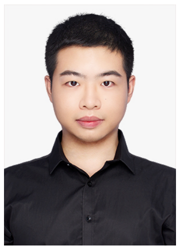
Wenbin Tian (Student Member, IEEE) received the B.Eng. degree in Mechanical and Automation and the M.Eng. degree in Control Science and Engineering from Chongqing University, Chongqing, China, in 2019 and 2021. He is currently working toward the Ph.D. degree with the College of Control Science and Engineering, Zhejiang University, Hangzhou, China. Her research interests include IoT, Deterministic Networking and LLM.



Changqing Long (Student Member, IEEE) received the B.S. degree in Applied Mathematics and the M.S. degree in Operational Research and Cybernetics from the School of Mathematics and Statistics, South-Central Minzu University, Wuhan, China, in 2018 and 2021, respectively. He is currently pursuing the Ph.D. degree in control science and engineering with the College of Control Science and Engineering, Zhejiang University, Hangzhou, China. His current research interests include Edge Computing, Stability Analysis, and Machine Learning.

Rui Tan (Senior Member, IEEE) is a Full Professor at College of Computing and Data Science, Nanyang Technological University, Singapore. Previously, he was a Research Scientist (2012-2015) and a Senior Research Scientist (2015) at Advanced Digital Sciences Center of University of Illinois at Urbana-Champaign, and a postdoctoral Research Associate (2010-2012) at Michigan State University. He received the Ph.D. (2010) degree in computer science from City University of Hong Kong, the B.S. (2004) and M.S. (2007) degrees from Shanghai Jiao Tong University. His research interests include cyber-physical systems and Internet of things. He is the recipient of Best Demo Award from SenSys'24, Best Paper Awards from ICCPS'23 and IPSN'17. He served as Associate Editor of IEEE Transactions on Mobile Computing and ACM Transactions on Sensor Networks, TPC Co-Chair of e-Energy'23, EWSN'24, SenSys'24, and General Co-Chair of eEnergy'24 and RTCSA'25. He received the Distinguished TPC Member recognition from SenSys in 2025 and from INFOCOM in 2017, 2020, and 2022.

Shibo He (Senior Member, IEEE) received the Ph.D. degree in control science and engineering from Zhejiang University, Hangzhou, China, in 2012. He is currently a professor with Zhejiang University. He was an associate research scientist from March 2014 to May 2014 and a post-doctoral scholar with Arizona State University, Tempe, AZ, USA, from May 2012 to February 2014. From 2010 to 2011, he was a visiting scholar with the University of Waterloo, Waterloo, ON, Canada. His research interests include the Internet of Things, crowdsensing, and Big Data analysis. He serves/served on the editorial board for IEEE Transactions on Network Science and Engineering, IEEE Transactions on Vehicular Technology, Peer-to-Peer Networking and Application (Springer), and KSII Transactions on Internet and Information Systems. He was the general chair for AIoTsys 2024 and general co-chair for iSCI 2022, the symposium co-chair for the IEEE/CIC ICCC 2022, IEEE GLOBECOM 2020, and the IEEE ICC 2017, the TPC co-chair for i-Span 2018, the Finance and Registration chair for ACM MobiHoc 2015, the TPC co-chair for the IEEE ScalCom 2014, the TPC vice co-chair for ANT 2013 and 2014, the Track co-chair for the Pervasive Algorithms, Protocols, and Networks of EUSPN 2013, the Web co-chair for the IEEE MASS 2013, and the Publicity co-chair of IEEE WiSARN 2010 and FCN 2014.



Chaojie Gu (Senior Member, IEEE) received the B.Eng. degree in information security from the Harbin Institute of Technology, Weihai, China, in 2016, and the Ph.D. degree in computer science and engineering from Nanyang Technological University, Singapore, in 2020. He was a Research Fellow with Singtel Cognitive and Artificial Intelligence Lab for Enterprise, in 2021. He is currently an Assistant Professor with the College of Control Science and Engineering, Zhejiang University, Hangzhou, China. His research interests include IoT, industrial IoT, edge computing, and low-power wide area network.

APPENDIX

NODE IMPORTANCE METRICS

To quantify the relative role of switches in the network, we adopt four classical graph-theoretic metrics: degree centrality, betweenness centrality, PageRank, and core number.

Degree Centrality (v_{dg})

Degree Centrality reflects the number of direct connections of a node, indicating its activity in the local network. For a node v, it is defined as:

$$v_{\text{dg}}(v) = \frac{k_v}{|V| - 1} \tag{17}$$

where k_v denotes the degree of node v, and $|V|$ is the total number of nodes in the network.

Core Number (v_{cc})

The core number originates from k-core decomposition and represents the maximum k such that the node belongs to a k-core subgraph, where each node has degree at least k. A higher core number indicates that the node is more structurally embedded in the network.

Betweenness Centrality (v_{bt})

Betweenness centrality measures the extent to which a node acts as a bridge in the network, i.e., how many shortest paths pass through it. For a node v:

$$v_{\text{bt}}(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{18}$$

where σ_{st} is the number of shortest paths between nodes s and t, and $\sigma_{st}(v)$ is the number of those paths passing through node v.

PageRank (v_{PR})

PageRank evaluates a node's importance by considering not only its number of connections but also the importance of its neighbors. It is defined as:

$$v_{\text{PR}}(v) = \frac{1-d}{|V|} + d \sum_{u \in N(v)} \frac{v_{\text{PR}}(u)}{k_u} \tag{19}$$

where d is the damping factor (typically 0.85), $N(v)$ is the set of neighbors pointing to v, and k_u is the out-degree of node u.