aerospace

MDPI

Article

Deadline-Aware Collaborative Reconstruction for Spaceborne TSN-RDMA Interconnects

Wenjie Zhao [1,2], Jianing Rao [1,2,*], Wanying Zhang [1], Enshi Li [1,2], Guodong Yin [1,2], Chen Jin [1,2] and Junshe An [1,2]

[1] National Space Science Center, Chinese Academy of Sciences, Beijing 101400, China; zwjok123@126.com (W.Z.); zhangwanying@nssc.ac.cn (W.Z.); lienshi23@mails.ucas.ac.cn (E.L.); egdon0216@163.com (G.Y.); a1964424806@163.com (C.J.); anjunshe@nssc.ac.cn (J.A.)
[2] University of Chinese Academy of Sciences, Beijing 100190, China
* Correspondence: raojianing@nssc.ac.cn

Abstract

Recovering a spaceborne interconnect requires both a usable network path and endpoint state that can support timely communication. We propose coordinated scheduling and collaborative reconfiguration (CSCR) for integrated time-sensitive networking and remote direct memory access. CSCR prepares network and endpoint recovery in parallel, then activates the configuration at a common gate-cycle boundary. A synthetic message-level evaluation used 44 flows, four individual functional fault classes and 60 paired random seeds. Under the assumed timing inputs, CSCR achieved a 6.157-ms P95 time to the first timely critical-message completion, a 0.875% mean critical deadline-violation rate and 6146.1-KiB P95 peak queued data. Its P95 confirmation time for all critical flows to meet their deadlines over a 10-ms release window was 16.143 ms. The 35.3% reduction in first-completion P95 relative to reactive route-only recovery reflects the complete policy, including different configuration and decision durations. Matched-input controls separate preparation concurrency and gate-phase alignment. Additional queue simulations examine control latency, preparation severity and background load. These results characterize recovery policies within a gated fluid-service model; they do not establish hardware recovery performance or spacecraft deployment readiness.

Keywords: time-sensitive networking; RDMA; fault recovery; spaceborne interconnects; deadline constraints; configuration consistency

1. Introduction

Spacecraft processing chains carry control messages alongside sustained payload data. An interruption can push sensor processing, coordination or storage beyond an application deadline, even after connectivity returns. Recovery must therefore be judged by the service it restores. Time-sensitive networking (TSN) provides synchronized time and bridge configuration [1,2], allowing forwarding schedules to be coordinated with endpoint readiness.

TSN offers several ways to protect and regulate traffic. Frame replication and elimination support redundant delivery [3]; scheduled traffic and frame preemption control transmission opportunities and interference [4,5]. Per-stream filtering, cyclic queuing and asynchronous shaping govern traffic conformance and service [6–8]. Here, a time-aware gate schedule governs transmission, while a separate policy governs recovery.

Fronthaul and aerospace profiles define deployment contexts [9,10], while DetNet extends deterministic service to routed networks [11,12]. TSN surveys, schedulability

Check for updates

Academic Editor: Hyun-Ung Oh

Received: 2 October 2026
Revised: 13 November 2026
Accepted: 17 November 2026
Published: 18 November 2026

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

studies and application integration identify the timing and configuration constraints that recovery must preserve [13–19].

Time-aware and no-wait scheduling formulate transmission constraints [20,21], and SMT-based synthesis constructs time-triggered schedules [22]. Software-defined TSN links scheduling to configuration [23]. Network-calculus studies of TTEthernet and formal shaper analysis establish tools for bounding service delay [24,25]. These tools can assess a recovery candidate, but the assessment must remain valid when that candidate is activated.

Scheduling and implementation studies address nominal latency and configuration choices [26–30]. These provide context for recovery but do not establish the joint restoration of a forwarding schedule and RDMA endpoint readiness.

Onboard implementations make the integration problem concrete. TSN buses for microlaunchers, hybrid processing–communication scheduling in integrated modular avionics and lite TSN for launcher services address timing across system components [31–33]. Software-defined in-vehicle networks and heterogeneous factory access also examine configuration across communication subsystems [34,35]. An integrated TSN–RDMA recovery procedure must account for endpoint preparation as well as the forwarding schedule.

SpaceWire and SpaceFibre provide established onboard communication alternatives [36–39]; radiation assurance concerns component qualification [40]. Standardization, backplane and link-reliability studies address complementary construction issues [41–44].

Other onboard, CAN and FlexRay studies illustrate platform-specific communication and scheduling requirements [45–55]. We do not transfer their fault or transport behavior into the Ethernet queue model.

Remote direct memory access (RDMA) adds endpoint and transport state to the network recovery problem. The InfiniBand architecture defines the protocol framework [56]. Congestion-control and feedback studies address loading [57,58], while network-support studies and design guidelines examine transport and application requirements [59,60]. RDMA key-value services illustrate how remote operations enter application execution [61]. Restoring a path alone may therefore leave the service unable to resume.

RDMA transaction and memory-semantics studies show why endpoint consistency matters [62–64]. Work on datagrams, Ethernet deployment, storage, RoCE, MPI and wide-area transport supplies additional application context [65–70]; none calibrates the recovery durations assumed here.

The recovery problem lies at the interface between forwarding, timing and endpoint state. FRER protects a localized path fault while a valid replica remains available [3], but it does not remove the need to restore shared endpoint state. Resumed traffic must still meet application deadlines. Table 1 summarizes how the present study relates to the standards and implementation work above.

We ask whether preparing the network and endpoints in parallel, then activating them at a common gate epoch, improves deadline-constrained recovery. CSCR groups the candidate path, GCL, endpoint restoration and queue policy into one recovery transaction. We compare it with reactive restoration and preconfigured policies, including hot replication that retains continuous service during isolated link loss.

Figure 1 outlines the study structure. We formulate a recovery budget and define a common-epoch reconstruction procedure. A reproducible simulator records message outcomes and queue trajectories for each policy. A matched-input factorial comparison isolates preparation concurrency and gate-phase alignment. Per-flow recovery and all-flow confirmation metrics complement the first timely completion, while additional queue simulations assess preparation severity, control latency and background load.

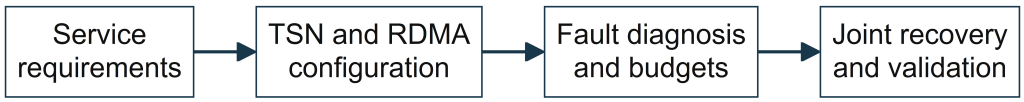


Figure 1. Study structure linking service requirements, integrated transport, fault assessment and collaborative reconstruction.

Table 1. Position of the study within related work.

Work	Main Contribution	Relation to This Study
IEEE/SAE aerospace profile [10]	Onboard TSN configuration context	Deployment scope
Fiori et al. [33]	Launcher traffic and lite TSN scheduling	Nominal deterministic operation
Chahed and Kassler [17]	Scheduling challenges and approaches	GCL planning context
Thiele et al. [25]	Formal timing analysis of TSN shapers	Candidate timing feasibility
IEEE 802.1CB [3]	Replication and elimination semantics	Hot-replication baseline
Kalia et al.; Hoefler et al. [60,66]	RDMA operation and deployment	Endpoint and workload constraints
This study	Common-epoch service reconstruction	Paired policy and ablation evaluation

Fault recovery must be distinguished from nominal schedule synthesis. Feng et al. [71] reroute and reschedule disrupted time-triggered flows while retaining existing routes and schedules for unaffected traffic. Gärtner et al. [72,73] address incremental runtime reconfiguration, while Zhao et al. [74] accelerate schedulability reverification after configuration changes. These studies address route/schedule construction or verification. CSCR assumes that a recovery configuration is available and studies the timing of network preparation, endpoint readiness and activation. It therefore complements these approaches; the present experiment does not establish a faster route-search or schedulability-verification algorithm.

Retransmission-based fault-tolerant scheduling, exemplified by ReT-FTS [75], explicitly accounts for another means of recovering faulty transmissions. The avionics TSN reconfiguration challenge [76] also makes runtime restoration of disrupted schedules a concrete engineering problem. Our single-bottleneck fluid experiment excludes frame retransmissions and multi-hop schedule synthesis, and its millisecond recovery metrics should not be compared directly with solver runtimes or end-to-end implementation measurements from these different settings.

RDMA recovery additionally requires consistency of transport and application state. Active-memory replication [77] uses an RDMA-based undo-logging protocol to preserve availability and correctness across failures; one-sided synchronization research [78] examines the correctness and performance consequences of synchronization choices. NVIDIA documents QP error transitions and resource recovery requirements [79]. These sources motivate the explicit endpoint-scope distinctions discussed below. They do not provide calibration for the aggregate endpoint-restoration durations assumed in our queue model.

The methodological contribution evaluated here is a matched comparison of cross-layer preparation concurrency and gate-phase alignment. Deadlines define service contracts and evaluation criteria. The executable policy does not select candidates or reject configurations

using online deadline predictions; accordingly, the results establish deadline-constrained policy behavior rather than a new deadline-aware scheduling algorithm.

2. Materials and Methods

2.1. System Scope and Service Contracts

The architecture has two forwarding planes with separate internal link and switch domains but shared source and destination endpoints. Redundant paths protect *a* plane-local link failure; an endpoint reset affects both planes. The numerical model reduces the active surviving plane to one gated bottleneck with 9.5 Gb/s effective payload capacity, abstracting multi-hop routing and physical frame overhead.

Each critical flow has *a* payload, release period and relative deadline. Its nominal delay includes scheduled network service, propagation and endpoint completion. Equation (1) defines recovery slack as the difference between the deadline and this nominal delay. The 12-ms threshold is *a* separate first-completion recovery budget used in the stress simulations; it is not *a* deadline assigned to every flow.

$$m_i = d_i - D_i^0. \tag{1}$$

A conservative post-fault delay budget adds the interruption and any increase in path, queueing or synchronization delay. Equation (2) gives the deadline test. The interruption H covers detection, preparation and epoch waiting, with parallel preparation contributing only its critical-path duration.

$$D_i^F \;\le\; D_i^0 + H + \Delta_{path,i} + \Delta_{queue,i} + \Delta_{sync,i} \;\le\; d_i. \tag{2}$$

Path diversity is assessed over internal forwarding nodes, links and specified common resources. Shared application endpoints are recorded separately and excluded from the internal node sets. Equation (3) states the domain-separation condition for plane-local protection; shared clock and endpoint faults require additional recovery actions.

The common-resource sets in Equation (3) refer only to resources claimed to be independent. A shared synchronization source is excluded from plane-local protection and remains *a* common failure domain. If it is included in both common-resource sets, the zero-intersection condition is not satisfied.

$$J_{int}(p_1, p_2) = |V_1^{int} \cap V_2^{int}| + |E_1 \cap E_2| + |C_1 \cap C_2| = 0. \tag{3}$$

2.2. Functional Faults and State Assessment

We consider four individual functional faults: clock drift, burst bit errors, link interruption and node reset. Figure 2 maps these faults to losses of timing or configuration. Each run injects one fault and samples its detection and restoration inputs. Service remains suspended until the recovery epoch, after which delivery is assumed to succeed. Table 2 lists assumed timing inputs. Simultaneous faults, correlated fault arrivals and persistent faults after activation are outside the evaluated coverage. A shared-endpoint reset is *a* single modeled event affecting both planes; it is not *a* correlated-failure experiment.

Figure 3 shows the proposed diagnostic interface, combining platform, synchronization, port and endpoint observations. The numerical study samples detection latency directly. Alarm fusion, EWMA/CUSUM parameter estimation and device diagnostics would need implementation measurements before they could replace this input.

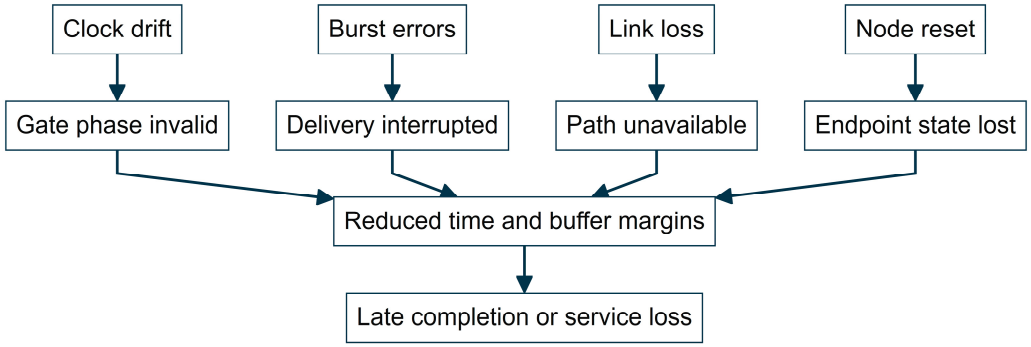


Figure 2. Functional faults and their effects on deterministic-service margins. The fault labels identify modeled consequences; radiation event rates are excluded.

Table 2. Functional fault inputs. All intervals are assumed model parameters.

Fault	Detection Range (ms)	Nominal Endpoint Restoration (ms)	Service Abstraction
Clock drift	0.35–0.65	2.0	Suspend valid service until timing restoration
Burst bit errors	0.20–0.45	0.7	Suspend delivery during error recovery
Link interruption	0.08–0.18	0.3	Restore service on the surviving plane
Node reset	0.40–0.80	4.0	Restore shared endpoint state

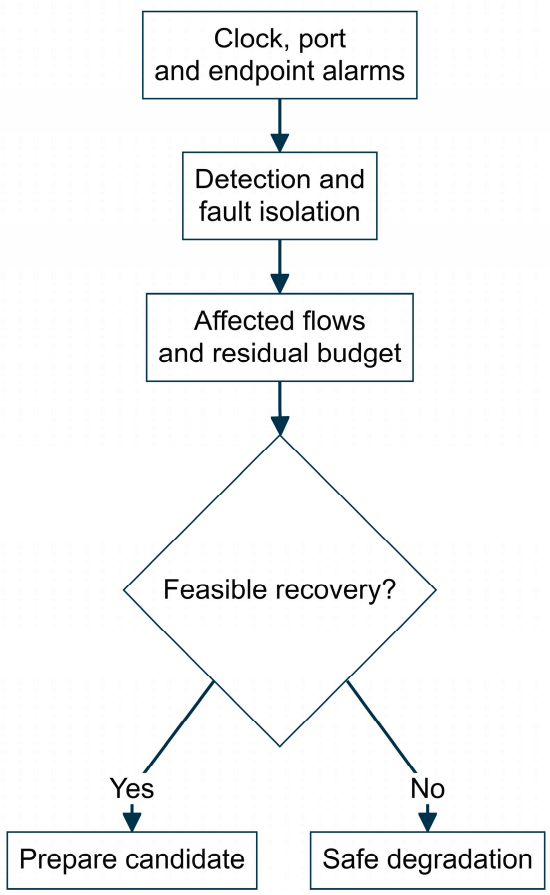


Figure 3. Proposed alarm-to-recovery decision flow. Detection latency is an input to the evaluated model.

Figure 4 shows the intended controller states: Healthy, Degraded, Fault and Recovery validation. Table 3 summarizes their guards and actions. A first timely completion is an early

service indicator only. Healthy requires all critical flows to satisfy *a* finite verification window, with all required completions observed. A validation timeout leads to safe degradation or rollback. The queue simulator evaluates these service metrics retrospectively; it does not execute *a* device-level validation or rollback controller.

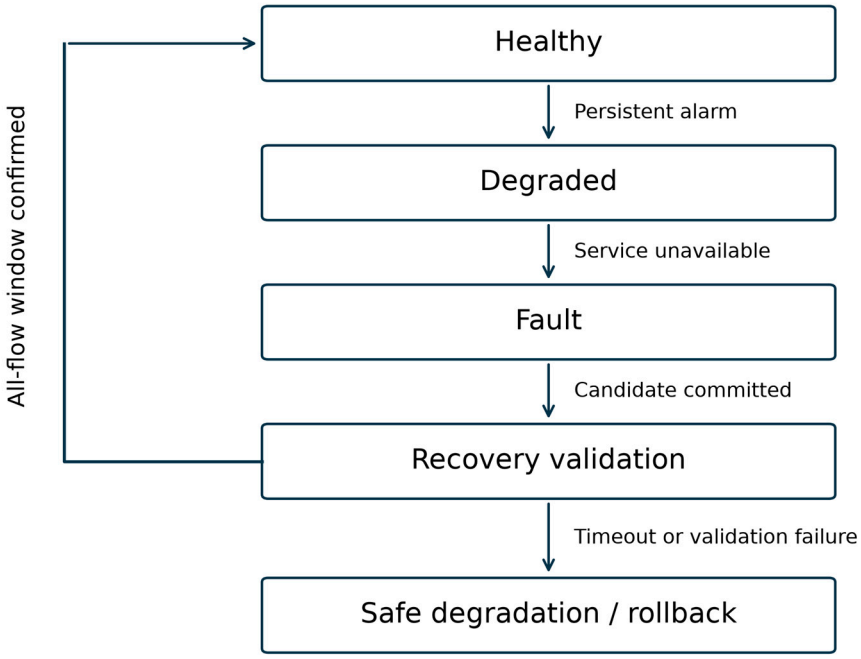


Figure 4. Intended controller transitions. Healthy requires observed all-flow window confirmation; timeout or failure triggers safe degradation. Device-level validation and rollback are not executed by the queue simulator.

Table 3. State guards and intended controller actions.

State	Guard	Action
Healthy	Configuration valid and service margins positive	Maintain active schedule
Degraded	Persistent anomaly or decreasing margin	Assess affected flows and candidate budgets
Fault	Valid service unavailable	Isolate the domain and prepare restoration
Recovery validation	All critical flows verified over the selected window	Enter Healthy on confirmation; timeout or failure leads to safe degradation or rollback

2.3. Queue and Deadline Feasibility

An arrival envelope and an assumed rate–latency service curve define *a* conditional design test for each traffic class [24,25]. Equation (4) is used only when the selected C and latency give *a* lower service bound for the candidate configuration. For *a* periodic fluid gate with open-gate rate R, open duration w and cycle T, $C = \mathrm{Rw/T}$ and *a* gate latency of at least $T-w$ provide *a* conservative rate–latency envelope; interruption H is then added. Initial backlog enters the arrival burst term. This conditional envelope is separate from the event simulator and does not establish multi-hop schedulability.

$$\alpha(t) = b + rt, \quad \beta(t) = C[t - T_0 - H]^+. \tag{4}$$

When the assumed service envelope is valid and C exceeds arrival rate r, vertical and horizontal deviations yield the conditional backlog and virtual-delay expressions in Equation (5). Excess backlog drains on the separate time scale $b/(C-r)$. These expressions describe candidate design budgets; the numerical policy does not enforce an online feasibility or admission test. The service-curve interpretation follows network calculus [80]; *a* valid arrival envelope and lower service curve are prerequisites for applying the bounds.

For the numerical critical class, the mean reserved rate is $0.2 \times 9.5 = 1.9$ Gb/s, whereas open-gate service uses 9.5 Gb/s. A rate–latency lower bound must include a latency covering the gate closure. The average reserved rate must not be substituted for the instantaneous open-gate rate in the event simulator. Equations (2), (4) and (5) are design envelopes, not an independently verified multi-hop schedulability proof. H ends at activation; the queue increment denotes additional postactivation waiting relative to the nominal service reference, excluding interruption already counted in H.

$$B_{max} \le b + r(T_0 + H), \quad D_{max} \le T_0 + H + \frac{b}{C}, \quad C > r. \tag{5}$$

The experiment keeps the admitted flow set fixed, allowing deadline failures to reveal policy differences. Queue occupancy is measured from remaining byte demand. The analytical guard describes *a* design envelope; the simulator does not synthesize multi-hop candidates or reject flows online.

2.4. Collaborative Reconstruction and Common Activation

CSCR prepares the surviving path and GCL while restoring endpoint state. Each candidate links compatible network, endpoint and queue-policy versions, as shown in Figure 5. If feasibility or version checks fail, deployment must provide rollback or safe degradation.

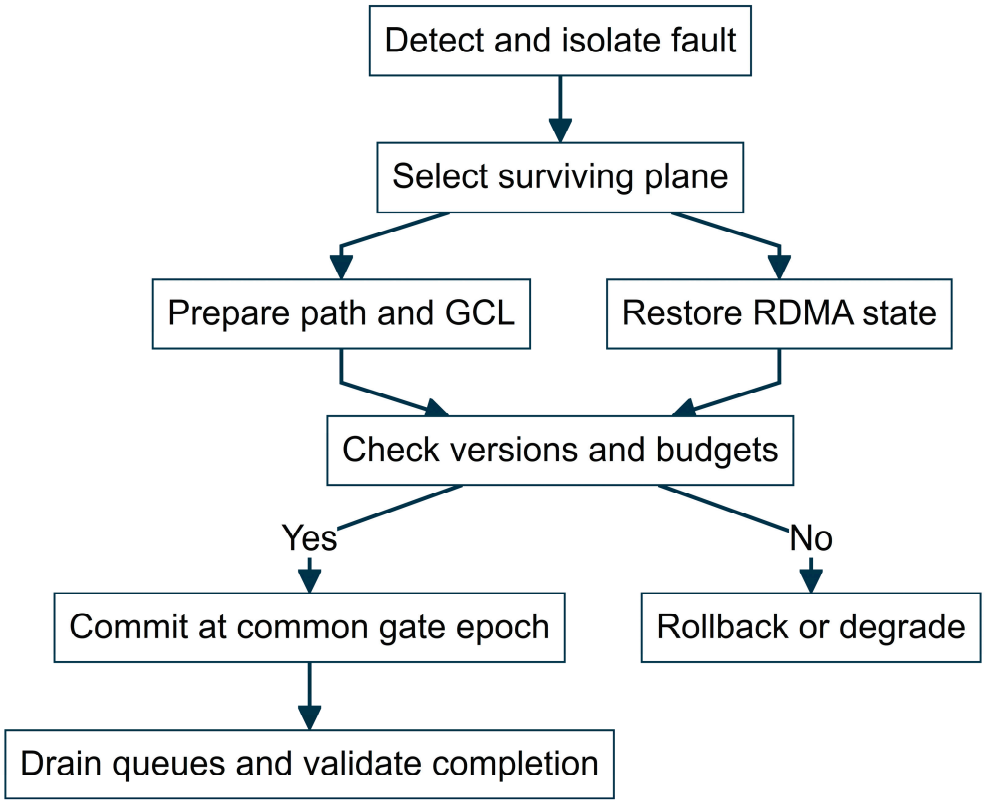


Figure 5. CSCR reconstruction procedure with parallel preparation and *a* common activation decision.

Preparation starts after detection, isolation and the controller decision. Absolute network and endpoint ready times are denoted by t_n and t_e. Equation (6) adds guard g and validation-readiness interval τ_v, then rounds readiness up to the next cycle boundary to obtain the activation epoch.

$$t_{commit} = T_c \left\lceil \frac{max(t_n,\ t_e) + g + \tau_v}{T_c} \right\rceil. \tag{6}$$

Both CSCR preparations start when the decision finishes, making readiness depend on the longer preparation duration. Serial policies restore endpoints only after network preparation. Figure 6 shows the intended exchange. Service resumes at one commit boundary, and recovery is measured at the first subsequent timely completion.

Queues retain FIFO order within each traffic class, and background data contributes to total occupancy throughout recovery. CSCR aligns the backup critical gate with the original cycle phase. The fluid service model does not implement queue shaping, sequence-register behavior or hardware atomicity.

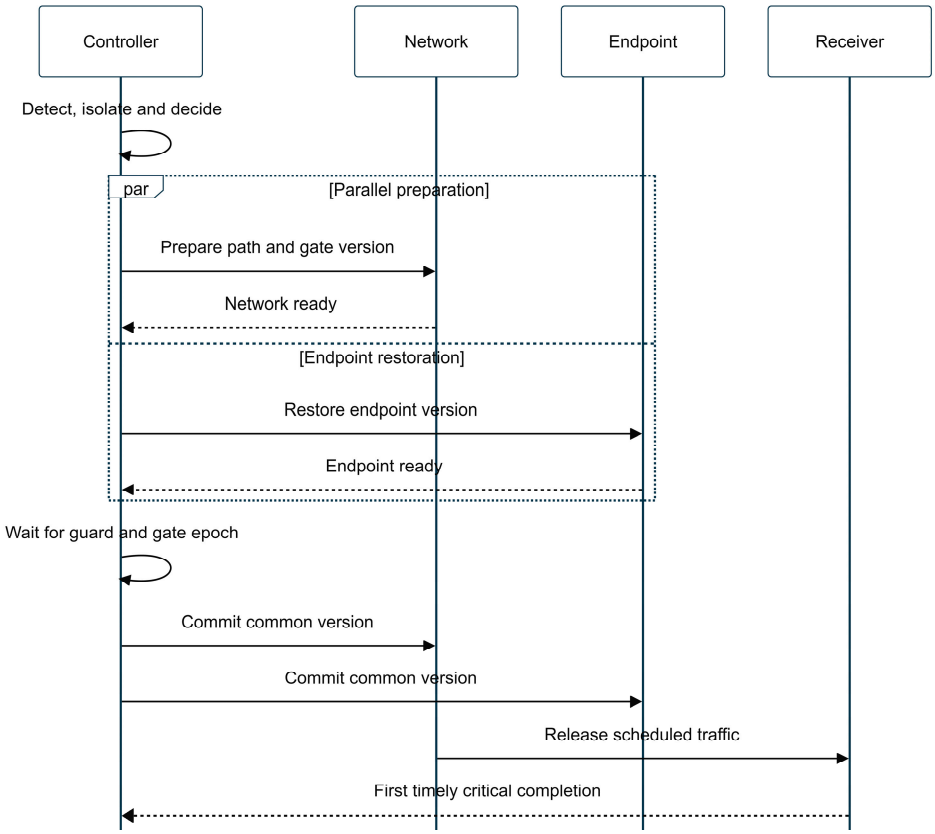


Figure 6. Common-epoch transaction. Network and endpoint preparation overlap; traffic resumes after common activation.

2.5. Workload and Discrete-Event Service Model

The simulator represents 44 periodic flows: 28 critical and 16 background. Each critical flow releases 8192 bytes every 2 ms. Its deadline is one of 4, 6, 8 or 12 ms, assigned cyclically by flow index. Background flows release every 1 ms with *a* 100-ms deadline. Their 55,917.9375-byte demand is derived from *a* total normalized load of 0.85. Fractional byte demand is retained in the fluid model.

The 1-ms gate cycle reserves its first 0.2 ms for critical traffic and the remaining 0.8 ms for background traffic. Service is FIFO within each class and stops when that class gate closes.

Partially served messages resume in the next open interval. The model uses effective payload service and excludes frame segmentation, link headers and non-preemptive frame blocking.

For each seed, flow phases are uniform within their periods, with release jitter uniform within ±0.03 ms. Policies and faults share the same workload. Source releases stop before 60 ms, but service continues to 120 ms. Endpoint completion and propagation add 0.05 and 0.01 ms, respectively. Table 4 summarizes the common parameters.

Table 4. Common numerical parameters.

Parameter	Setting
Effective payload capacity	9.5 Gb/s = 1,187,500 bytes/ms
Forwarding architecture	Two internally disjoint planes; shared endpoints
Critical flows	28; 8192 bytes every 2 ms
Critical deadlines	4, 6, 8 and 12 ms, assigned cyclically
Background flows	16; 55,917.9375 bytes every 1 ms
Normalized total load	0.85
Gate cycle/critical open time	1.0/0.2 ms
Endpoint/propagation latency	0.05/0.01 ms
Release jitter	Uniform ±0.03 ms
Fault onset/evaluation window	20 ms/releases in [20,45) ms
Release end/simulation end	60/120 ms
Paired seeds	0–59
Guard/validation-readiness interval	0.02/0.20 ms
Timing multipliers	Independent uniform [0.8, 1.2] for configuration and endpoint state
Sensitivity recovery budget	12 ms; separate from per-flow deadlines

Fault detection and isolation intervals are sampled uniformly; configuration and endpoint durations use independent uniform multipliers from 0.8 to 1.2, paired across methods. The model samples functional service suspension rather than *a* physical bit-error process or *a* clock waveform. Error-burst duration and synchronization-offset magnitude are therefore not modeled parameters and cannot be inferred from the fault labels.

The simulator advances between releases, gate transitions, fault onset, commit and service completions, choosing the next boundary or queue-head completion. During an open gate, service reduces remaining byte demand at the specified capacity. Queue peaks include both classes and are recorded before service at each boundary. All released messages must complete; pending messages at truncation trigger an execution error.

2.6. Baselines, Paired Inputs and Ablations

SR uses static standby resources with serial restoration, whereas ROR prepares routes reactively before recovering endpoints. FRER-Hot assumes an ideal surviving hot copy during isolated link loss. Pre-ROR shares CSCR's offline preparation opportunity but restores serially. FRER-Hot still requires platform restoration after clock, error and endpoint faults. Table 5 gives the policy inputs.

Table 5. Policy inputs. Configuration vectors follow the fault order in Table 2.

Policy	Mode	Nominal Configuration (ms)	Decision Range (ms)	Backup Gate Offset (ms)
SR	Serial	4.0, 2.0, 6.0, 3.0	0.20–0.40	0.35
ROR	Serial	3.0, 0.8, 4.0, 2.5	0.35–0.55	0.35
CSCR	Parallel	0.6, 0.4, 0.7, 0.8	0.10–0.20	0.00
FRER-Hot	Serial [1]	0.6, 0.4, 0.0, 0.8	0.10–0.20 [1]	0.00
Pre-ROR	Serial	0.6, 0.4, 0.7, 0.8	0.10–0.20	0.35
CSCR-Serial	Serial	0.6, 0.4, 0.7, 0.8	0.10–0.20	0.00
CSCR-Unaligned	Parallel	0.6, 0.4, 0.7, 0.8	0.10–0.20	0.35

[1] For FRER-Hot under link interruption, detection and all restoration durations are set to zero; service continues on the surviving copy. For all remaining cases, isolation is uniform over 0.10–0.25 ms. Fault-specific endpoint durations in Table 2 and policy configuration durations in Table 5 are multiplied by their paired timing draws.

NumPy SeedSequence uses [20261002, seed, 0] for workloads and [20261002, seed, fault index + 1] for fault timing. Methods share detection and isolation values, configuration and state multipliers, and the decision quantile. Each policy maps that quantile to its own decision interval, preserving paired workload and fault realizations.

CSCR, CSCR-Serial, CSCR-Unaligned and Pre-ROR form *a* matched-input 2 × 2 comparison of parallel versus serial preparation and aligned versus unaligned backup gates. Detection, isolation, configuration durations, endpoint durations, decision intervals and random draws are identical within this comparison. The original SR and ROR comparisons retain their distinct configuration and decision assumptions and therefore estimate complete-policy effects. The baseline dataset contains 1680 distinct runs from 60 seeds, four faults and seven policies, with 240 runs per policy.

2.7. Metrics, Inference and Numerical Verification

First-completion recovery time begins at fault onset and ends at the first timely critical completion released after the fault and completed after commit. It includes detection and residual gate waiting but does not establish recovery of every flow. For each critical flow, we additionally record its first such timely completion. An all-flow confirmation metric requires every critical flow to release at least one message and every critical message released within *a* window to meet its own deadline. Candidate window starts are the commit time and subsequent critical release times; the earliest qualifying window is selected. Confirmation occurs at the later of its window end and its last message completion. We use *a* 10-ms window, with 5- and 20-ms checks, and report elapsed time from fault onset. Windows must end by the 60-ms release horizon. Runs without *a* qualifying window are recorded as unconfirmed, not zero. Prefault releases are excluded from these recovery definitions. Mean violation percentage retains critical releases in [20,45) ms and Equation (7).

$$V = \frac{100}{N_{crit}} \sum_{i,j} 1\{\ell_{i,j} > d_i\}, \quad \ell_{i,j} = c_{i,j} - a_{i,j}. \tag{7}$$

Peak queued data is the largest remaining critical and background byte demand during [20,45) ms, expressed in KiB. P95 uses linear interpolation across runs. The conditional timely service index gives equal weight to each critical flow's timely completion fraction. It describes the injected-fault window; mission availability also depends on fault rates, mission duration and dependencies.

Paired effects are calculated as CSCR minus each comparator for matching seed and fault. Percentile confidence intervals use 4000 bootstrap draws of 60 seed blocks, retaining

all four faults per block. The bootstrap seed is 20261002. These intervals concern paired mean differences. All reported P95 values use NumPy's linear quantile definition.

Absolute P95 uncertainty and CSCR-minus-comparator P95 differences are additionally estimated with 4000 percentile bootstrap draws. Each draw resamples 60 seed blocks and retains all four faults and all methods within *a* block. Fault-specific intervals resample the 60 seeds within that fault. Both estimates use linear quantiles. These intervals describe uncertainty under the sampled synthetic scenarios; the finite fault mixture and assumed timing inputs remain fixed.

The review package provides parameters, simulation source, message outcomes, flow and run statistics, and verification commands. Boundary tests cover open-gate completion, closed-gate waiting, service across cycles and fault interruption. The full audit recalculates deadlines, checks record counts, reconstructs policy timing and reruns every seed. The released environment uses Python 3.12.14, NumPy 2.3.5 and pandas 3.0.1.

2.8. Queue-Based Stress Evaluation

We simulate queue service across preparation-severity multipliers of 0.6, 1.0, 1.4 and 1.8, additional controller delays of 0, 1, 2 and 4 ms, and normalized total loads of 0.70, 0.85 and 0.95. Equation (8) scales network-configuration duration and endpoint-restoration duration by κ and adds δ to controller-decision duration, before preparation starts. Workloads are regenerated at each load; policies share the corresponding seed and fault draws. Each grid cell includes SR, ROR, CSCR and Pre-ROR, four faults and 60 seeds, giving 46,080 runs. The subscripts cfg, end and dec identify configuration, endpoint and decision durations; primes denote the changed inputs. The stress figures and tables are calculated from message-level queue simulations.

$$\tau_{\mathrm{cfg}}{}' = \kappa\tau_{\mathrm{cfg}}, \quad \tau_{\mathrm{end}}{}' = \kappa\tau_{\mathrm{end}}, \quad \tau_{\mathrm{dec}}{}' = \tau_{\mathrm{dec}} + \delta. \tag{8}$$

Deadline violations are recalculated from simulated message completion times using Equation (9), where c and a denote completion and release times and d is the flow deadline. Load changes adjust background payloads while retaining critical traffic and gate reservations. Background-load changes therefore need not change critical-message outcomes in this class-isolated service model. At load 0.95, background arrivals exceed their reserved service share under sustained input; this is *a* finite-duration overload experiment, with releases stopping at 60 ms, rather than *a* stable operating point.

$$V(\kappa,\ \delta,\ \rho) = 100/N_{\mathrm{crit}} \sum_{i,j} 1\left\{c_{i,j}(\kappa,\ \delta,\ \rho) - a_{i,j} > d_i\right\}. \tag{9}$$

The separate recovery budget requires first-completion recovery within 12 ms. Equation (10) gives the empirical percentage for each severity–delay–load-policy cell, using $N = 240$ runs. Simulations run to 120 ms; pending released messages trigger an error. Section 3.5 reports the results; Supplementary Materials stress_summary.csv provides all cells. Here, N_{crit} counts critical releases in [20,45) ms; ρ denotes normalized total load.

$$F(\kappa,\ \delta,\ \rho) = 100/N \sum_{j} 1\left\{T_j(\kappa,\ \delta,\ \rho)\ \le\ 12\ \mathrm{ms}\right\}. \tag{10}$$

3. Results

3.1. Primary Paired Comparison

CSCR reduced P95 recovery time to 6.157 ms, compared with 9.513 ms for ROR and 9.541 ms for SR. Mean critical deadline violations were 0.875% for CSCR, 7.311% for ROR and 10.952% for SR. Table 6 and Figure 7 report first-completion recovery, critical deadline violations, peak queued data and conditional timely service across the common fault set.

Table 6. Primary paired results, 240 runs per method.

Method	P95 Recovery (ms)	Mean Violations (%)	P95 Queued Data (KiB)	Timely Service (%)
SR	9.541	10.952	9183.7	89.084
ROR	9.513	7.311	9083.9	92.717
CSCR	6.157	0.875	6146.1	99.128

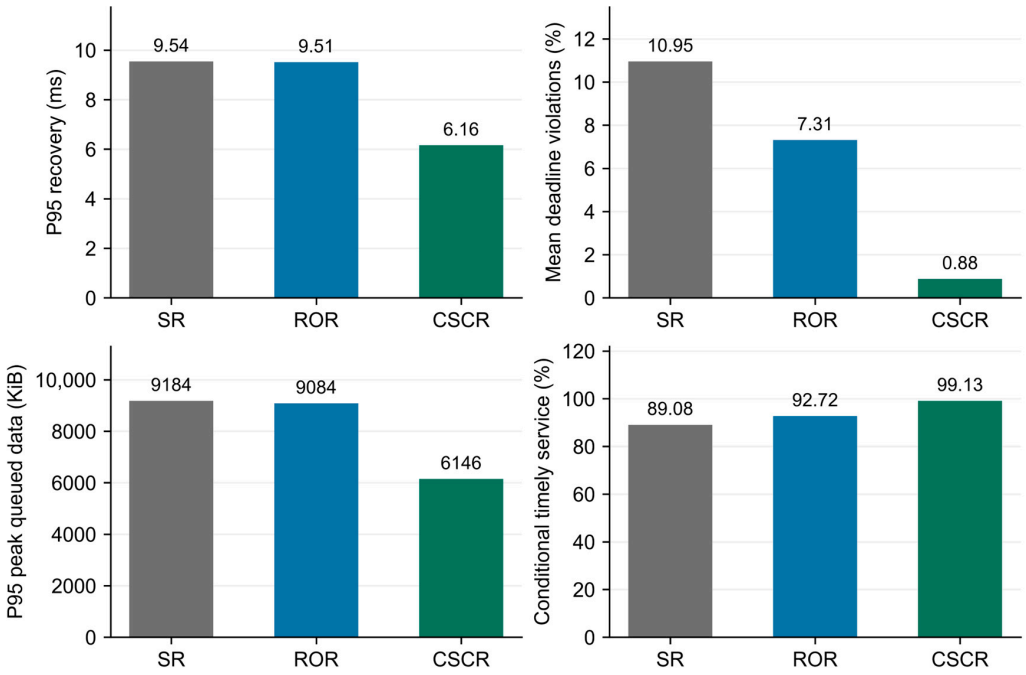


Figure 7. Primary results. Recovery and queued-data metrics are P95 values across runs; violations and conditional timely service are run means.

Compared with ROR, CSCR reduced P95 recovery time by 35.3% and P95 peak queued data by 32.3%. The paired mean recovery difference was −3.015 ms (95% seed-block bootstrap interval, −3.077 to −2.954 ms). The mean violation difference was −6.436 percentage points (95% interval, −6.703 to −6.169 points). These estimates used matched fault and workload samples.

Seed-block bootstrap intervals for first-completion recovery P95 are CSCR: 6.157 ms (95% interval 6.143–6.163 ms); ROR: 9.513 ms (95% interval 9.487–9.534 ms); SR: 9.541 ms (95% interval 9.514–9.555 ms). Full fault-specific and paired P95-difference estimates are provided in Supplementary Materials, v2_data/p95_intervals.csv.

Table 7 reports the primary P95 uncertainty directly; fault-specific and paired P95 differences are supplied in Supplementary Materials, v2_data/p95_intervals.csv.

Table 7. P95 estimates and 95% seed-block bootstrap intervals across 240 runs per policy. Recovery uses first timely critical completion; queued data includes both classes. Intervals use 4000 draws of 60 seed blocks.

Policy	Recovery P95 [95% Interval] (ms)	Queue P95 [95% Interval] (KiB)
SR	9.541 [9.514, 9.555]	9183.7 [9112.7, 9265.3]
ROR	9.513 [9.487, 9.534]	9083.9 [9018.6, 9150.1]
CSCR	6.157 [6.143, 6.163]	6146.1 [6093.4, 6188.3]
FRER-Hot	7.136 [7.115, 7.163]	7082.2 [7031.2, 7138.5]
Pre-ROR	7.486 [7.465, 7.513]	7082.2 [7031.2, 7138.5]

3.2. Preconfigured Comparison

FRER-Hot maintained service during isolated link loss, so the model imposed no scheduled interruption. The first timely postfault completion occurred after 0.174 ms on average. Across all four faults, its mean recovery time was 3.355 ms, slightly below CSCR's 3.451 ms. CSCR nevertheless had lower P95 recovery and fewer deadline violations (Table 8 and Figure 8).

Table 8. Preconfigured results, 240 paired runs per method.

Method	P95 Recovery (ms)	Mean Recovery (ms)	Mean Violations (%)	P95 Queued Data (KiB)	Timely Service (%)
FRER-Hot	7.136	3.355	1.503	7082.2	98.501
Pre-ROR	7.486	4.198	1.916	7082.2	98.096
CSCR	6.157	3.451	0.875	6146.1	99.128

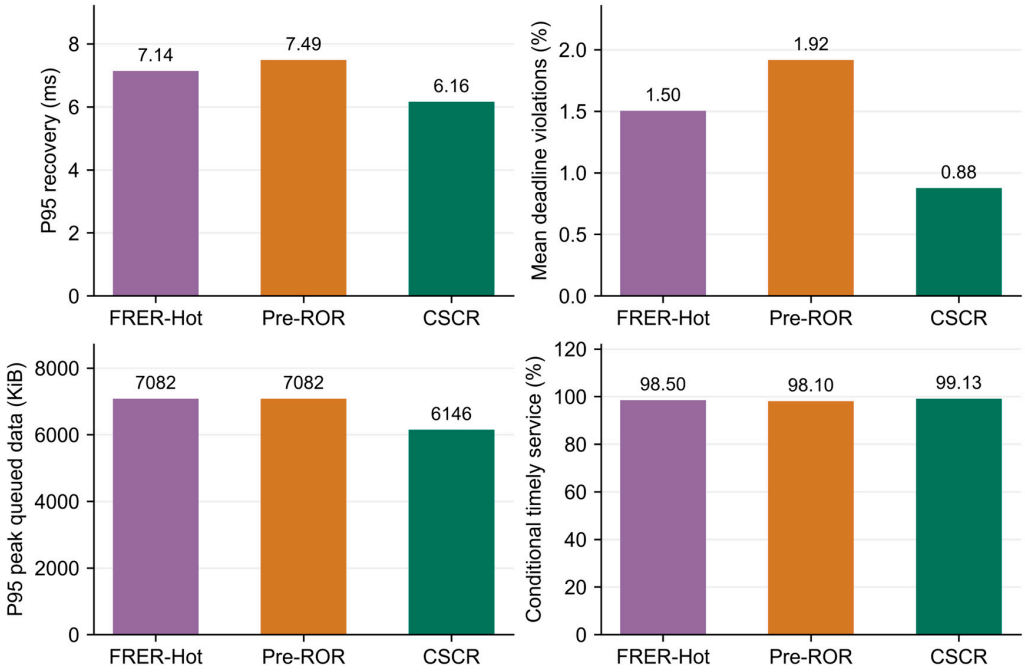


Figure 8. Preconfigured comparison under identical workload and fault draws. The shared CSCR runs also appear in the primary comparison.

CSCR's mean recovery time exceeded FRER-Hot's by +0.096 ms (95% interval, +0.041 to +0.154 ms). The paired mean violation difference was −0.628 percentage points (95% interval, −0.721 to −0.529 points). CSCR also had 70.3 KiB higher mean peak queue demand but *a* lower queue P95. The ranking thus depended on whether the comparison emphasized average or tail performance.

3.3. Fault-Specific Behavior and Queue Evolution

Most CSCR deadline violations occurred after node reset, which produced *a* mean violation percentage of 3.406%. Clock drift produced 0.095%, while burst errors and link interruption produced none under the tested parameters (Figure 9). Figure 10 shows the preparation schedule for *a* representative node-reset run. Node reset had the longest endpoint-state restoration interval in the model.

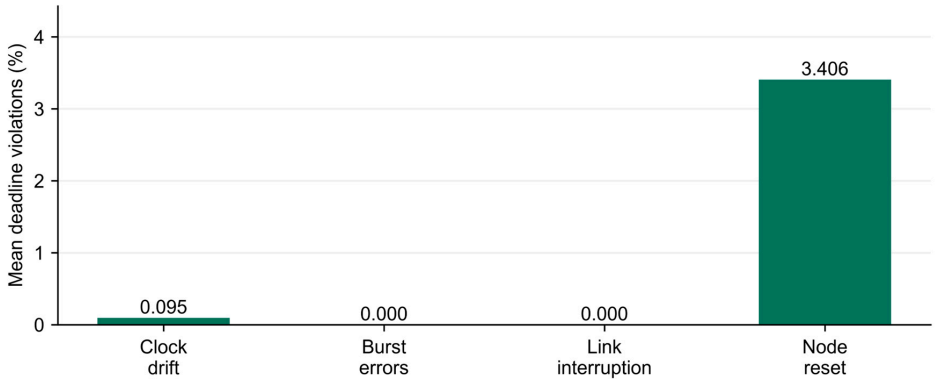


Figure 9. Fault-specific CSCR mean deadline-violation percentages, with 60 paired seeds per fault.

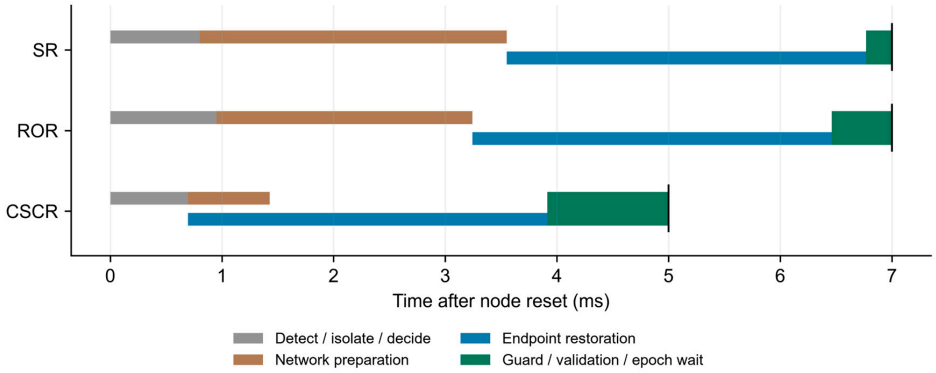


Figure 10. Recorded controller schedule for seed 0 under node reset. Thin vertical markers denote the commit epoch; the upper and lower bars distinguish network and endpoint preparation.

The seed-0 node-reset schedule in Figure 10 shows network and endpoint preparation overlapping in CSCR, while SR and ROR execute these stages serially. Service resumes after waiting for the common epoch. The recovery metric also includes the time to *a* subsequent timely message completion.

After link interruption, FRER-Hot continued serving traffic while the other policies accumulated backlog (Figure 11, seed 0). The curves include critical and background bytes and show gated drainage after commit. Variation within each curve comes from the simulated message releases.

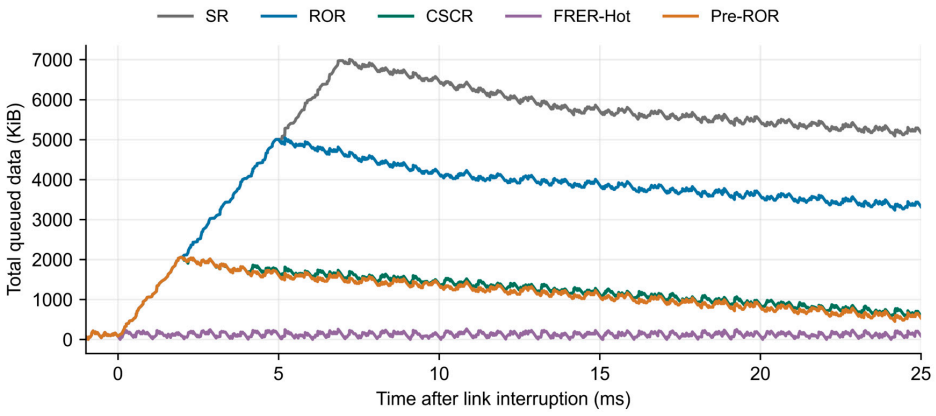


Figure 11. Total queued data around link interruption for seed 0, computed from remaining bytes at event boundaries.

3.4. Controlled Ablations

Serial preparation increased CSCR's P95 recovery time from 6.157 to 7.136 ms and mean violations from 0.875% to 1.503% (Table 9). With an unaligned backup gate, P95 recovery rose to 6.513 ms and mean violations to 1.186%. Queue P95 stayed at 6146.1 KiB because arrival accumulation during the interruption was unchanged.

Table 9. Controlled ablations with identical timing draws and configuration inputs.

Policy	P95 Recovery (ms)	Mean Violations (%)	P95 Queued Data (KiB)	Timely Service (%)
CSCR	6.157	0.875	6146.1	99.128
CSCR-Serial	7.136	1.503	7082.2	98.501
CSCR-Unaligned	6.513	1.186	6146.1	98.823
Pre-ROR	7.486	1.916	7082.2	98.096

With matched inputs, parallel preparation reduced mean first-completion recovery by 0.396 ms with aligned gates and 0.396 ms with unaligned gates. Phase alignment reduced the mean by 0.350 ms under both preparation modes. The mean difference-in-differences was 0.000230 ms for this parameter setting; these are mean effects, not an additive decomposition of P95 improvement against ROR. A further 8640 queue runs independently varied configuration and endpoint-restoration multipliers over 0.5, 1 and 2 for the same four policies. The Supplementary Files mechanism_runs.csv and mechanism_summary.csv provide all nine preparation scenarios. These exploratory ranges are not hardware-calibrated.

Table 10 quantifies paired mean mechanism effects and their uncertainty; it is not *a* decomposition of the P95 comparison with ROR.

Table 10. Paired mean recovery contrasts in the matched 2 × 2 experiment. Positive benefits indicate shorter recovery under the named mechanism. Interaction is the unaligned parallel benefit minus the aligned parallel benefit. Intervals use seed-block resampling.

Contrast	Mean (ms)	95% Interval (ms)
Parallel benefit, aligned	0.396236	[0.341983, 0.450346]
Parallel benefit, unaligned	0.396466	[0.342127, 0.450405]
Alignment benefit, parallel	0.350086	[0.350000, 0.350201]
Alignment benefit, serial	0.350316	[0.350057, 0.350661]
Interaction	0.000230	[−0.000029, 0.000575]

3.5. Queue-Based Recovery Stress Results

The queue-based stress evaluation comprises 46,080 simulation runs. Figure 12 shows CSCR mean first-completion recovery and critical deadline violations at load 0.85. Figure 13 compares the empirical 12-ms budget fractions of four policies. At severity 1.8, added controller delay 4 ms and load 0.95, CSCR mean recovery was 9.130 ms, mean critical violations were 18.479% and 77.08% of runs met the budget (see the selected stress results below). The corresponding critical outcomes are identical at loads 0.70 and 0.85 because only background payloads change and class gates are isolated. This equality is obtained from message-level queue simulations. It does not imply equal background backlog or steady-state feasibility. The Supplementary Files report every grid cell.

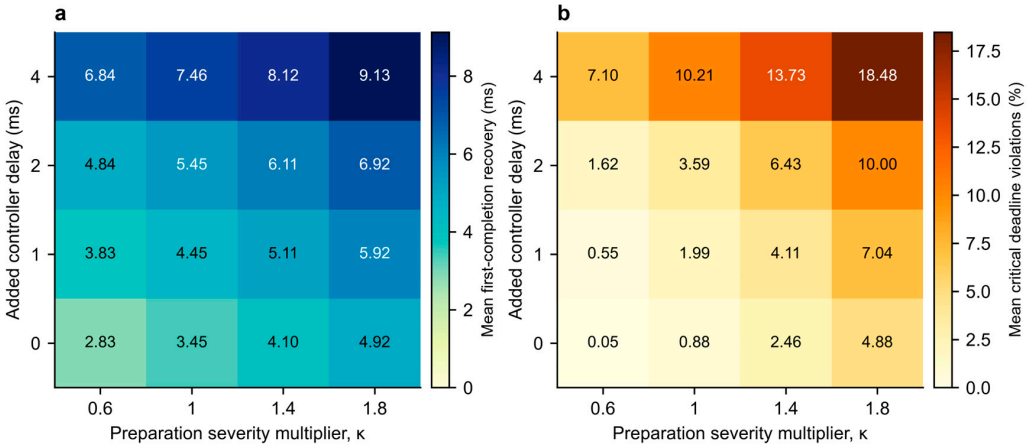


Figure 12. CSCR queue-based stress results at normalized total load 0.85: (**a**) mean first-completion recovery time and (**b**) mean critical deadline-violation percentage. Each cell summarizes 240 runs from 60 paired seeds and four individual faults. Severity scales configuration and endpoint-restoration durations; added controller latency precedes preparation. Values are calculated directly from message-level queue simulations without interpolation.

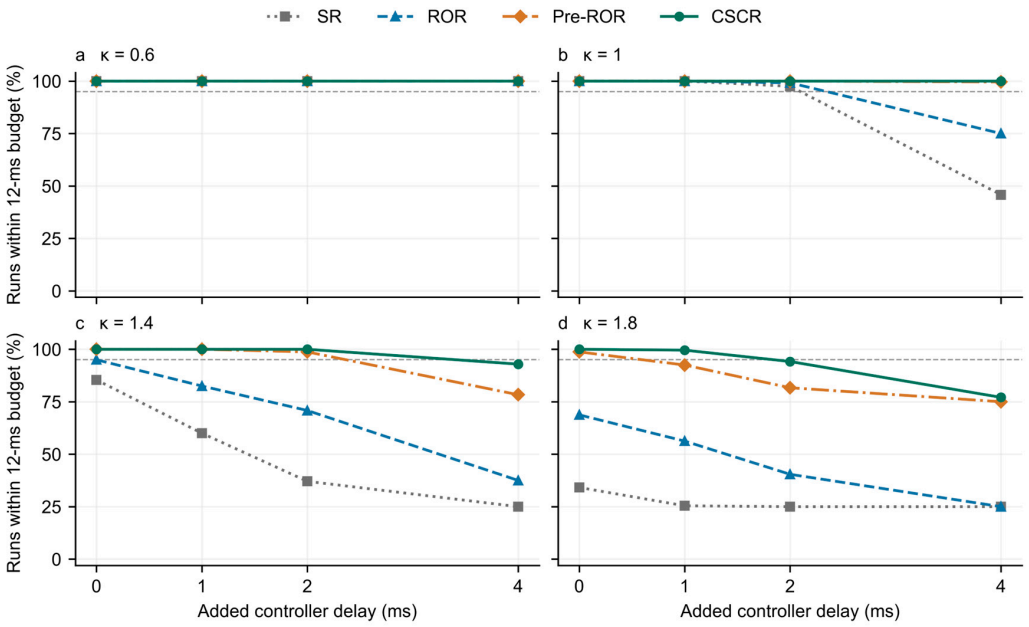


Figure 13. Percentage of queue runs meeting the separate 12-ms first-completion recovery budget, comparing SR, ROR, CSCR and Pre-ROR. Panels (**a**–**d**) show preparation-severity multipliers 0.6, 1.0, 1.4 and 1.8. Each point summarizes 240 runs from 60 paired seeds and four faults. The plotted curves apply to all three tested loads (0.70, 0.85 and 0.95), whose critical outcomes coincide under class-isolated gates. The dashed horizontal line is *a* 95% reference, not *a* confidence interval.

An additional 2880 matched queue runs vary the critical gate width over 0.12, 0.16 and 0.20 ms per 1-ms cycle, retaining 60 seeds, four faults and four mechanism-control policies. CSCR mean critical violation percentages are 3.110% at 0.12 ms, 1.278% at 0.16 ms, 0.875% at 0.20 ms. Unlike background-only load changes, this experiment changes critical-class service opportunities. All gate widths admit the nominal critical arrival rate; this remains a fluid-service sensitivity experiment.

Table 11 reports all four matched-input policies and uncertainty, rather than interpreting the background-load invariance as robustness to critical-class contention. Table 12 presents selected CSCR stress results across severity, controller delay and load.

Table 11. Critical-gate sensitivity under matched inputs, 240 runs per cell. Entries are mean per-run critical violation percentages with 95% seed-block bootstrap intervals.

Gate (ms)	Policy	Violations (%)	95% Interval (%)
0.12	CSCR	3.110	[2.823, 3.404]
0.12	CSCR-Serial	5.030	[4.701, 5.364]
0.12	CSCR-Unaligned	4.086	[3.701, 4.478]
0.12	Pre-ROR	6.368	[6.012, 6.726]
0.16	CSCR	1.278	[1.153, 1.413]
0.16	CSCR-Serial	2.128	[1.965, 2.292]
0.16	CSCR-Unaligned	1.734	[1.560, 1.913]
0.16	Pre-ROR	2.737	[2.556, 2.914]
0.20	CSCR	0.875	[0.787, 0.969]
0.20	CSCR-Serial	1.503	[1.374, 1.628]
0.20	CSCR-Unaligned	1.186	[1.062, 1.310]
0.20	Pre-ROR	1.916	[1.780, 2.049]

Table 12. Selected queue-based stress results for CSCR, with 240 runs per row. Recovery is the first-completion metric; the 12-ms budget is separate from per-flow deadlines.

κ	Load	Added Delay (ms)	Mean Recovery (ms)	Mean Violations (%)	Budget Feasible (%)
1.0	0.85	4	7.46	10.21	100.00
1.8	0.85	4	9.13	18.48	77.08
1.8	0.95	0	4.92	4.88	100.00
1.8	0.95	2	6.92	10.00	94.17
1.8	0.95	4	9.13	18.48	77.08

4. Discussion

4.1. Mechanisms and Baseline Interpretation

Three choices distinguish CSCR from the reactive baselines: offline configuration, overlapping preparation and gate alignment. Configuration and decision durations also differ in the original policy inputs. The 35.3% first-completion P95 reduction against ROR is therefore *a* complete-policy comparison. The four matched-input controls identify concurrency and phase effects under common timing assumptions; these effects should not be added as *a* decomposition of the original P95 difference.

When network and endpoint work can proceed independently, parallel preparation shortens the critical path to readiness. Aligning activation with the gate phase then reduces the wait for critical service. The unaligned ablation increased completion latency without changing the interruption or queue peak. Implementing this behavior requires independent preparation interfaces and bounded activation timing.

Hot replication is particularly effective for isolated link failure because the surviving copy can continue serving traffic. Shared clock or endpoint faults still require restoration. This explains why FRER-Hot had the lower overall mean recovery time, whereas CSCR had the lower tail across all four faults. A deployment could combine hot protection for tight link-fault budgets with coordinated restoration for shared service faults.

The conditional timely service index gives *a* compact view of per-flow completion during the injected-fault window. Equal flow weights simplify comparison, although mission modes may assign different priorities. Translating this index into task-level availability

would also require correlated-failure and shared-resource dependencies, together with mission-specific event rates.

4.2. Model Boundaries and Sensitivity Assumptions

The model represents two traffic classes at one equivalent surviving bottleneck, with fluid message service. It omits physical frame blocking, transport retries, duplication overhead and multi-hop gate synthesis. Endpoint restoration is represented by *a* duration and completion offset. These abstractions allow policy comparisons but limit direct predictions for *a* device.

Preparation durations are especially consequential: configuration and endpoint-restoration assumptions determine much of the recovery time. These values have not been calibrated against TSN switches or RDMA-capable endpoints. Matched-input controls and queue-based parameter scans characterize the modeled mechanism but do not establish device-specific absolute timings or improvement magnitudes. Tables 2 and 5 should be read as declared simulation scenarios rather than experimentally established timing ranges.

The stress evaluation simulates queue service after changing preparation durations, controller latency and background payloads. Deadline violations reflect the added control delay through simulated message completion times. Critical and background classes have separate reserved gates, so increasing background load does not introduce frame blocking or shared-class interference into critical service. The three tested loads yield identical critical outcomes in these runs; background queue demand remains load-dependent. The 0.95 case overloads the background reservation under sustained arrivals; drainage after releases stop does not demonstrate steady-state feasibility.

A first timely completion does not mean that every critical flow has recovered. The P95 time by which each critical flow had completed at least one timely postfault message was 9.122 ms for CSCR and 15.472 ms for ROR. Across 240 runs per policy, CSCR achieved *a* 16.143-ms P95 all-flow confirmation time for the 10-ms window, compared with 20.493 ms for ROR, 20.566 ms for SR, 17.452 ms for Pre-ROR and 17.129 ms for FRER-Hot. Every baseline run had *a* qualifying 5-, 10- and 20-ms window before the release horizon. These are retrospective confirmations over finite release windows, not guarantees that every later message will remain timely. Per-flow first completions and window results are included in the Supplementary Data.

For the 10-ms all-flow confirmation window, all 240 runs per policy were confirmed. P95 and 95% seed-block intervals are CSCR: 16.143 ms (16.136–16.150 ms); ROR: 20.493 ms (20.459–20.530 ms); SR: 20.566 ms (20.524–21.417 ms). The 5- and 20-ms window results and confirmation counts are supplied in Supplementary Materials, v2_data/allflow_intervals.csv.

Table 13 presents the confirmation counts and uncertainty; 5- and 20-ms windows remain available in the reproducible source data.

Table 13. Confirmation of all critical flows over *a* 10-ms release window. Time is measured from fault onset; intervals are seed-block bootstrap estimates.

Policy	Confirmed/Runs	P95 (ms)	95% Interval (ms)
SR	240/240	20.566	[20.524, 21.417]
ROR	240/240	20.493	[20.459, 20.530]
CSCR	240/240	16.143	[16.136, 16.150]
FRER-Hot	240/240	17.129	[17.102, 17.143]
Pre-ROR	240/240	17.452	[17.431, 17.465]

4.3. Hardware Calibration and Review Package

Hardware measurements are needed to establish performance on *a* specific switch and endpoint. This study provides no such measurements and makes no hardware-validation claim. The common commit epoch is *a* simulated service boundary, not evidence of hardware-atomic activation. QP transitions, packet sequence state, outstanding RDMA operations, retries, configuration-version checks and rollback are design requirements whose protocol-level correctness is not evaluated by the duration-based endpoint model. The results do not quantify correlated-fault tolerance or mission-level reliability.

Table 14 summarizes the scope of RDMA recovery state in the numerical model. QP lifecycle and packet sequence numbers are not simulated; deployment must establish compatible peer states before enabling traffic. Outstanding work requests and completion queues are not replayed or reconciled; exactly once completion is unverified. Memory registration, access keys and remote-access authorization are not modeled. Retransmission counters, acknowledgments and retry timers are excluded. Configuration versions, transaction commit acknowledgments and rollback are required implementation mechanisms, not simulator outputs. The endpoint restoration duration aggregates these unspecified activities and provides no protocol-correctness evidence.

Table 14. Scope of endpoint restoration in the numerical model.

State or Mechanism	Numerical Treatment
QP lifecycle/packet sequence	Not simulated; compatibility assumed after restoration
Outstanding WR/completion queue	No replay or reconciliation; exactly once unverified
Memory registration/access rights	Not modeled
Transport retries/acknowledgments	Not modeled
Configuration versions/atomic commit	Common service epoch only; protocol consistency unverified
Validation/rollback	Retrospective service confirmation; no device rollback
Restoration duration	Assumed aggregate elapsed time, not protocol execution

The source package includes simulation inputs, message outcomes, per-flow and per-run statistics, seed-0 queue trajectories and paired estimates. Reviewers can regenerate the datasets and recalculate the tables from the recorded assumptions. The executable package covers the numerical study; document and graphics production are separate.

5. Conclusions

CSCR coordinates network and endpoint restoration through parallel preparation and *a* common gate epoch. Under the declared timing assumptions, paired simulation produced a 6.157-ms P95 first-completion recovery time and 0.875% mean critical deadline violations. The more demanding 10-ms all-flow confirmation metric had a P95 of 16.143 ms. Matched-input controls support the preparation and phase mechanisms, while queue-based stress simulations describe their dependence on modeled preparation and controller delays. Hot replication retains its advantage for isolated link interruption and some mean metrics. These findings apply to individual functional faults in the specified gated fluid-service model; hardware recovery, RDMA protocol consistency and spacecraft deployment performance remain unvalidated.

Supplementary Materials: The following supporting information can be downloaded at https://doi.org/10.57760/sciencedb.012a6, CSCR_revised_code_and_data_v3.zip contains numerical inputs, simulation source, message outcomes, per-flow and per-run recovery statistics, queue-based stress results, statistical analyses and figure-generation instructions.

Author Contributions: Conceptualization, W.Z. (Wenjie Zhao) and J.R.; methodology, W.Z. (Wenjie Zhao); software, W.Z. (Wenjie Zhao); formal analysis, W.Z. (Wenjie Zhao); investigation and validation, W.Z. (Wenjie Zhao), W.Z. (Wanying Zhang) and E.L.; resources, G.Y. and C.J.; data curation and visualization, W.Z. (Wanying Zhang); writing—original draft preparation, W.Z. (Wenjie Zhao); writing—review and editing, J.R., W.Z. (Wanying Zhang), E.L., G.Y., C.J. and J.A.; supervision and project administration, J.R. and J.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The numerical inputs, simulation source, message outcomes, recovery metrics, stress results, statistical analysis and figure-generation instructions are provided in CSCR_revised_code_and_data_v3.zip (https://doi.org/10.57760/sciencedb.012a6). The stress figures and tables use queue-simulation data.

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

Abbreviations

Abbreviation	Meaning
TSN	Time-sensitive networking
RDMA	Remote direct memory access
GCL	Gate-control list
CSCR	Coordinated scheduling and collaborative reconfiguration
SR	Static standby redundancy
ROR	Route-only recovery
FRER	Frame replication and elimination for reliability

References

1. *IEEE Std 802.1AS-2020*; Timing and Synchronization for Time-Sensitive Applications. IEEE: New York, NY, USA, 2020. Available online: https://standards.ieee.org/ieee/802.1AS/7121/ (accessed on 2 October 2026).
2. *IEEE Std 802.1Q-2022*; IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks. IEEE: New York, NY, USA, 2022. Available online: https://standards.ieee.org/ieee/802.1Q/10323/ (accessed on 2 October 2026).
3. *IEEE Std 802.1CB-2017*; IEEE Standard for Local and Metropolitan Area Networks—Frame Replication and Elimination for Reliability. IEEE: New York, NY, USA, 2017. Available online: https://1.ieee802.org/tsn/802-1cb/ (accessed on 2 October 2026).
4. *IEEE Std 802.1Qbv-2015*; Enhancements for Scheduled Traffic. IEEE: New York, NY, USA, 2016. Available online: https://www.ieee802.org/1/pages/802.1bv.html (accessed on 2 October 2026).
5. *IEEE Std 802.1Qbu-2016*; Frame Preemption. IEEE: New York, NY, USA, 2016. Available online: https://www.ieee802.org/1/pages/802.1bu.html (accessed on 2 October 2026).
6. *IEEE Std 802.1Qci-2017*; Per-Stream Filtering and Policing. IEEE: New York, NY, USA, 2017. Available online: https://1.ieee802.org/tsn/802-1qci/ (accessed on 2 October 2026).
7. *IEEE Std 802.1Qch-2017*; Cyclic Queuing and Forwarding. IEEE: New York, NY, USA, 2017. Available online: https://1.ieee802.org/tsn/802-1qch/ (accessed on 2 October 2026).
8. *IEEE Std 802.1Qcr-2020*; Asynchronous Traffic Shaping. IEEE: New York, NY, USA, 2020. Available online: https://1.ieee802.org/tsn/802-1qcr/ (accessed on 2 October 2026).
9. *IEEE Std 802.1CM-2018*; Time-Sensitive Networking for Fronthaul. IEEE: New York, NY, USA, 2018. Available online: https://1.ieee802.org/tsn/802-1cm/ (accessed on 2 October 2026).
10. *IEEE Std 802.1DP-2025*; Time-Sensitive Networking Profile for Aerospace Onboard Ethernet Communications. IEEE: New York, NY, USA, 2025. Available online: https://1.ieee802.org/publication-ieee-802-1dp-2025/ (accessed on 2 October 2026).

11. IETF. *RFC 8655: Deterministic Networking Architecture*; RFC Editor: Fremont, CA, USA, 2019. Available online: https://www.rfc-editor.org/rfc/rfc8655 (accessed on 2 October 2026).
12. IETF. *RFC 8938: Deterministic Networking Data Plane Framework*; RFC Editor: Fremont, CA, USA, 2020. Available online: https://www.rfc-editor.org/rfc/rfc8938 (accessed on 2 October 2026).
13. Finn, N. Introduction to Time-Sensitive Networking. *IEEE Commun. Stand. Mag.* **2018**, *2*, 22–28. [CrossRef]
14. Nasrallah, A.; Thyagaturu, A.S.; Alharbi, Z.; Wang, C.; Shao, X.; Reisslein, M.; ElBakoury, H. Ultra-Low Latency (ULL) Networks: The IEEE TSN and IETF DetNet Standards and Related 5G ULL Research. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 88–145. [CrossRef]
15. Deng, L.; Xie, G.; Liu, H.; Han, Y.; Li, R.; Li, K. A Survey of Real-Time Ethernet Modeling and Design Methodologies: From AVB to TSN. *ACM Comput. Surv.* **2023**, *55*, 31. [CrossRef]
16. Zhang, T.; Wang, G.; Xue, C.; Wang, J.; Nixon, M.; Han, S. Time-Sensitive Networking (TSN) for Industrial Automation: Current Advances and Future Directions. *ACM Comput. Surv.* **2024**, *57*, 30. [CrossRef]
17. Chahed, H.; Kassler, A. TSN Network Scheduling—Challenges and Approaches. *Network* **2023**, *3*, 585–624. [CrossRef]
18. Wang, Z.; Luo, F.; Li, Y.; Gan, H.; Zhu, L. Schedulability analysis in time-sensitive networking: A systematic literature review. *Ad. Hoc Netw.* **2025**, *176*, 103897. [CrossRef]
19. Bruckner, D.; Stanica, M.P.; Blair, R.; Schriegel, S.; Kehrer, S.; Seewald, M.; Sauter, T. An Introduction to OPC UA TSN for Industrial Communication Systems. *Proc. IEEE* **2019**, *107*, 1121–1131. [CrossRef]
20. Craciunas, S.S.; Oliver, R.S.; Chmelík, M.; Steiner, W. Scheduling Real-Time Communication in IEEE 802.1Qbv Time Sensitive Networks. In *Proceedings of 24th International Conference on Real-Time Networks and Systems*; ACM: New York, NY, USA, 2016; pp. 183–192. [CrossRef]
21. Dürr, F.; Nayak, N.G. No-wait Packet Scheduling for IEEE Time-sensitive Networks (TSN). In *Proceedings of 24th International Conference on Real-Time Networks and Systems*; ACM: New York, NY, USA, 2016; pp. 203–212. [CrossRef]
22. Steiner, W. An Evaluation of SMT-Based Schedule Synthesis for Time-Triggered Multi-hop Networks. In *Proceedings of 2010 31st IEEE Real-Time Systems Symposium*; IEEE: New York, NY, USA, 2010; pp. 375–384. [CrossRef]
23. Nayak, N.G.; Dürr, F.; Rothermel, K. Time-sensitive Software-defined Network (TSSDN) for Real-time Applications. In *Proceedings of 24th International Conference on Real-Time Networks and Systems*; ACM: New York, NY, USA, 2016; pp. 193–202. [CrossRef]
24. Zhao, L.; Pop, P.; Li, Q.; Chen, J.; Xiong, H. Timing analysis of rate-constrained traffic in TTEthernet using network calculus. *Real-Time Syst.* **2017**, *53*, 254–287. [CrossRef]
25. Thiele, D.; Ernst, R.; Diemer, J. Formal worst-case timing analysis of Ethernet TSN's time-aware and peristaltic shapers. In *Proceedings of 2015 IEEE Vehicular Networking Conference (VNC)*; IEEE: New York, NY, USA, 2015; pp. 251–258. [CrossRef]
26. Kim, H.; Kim, Y.J.; Kim, W.T. Deep Reinforcement Learning-Based Adaptive Scheduling for Wireless Time-Sensitive Networking. *Sensors* **2024**, *24*, 5281. [CrossRef] [PubMed]
27. Jiang, J.; Li, Y.; Zhang, X.; Yu, M.; Lee, C.D.; Hong, S.H. Assessing the traffic scheduling method for time-sensitive networking (TSN) by practical implementation. *J. Ind. Inf. Integr.* **2023**, *33*, 100464. [CrossRef]
28. Bezerra, D.; de Oliveira Filho, A.T.; Rodrigues, I.R.; Dantas, M.; Barbosa, G.; Souza, R.; Kelner, J.; Sadok, D. A machine learning-based optimization for end-to-end latency in TSN networks. *Comput. Commun.* **2022**, *195*, 424–440. [CrossRef]
29. Pei, J.; Hu, Y.; Tian, L. A Review on Key Mechanisms of Time-Sensitive Networking. In *Proceedings of 2021 International Conference on Advanced Computing and Endogenous Security*; IEEE: New York, NY, USA, 2022; pp. 01–07. [CrossRef]
30. Zanbouri, K.; Noor-A-Rahim, M.; John, J.; Sreenan, C.J.; Vincent Poor, H.; Pesch, D. A Comprehensive Survey of Wireless Time-Sensitive Networking (TSN): Architecture, Technologies, Applications, and Open Issues. *IEEE Commun. Surv. Tutor.* **2025**, *27*, 2129–2155. [CrossRef]
31. Sanchez-Garrido, J.; Aparicio, B.; Ramirez, J.G.; Rodriguez, R.; Melara, M.; Cercos, L.; Ros, E.; Diaz, J. Implementation of a Time-Sensitive Networking (TSN) Ethernet Bus for Microlaunchers. *IEEE Trans. Aerosp. Electron. Syst.* **2021**, *57*, 2743–2758. [CrossRef]
32. Zhou, X.; He, F.; Zhao, L. Loosely Coupled Hybrid Scheduling of Processing and Communication for TSN-Based IMA Systems. *IEEE Trans. Ind. Inform.* **2024**, *20*, 8884–8895. [CrossRef]
33. Fiori, T.; Lavacca, F.G.; Valente, F.; Eramo, V. Proposal and Investigation of a Lite Time Sensitive Networking Solution for the Support of Real Time Services in Space Launcher Networks. *IEEE Access* **2024**, *12*, 10664–10680. [CrossRef]
34. Li, B.; Zhu, Y.; Liu, Q.; Yao, X. Development of Deterministic Communication for In-Vehicle Networks Based on Software-Defined Time-Sensitive Networking. *Machines* **2024**, *12*, 816. [CrossRef]
35. Xia, D.; Jiang, C.; Wan, J.; Jin, J.; Leung, V.C.M.; Martínez-García, M. Heterogeneous Network Access and Fusion in Smart Factory: A Survey. *ACM Comput. Surv.* **2023**, *55*, 113. [CrossRef]
36. ECSS. ECSS-E-ST-50-12C Rev.1: SpaceWire—Links, Nodes, Routers and Networks. Noordwijk: ESA-ESTEC. 2019. Available online: https://ecss.nl/standard/ecss-e-st-50-12c-rev-1-spacewire-links-nodes-routers-and-networks-15-may-2019/ (accessed on 2 October 2026).

37. ECSS. ECSS-E-ST-50-11C: SpaceFibre—Very High-Speed Serial Link. Noordwijk: ESA-ESTEC. 2019. Available online: https://ecss.nl/standard/ecss-e-st-50-11c-spacefibre-very-high-speed-serial-link/ (accessed on 2 October 2026).
38. Parkes, S.M.; Armbruster, P. SpaceWire: A spacecraft onboard network for real-time communications. In *Proceedings of 14th IEEE-NPSS Real Time Conference*; IEEE: New York, NY, USA, 2005; pp. 6–10. [CrossRef]
39. Parkes, S.; McClements, C.; McLaren, D.; Florit, A.F.; Gonzalez Villafranca, A. SpaceFibre: A multi-Gigabit/s interconnect for spacecraft onboard data handling. In *Proceedings of 2015 IEEE Aerospace Conference*; IEEE: New York, NY, USA, 2015; pp. 1–13. [CrossRef]
40. ECSS. ECSS-Q-ST-60-15C: Radiation Hardness Assurance—EEE Components. Noordwijk: ESA-ESTEC. 2012. Available online: https://ecss.nl/wp-content/uploads/standards/ecss-q/ECSS-Q-ST-60-15C1October2012.pdf (accessed on 2 October 2026).
41. Hihara, H.; Nomachi, M.; Takahashi, T. Contributing to the SpaceWire international standard. *Synth. Engl. Ed.* **2019**, *11*, 146–157. [CrossRef] [PubMed]
42. Senior, A.; Coetzee, J.P.; Ilstad, J. The draft ECSS SpaceWire Backplane standard. In *Proceedings of 2014 International SpaceWire Conference (SpaceWire)*; IEEE: New York, NY, USA, 2014; pp. 1–4. [CrossRef]
43. Zhou, Q.; Zhu, L.; Fei, H.; Wang, X. A reliability analysis tool for SpaceWire network. *AIP Conf. Proc.* **2017**, *1834*, 020008. [CrossRef]
44. Bravhar, K.; Kramberger, I.; Falcon, L.S.; Codinachs, D.M.; Gačnik, D. SerDes Integrated Into the SpaceWire Interface Helps in Achieving Higher Data Rates. *IEEE Aerosp. Electron. Syst. Mag.* **2023**, *38*, 16–27. [CrossRef]
45. Notebaert, O. On-board data links and networks in space applications. *IFAC Proc. Vol.* **2007**, *40*, 59. [CrossRef]
46. Scholz, A.; Hsiao, T.H.; Juang, J.N.; Cherciu, C. Open source implementation of ECSS CAN bus protocol for CubeSats. *Adv. Space Res.* **2018**, *62*, 3438–3448. [CrossRef]
47. Paulitsch, M.; Hall, B. FlexRay in aerospace and safety-sensitive systems. *IEEE Aerosp. Electron. Syst. Mag.* **2008**, *23*, 4–13. [CrossRef]
48. Shaw, R.; Jackman, B. An introduction to FlexRay as an industrial network. In *Proceedings of 2008 IEEE International Symposium on Industrial Electronics*; IEEE: New York, NY, USA, 2008; pp. 1849–1854. [CrossRef]
49. Makowitz, R.; Temple, C. Flexray—A communication network for automotive control systems. In *Proceedings of 2006 IEEE International Workshop on Factory Communication Systems*; IEEE: New York, NY, USA, 2006; pp. 207–212. [CrossRef]
50. Shreejith, S.; Fahmy, S.A. Extensible FlexRay Communication Controller for FPGA-Based Automotive Systems. *IEEE Trans. Veh. Technol.* **2015**, *64*, 453–465. [CrossRef]
51. Vaz, R.M.; Hodel, K.N.; Santos, M.M.D.; Arruda, B.A.; Netto, M.L.; Justo, J.F. An efficient formulation for optimization of FlexRay frame scheduling. *Veh. Commun.* **2020**, *24*, 100234. [CrossRef]
52. Kumar, N.; Mondal, A. Timing Analysis of Precedence Constraint Messages Scheduled With Slot Multiplexing Over Dynamic Segment of FlexRay. *IEEE Trans. Autom. Sci. Eng.* **2020**, *17*, 222–236. [CrossRef]
53. Boland, H.M.; Burgett, M.I.; Etienne, A.J.; Stwalley, R.M., III. An Overview of CAN-BUS Development, Utilization, and Future Potential in Serial Network Messaging for Off-Road Mobile Equipment. In *Technology in Agriculture*; IntechOpen: London, UK, 2021. [CrossRef]
54. Han, K.; Mun, H.; Balakrishnan, M.; Yeun, C.Y. Enhancing security and robustness of Cyphal on Controller Area Network in unmanned aerial vehicle environments. *Comput. Secur.* **2023**, *135*, 103481. [CrossRef]
55. Alhosani, A.; Alshehhi, F.; Almenhali, M.; Abu Hilal, H. Optical Communication Advancements in Free Space and Applications of Free Space Orbital Technology. *J. Inst. Eng. (India) Ser. B* **2025**, *106*, 805–814. [CrossRef]
56. InfiniBand Trade Association. *InfiniBand Architecture Specification Volume 1 Release 1.5*; IBTA: Beaverton, OR, USA, 2021. Available online: https://infinibandta.org/new-infiniband-and-roce-specification-introduces-memory-placement-extensions-to-significantly-reduce-persistent-memory-latency/ (accessed on 2 October 2026).
57. Zhu, Y.; Eran, H.; Firestone, D.; Guo, C.; Lipshteyn, M.; Liron, Y.; Padhye, J.; Raindel, S.; Yahia, M.H.; Zhang, M. Congestion Control for Large-Scale RDMA Deployments. *ACM SIGCOMM Comput. Commun. Rev.* **2015**, *45*, 523–536. [CrossRef]
58. Li, Y.; Miao, R.; Liu, H.H.; Zhuang, Y.; Feng, F.; Tang, L.; Cao, Z.; Zhang, M.; Kelly, F.; Alizadeh, M.; et al. HPCC: High Precision Congestion Control. In *Proceedings of ACM Special Interest Group on Data Communication*; ACM: New York, NY, USA, 2019; pp. 44–58. [CrossRef]
59. Mittal, R.; Shpiner, A.; Panda, A.; Zahavi, E.; Krishnamurthy, A.; Ratnasamy, S.; Shenker, S. Revisiting network support for RDMA. In *Proceedings of 2018 Conference of the ACM Special Interest Group on Data Communication*; ACM: New York, NY, USA, 2018; pp. 313–326. [CrossRef]
60. Kalia, A.; Kaminsky, M.; Andersen, D.G. Design guidelines for high performance RDMA systems. In Proceedings of the 2016 USENIX Annual Technical Conference, Denver, CO, USA, 22–24 June 2016; pp. 437–450. Available online: https://www.usenix.org/conference/atc16/technical-sessions/presentation/kalia (accessed on 2 October 2026).
61. Kalia, A.; Kaminsky, M.; Andersen, D.G. Using RDMA efficiently for key-value services. In *Proceedings of 2014 ACM conference on SIGCOMM*; ACM: New York, NY, USA, 2014; pp. 295–306. [CrossRef]

62. Dragojević, A.; Narayanan, D.; Nightingale, E.B.; Renzelmann, M.; Shamis, A.; Badam, A.; Castro, M. No compromises: Distributed transactions with consistency, availability, and performance. In *Proceedings of 25th Symposium on Operating Systems Principles*; ACM: New York, NY, USA, 2015; pp. 54–70. [CrossRef]
63. Wei, X.; Shi, J.; Chen, Y.; Chen, R.; Chen, H. Fast in-memory transaction processing using RDMA and HTM. In *Proceedings of 25th Symposium on Operating Systems Principles*; ACM: New York, NY, USA, 2015; pp. 87–104. [CrossRef]
64. Ma, T.; Chen, K.; Ma, S.; Song, Z.; Wu, Y. Thinking More about RDMA Memory Semantics. In *Proceedings of 2021 IEEE International Conference on Cluster Computing (CLUSTER)*; IEEE: New York, NY, USA, 2021; pp. 456–467. [CrossRef]
65. Grant, R.E.; Rashti, M.J.; Balaji, P.; Afsahi, A. Scalable connectionless RDMA over unreliable datagrams. *Parallel Comput.* **2015**, *48*, 15–39. [CrossRef]
66. Hoefler, T.; Roweth, D.; Underwood, K.; Alverson, R.; Griswold, M.; Tabatabaee, V.; Kalkunte, M.; Anubolu, S.; Shen, S.; McLaren, M.; et al. Data Center Ethernet and Remote Direct Memory Access: Issues at Hyperscale. *Computer* **2023**, *56*, 67–77. [CrossRef]
67. Ma, S.; Ma, T.; Chen, K.; Wu, Y. A Survey of Storage Systems in the RDMA Era. *IEEE Trans. Parallel Distrib. Syst.* **2022**, *33*, 4395–4409. [CrossRef]
68. Zeng, S.; Qi, F.; Han, L.; Gong, X.; Wu, T. Research and Evaluation of RoCE in IHEP Data Center. *EPJ Web Conf.* **2021**, *251*, 02018. [CrossRef]
69. Liu, J.; Jiang, W.; Wyckoff, P.; Panda, D.K.; Ashton, D.; Buntinas, D.; Gropp, W.; Toonen, B. Design and Implementation of MPICH2 over InfiniBand with RDMA Support. In *Proceedings of the 18th International Parallel and Distributed Processing Symposium*; IEEE: New York, NY, USA, 2004; pp. 16–25. [CrossRef]
70. Kissel, E.; Swany, M. Evaluating High Performance Data Transfer with RDMA-based Protocols in Wide-Area Networks. In *Proceedings of 2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems*; IEEE: New York, NY, USA, 2012; pp. 802–811. [CrossRef]
71. Feng, Z.; Gu, Z.; Yu, H.; Deng, Q.; Niu, L. Online Rerouting and Rescheduling of Time-Triggered Flows for Fault Tolerance in Time-Sensitive Networking. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2022**, *41*, 4253–4264. [CrossRef]
72. Gärtner, C.; Rizk, A.; Koldehofe, B.; Guillaume, R.; Kundel, R.; Steinmetz, R. Fast incremental reconfiguration of dynamic time-sensitive networks at runtime. *Comput. Netw.* **2023**, *224*, 109606. [CrossRef]
73. Gartner, C.; Rizk, A.; Koldehofe, B.; Guillaume, R.; Kundel, R.; Steinmetz, R. On the Incremental Reconfiguration of Time-sensitive Networks at Runtime. In *Proceedings of the 2022 IFIP Networking Conference (IFIP Networking)*; IEEE: New York, NY, USA, 2022; pp. 1–9. [CrossRef]
74. Zhao, L.; Zhang, X.; He, F.; Li, Z. Incremental Performance Analysis for Accelerating Verification of TSN Network Reconfigurations. *IEEE Trans. Netw. Serv. Manag.* **2024**, *21*, 5475–5490. [CrossRef]
75. Feng, Z.; Wu, C.; Deng, Q.; Han, M.; Lin, Y. ReT-FTS: Re-transmission-based fault-tolerant scheduling in TSN. *J. Syst. Archit.* **2023**, *142*, 102959. [CrossRef]
76. Boyer, M.; Henia, R. Embedded Reconfiguration of TSN (Industrial Challenge Description). In *37th Euromicro Conference on Real-Time Systems (ECRTS 2025)*; LIPIcs; Schloss Dagstuhl—Leibniz-Zentrum für Informatik: Brussels, Belgium, 2025; Volume 335, pp. 22:1–22:11. [CrossRef]
77. Zamanian, E.; Yu, X.; Stonebraker, M.; Kraska, T. Rethinking database high availability with RDMA networks. *Proc. VLDB Endow.* **2019**, *12*, 1637–1650. [CrossRef]
78. Jasny, M.; Ziegler, T.; Nelson-Slivon, J.; Leis, V.; Binnig, C. Synchronizing Disaggregated Data Structures with One-Sided RDMA: Pitfalls, Experiments and Design Guidelines. *ACM Trans. Database Syst.* **2025**, *50*, 4. [CrossRef]
79. NVIDIA. RDMA Aware Networks Programming User Manual, Rev. 1.7. Available online: https://docs.nvidia.com/rdma-aware-networks-programming-user-manual-1-7.pdf (accessed on 6 November 2026).
80. Le Boudec, J.-Y.; Thiran, P. Network Calculus: A Theory of Deterministic Queuing Systems for the Internet. In *Lecture Notes in Computer Science*; Springer: Berlin/Heidelberg, Germany, 2001; Volume 2050. [CrossRef]

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