

**ADAPTIVE MULTI-FABRIC FAILOVER BETWEEN NVMe/FC AND NVMe/TCP
USING REAL-TIME PATH-HEALTH TELEMETRY**

Gagan Vyas
Member of technical staff
Pure storage
Santa clara California, United States
gaganvyas@gmail.com

Abstract

Storage networks that use Non-Volatile Memory Express over Fabrics (NVMe-oF) must tolerate fabric failures without causing latency spikes for the applications they support, yet no unified approach exists for handling failures between a Fibre Channel (FC) fabric and a Transmission Control Protocol (TCP) fabric. This systematic review summarizes 27 sources, comprising 22 peer-reviewed studies and five Internet standards, that cover remote flash access, data center failure behavior, telemetry, congestion signaling, and multipath transport. The extracted evidence was compiled into a tabular matrix, and an analytical framework was built in which telemetry signals are normalized and combined into a path-health score, with a hysteresis rule governing switching between fabrics. The principal quantitative results are as follows: (1) a legacy block layer has a ceiling of approximately 800,000 operations per second, compared with a ceiling of up to 10 million operations per second for a multi-queue design; (2) NVMe-oF adds a remote-access latency of 11.7 microseconds; and (3) when each of 100 parallel sub-requests exceeds a latency threshold with probability 0.01, the probability that the aggregate request is delayed is 63 percent. An illustrative parameterization yields a failover time of 200 milliseconds for a hard failure. No reviewed source reports controlled measurements of failover between two fabrics; the thresholds in the framework are therefore design values that require empirical validation. The synthesis provides a structured approach to the telemetry-driven design of multi-fabric storage.

Index Terms – NVMe over Fabrics, Fibre Channel, NVMe/TCP, Failover, Path Health, Telemetry, Multipath Transport, Tail Latency, Congestion Signaling, Storage Networking.

I. INTRODUCTION

Non-volatile memory express (NVMe) devices have shifted the performance bottleneck of enterprise storage from the storage medium to the surrounding software and network. The Linux block layer, which was originally designed to support thousands of input/output operations per second (IOPS), reached its limit of 800,000 IOPS because a single coarse lock protected the request queue, whereas a multi-queue redesign was demonstrated to support up to 10 million IOPS [1]. This scaling gap is summarized in Fig. 1. A remote path that is not as parallel as the device wastes the parallel investment in flash.

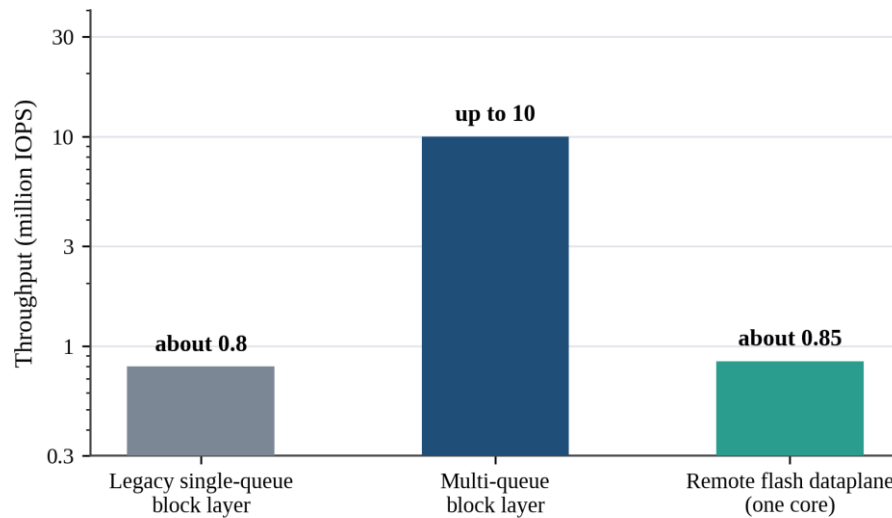


Fig. 1. Throughput of the legacy single-queue block layer, the multi-queue block layer, and a remote flash dataplane on one core.

For database workloads, measurements of NVMe solid-state drives (SSDs) suggest that device-level parallelism is beneficial only when the rest of the input/output (I/O) path introduces minimal queuing and serialization [2]. Storage traffic that traverses a network fabric then has stringent latency and availability requirements. If the fabric degrades or fails, the initiator either slows down or drops the path, and latency excursions flow directly into application response times.

NVMe over Fabrics (NVMe-oF) extends the NVMe command set to network transports. A detailed characterization over a remote direct memory access (RDMA)-capable 100 Gb/s Ethernet fabric measured an added latency of 11.7 μ s relative to direct-attached access, together with approximately 10 percent additional host processor utilization [3]. Transport diversity, such as Fibre Channel (FC) and Transmission Control Protocol (TCP), provides the opportunity explored here: a single initiator can be attached to two different fabrics and can direct I/O traffic between them based on the health metrics of the paths. The present review summarizes the available findings and develops an analytical failover model. The remainder of this paper describes the review methodology, remote flash access and fabric design, failure and latency evidence, telemetry and path-health scoring, multipath mechanisms, and conclusions.

II. REVIEW METHODOLOGY

A total of 27 sources were reviewed, comprising 22 peer-reviewed conference or journal papers and 5 Internet standards published as Requests for Comments (RFCs). The publication dates span approximately 20 years (2001–2020), and the corpus ranges from loss-based congestion signaling to explicit in-network telemetry. To be included, a source had to relate to one of the following: remote flash access, fabric failure behavior, path telemetry, congestion signaling, or multipath transport. Legacy protocols such as the Internet small computer systems interface (iSCSI) were retained as baselines for comparing NVMe-oF overheads [4].

Each source was categorized according to its type of evidence (measurement study, system design,

or normative specification) and the presence of quantitative results. The studies on the application-level costs of remote flash [5] were given the highest priority for parameterizing the model, whereas the specifications provided timing and signaling semantics. That no source in the corpus reports controlled failover measurements between FC and TCP fabrics is treated as a limitation, and transferable evidence from adjacent domains is synthesized instead. Table I maps the sources to concepts.

TABLE I. MAPPING OF REVIEWED SOURCES TO CONCEPTS

Concept	Sources	Count	Share (%)	Principal evidence type
Storage stack and remote flash access	[1], [2], [3], [5], [6]	5	18.5	Performance characterizations
Baseline protocols and standards	[4], [16]	2	7.4	Normative specifications
Topology and load distribution	[7], [8], [24], [25]	4	14.8	System designs and evaluations
Failure and latency characterization	[9], [10], [11]	3	11.1	Measurement studies
Telemetry and fault detection	[12], [13], [14]	3	11.1	Systems and specifications
Congestion signaling and control	[15], [17], [18], [19], [20]	5	18.5	Algorithms and evaluations
Lossless Ethernet and RDMA operation	[21], [22]	2	7.4	Deployment studies
Multipath transport	[23], [26], [27]	3	11.1	Protocol designs and specifications
Total	[1]-[27]	27	100.0	—

Note. Shares are rounded to one decimal place and therefore may not sum exactly to 100.0; each source is assigned to exactly one concept.

III. REMOTE FLASH ACCESS AND FABRIC DESIGN

A streamlined software path allows remote flash performance to approach local performance. A dataplane for remote flash was found to support 850,000 IOPS per core on commodity Ethernet and to add only 21 microseconds of latency relative to local flash while enforcing tail-latency objectives among competing tenants [6]. This is higher than the 11.7 μ s measured for NVMe-oF, but it is small compared with the approximately 20 percent application throughput loss reported for iSCSI (Table II). Fig. 2 summarizes these remote-access overheads.

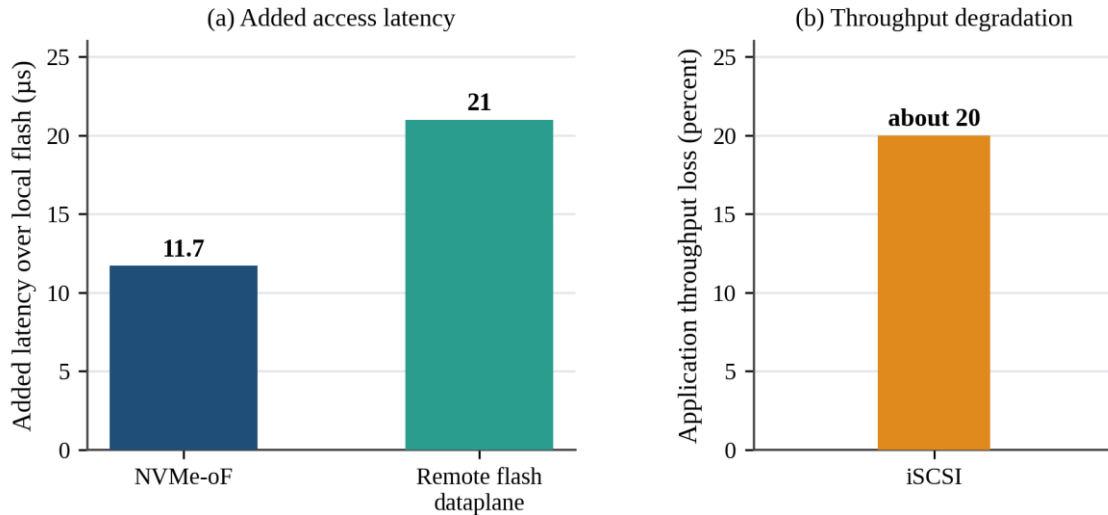


Fig. 2. Remote-access overhead: (a) added latency over local flash; (b) application throughput loss relative to local access.

The topology of the fabric determines how many failover routes are possible. A k -ary fat tree constructed from identical k -port commodity switches can provide $(k/2)^2$ equal-cost paths between hosts in different pods and supports $k^3/4$ hosts [7]. For $k = 48$, these expressions yield 27,648 hosts and 576 paths. Redundancy within a single fabric is thus abundant, yet failover between dissimilar fabrics is still required when a shared dependency, such as a software bug or a configuration mistake, affects every path of a single fabric.

Valiant load balancing and equal-cost multipath (ECMP) forwarding were used to distribute traffic uniformly across a Clos topology, and flat addressing was used to separate server identity from location [8]. These designs provide a statistical load balance but are unaware of the instantaneous quality of each path, which motivates the use of telemetry. Table II summarizes the quantitative evidence extracted from the corpus that is cited in the following sections.

TABLE II. QUANTITATIVE EVIDENCE EXTRACTED FROM THE CORPUS

Metric	Value	Source
Throughput ceiling of the legacy block layer	About 800,000 IOPS	[1]
Throughput of the multi-queue block layer	Up to 10 million IOPS	[1]
NVMe-oF added latency over direct-attached access	11.7 μs	[3]
NVMe-oF host processor utilization overhead	About 10%	[3]
Application throughput loss with iSCSI	About 20%	[5]
Remote flash dataplane throughput per core	About 850,000 IOPS	[6]
Remote flash dataplane added latency	21 μs	[6]
Fat-tree capacity at $k = 48$	27,648 hosts; 576 paths	[7]
Median failure-impact reduction from redundancy	40%	[9]
Probing scale and data volume	2,000–5,000 targets per server; tens of TB per day	[10]

Metric	Value	Source
Packet drop rate	About 10^{-5} ; inter-rack 2–6 times intra-rack	[10]
Delayed aggregate requests at $n = 100, p = 0.01$	63%	[11]
Hedged-request 99.9th percentile latency	1,800 ms to 74 ms with about 2% extra requests	[11]
DCTCP weight and marking thresholds	$g = 1/16$; 20 packets (1 Gb/s); 65 packets (10 Gb/s)	[15]

IV. FAILURE AND LATENCY EVIDENCE

A measurement study of data center network failure logs found that failure rates differed widely across device classes and that network redundancy reduced the median failure impact by only approximately 40 percent [9]. Redundancy within one fabric therefore provides only partial protection. This finding reinforces the case for a second, independent fabric, although failures may be correlated between fabrics, as discussed in the final section.

A latency measurement system that had been in operation for over four years was reported. Each server probed 2,000–5,000 targets, and latency data were collected at tens of terabytes per day [10].

The observed packet drop rates were of the order of 10^{-5} , with drops between racks two to six times higher than drops within a rack, and silent drops occurring on individual switches [10]. Partial degradations that link-state signals cannot see are thus exposed by active probes.

As the fan-out increases, latency increases as well. Assume that the n sub-requests of an aggregate request are independent and that each exceeds a latency threshold with probability p ; the probability P_D that the aggregate request exceeds the threshold is then given by Eq. 1 [11]. The delayed fraction is 63 percent for $p = 0.01$ and $n = 100$. Hedged requests, which are issued after a brief delay, reduced the 99.9th percentile latency of a 1,000-key read across 100 servers from 1,800 ms to 74 ms, at the cost of slightly more than 2 percent additional requests. The amplification is quantified in Table III and illustrated in Fig. 3.

$$P_D = 1 - (1 - p)^n \quad (1)$$

TABLE III. PROBABILITY OF A DELAYED AGGREGATE REQUEST COMPUTED FROM EQ. 1

Fan-out n	$p = 0.001$ (%)	$p = 0.01$ (%)	$p = 0.05$ (%)
1	0.1	1.0	5.0
10	1.0	9.6	40.1
50	4.9	39.5	92.3
100	9.5	63.4	99.4
200	18.1	86.6	100.0
500	39.4	99.3	100.0

Note. Values are computed directly from Eq. 1 and are rounded to one decimal place.

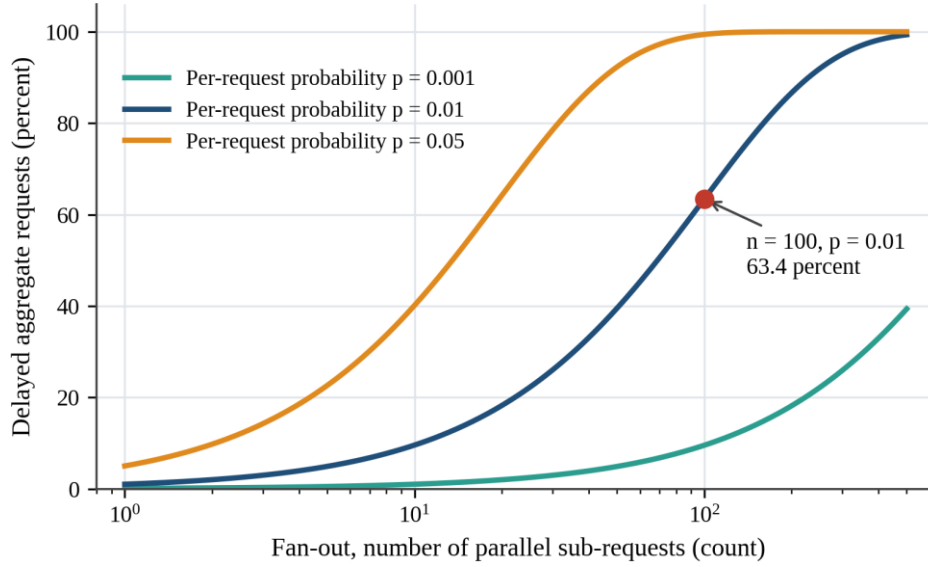


Fig. 3. Probability of a delayed aggregate request as a function of fan-out (Eq. 1).

V. TELEMETRY, DETECTION, AND PATH-HEALTH SCORING

A. Liveness and Packet-Level Telemetry

Aggregate counters provide an overview of packet drops, loops, and load imbalance, whereas packet-level telemetry can mirror a packet that satisfies operator-specific criteria and can initiate guided probes along specific paths, thereby providing more detailed information about these events [12]. Not mirroring all traffic keeps the collection overhead manageable, because only a subset of the packets is collected. The equivalent per-I/O signal for an NVMe path is the completion record of each command, which provides real-time information on path behavior without extra probe traffic.

Bidirectional Forwarding Detection (BFD) treats a path as down if no control packets are received within the detection time T_{det} , which is defined in Eq. 2 as the product of the detect multiplier m and the transmission interval τ [13].

$$T_{det} = m\tau \quad (2)$$

As stated in Eq. 3, the total failover time T_{fo} comprises the detection time, the decision delay T_{dec} , and the switchover delay T_{sw} .

$$T_{fo} = T_{det} + T_{dec} + T_{sw} \quad (3)$$

A longer interval lowers the load on the control plane but lengthens the detection time, so the interval should be chosen according to the latency requirements of the workload. The resulting detection times for various parameter choices are shown in Fig. 4.

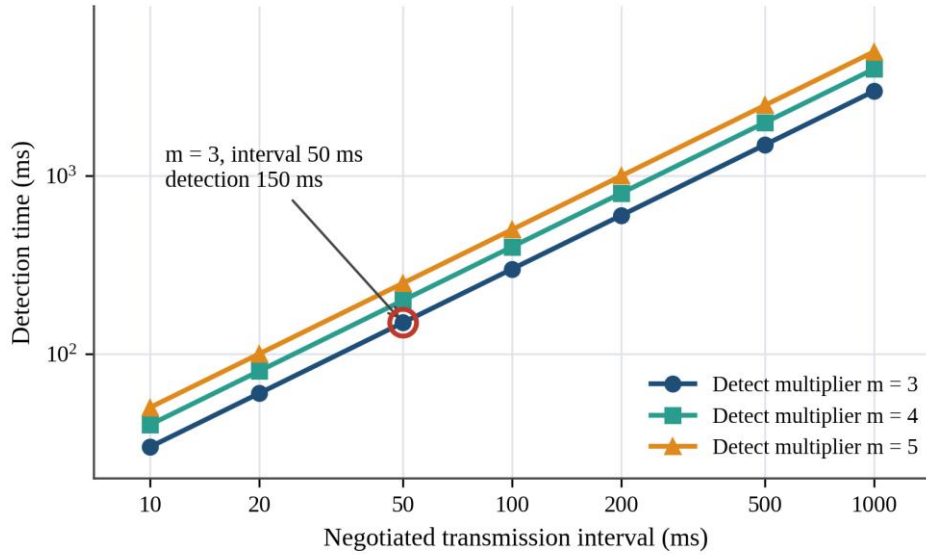


Fig. 4. BFD detection time as a function of transmission interval and detect multiplier (Eq. 2).

B. Congestion Signals

Explicit congestion notification (ECN) allows a router to set a code point in the Internet Protocol (IP) header to mark a packet as having experienced congestion; the receiver then echoes the mark to the sender by means of a flag in the TCP header [14]. The fraction of marked packets in a window is thus a loss-free measure of queue buildup on a TCP path, well before retransmission timers expire.

Data center TCP (DCTCP) smooths the marked fraction F observed over the last window into an estimate α , which is updated by exponential averaging with weight g according to Eq. 4, and reduces the congestion window W according to Eq. 5 [15].

$$\alpha_{i+1} = (1 - g)\alpha_i + gF \quad (4)$$

$$W_{i+1} = W_i \left(1 - \frac{\alpha_i}{2}\right) \quad (5)$$

A weight of $g = 1/16$ was reported, together with marking thresholds of 20 packets at 1 Gb/s and 65 packets at 10 Gb/s. The smoothed estimate α is used as one of the inputs to the path-health score.

Classical loss-based control is the norm for any compliant TCP stack; slow start, congestion avoidance, fast retransmit after three duplicate acknowledgments, and fast recovery are specified for it [16]. Loss-based recovery, however, operates on a much longer timescale than flash access, so loss-based signals provide confirmation rather than early warning for storage.

Congestion-based control, in contrast, constructs an explicit model of the path. The bottleneck bandwidth (BtlBw) and the minimum round-trip propagation delay (RT_{prop}) are continually estimated, and the operating point is determined by their product, the bandwidth-delay product (BDP) of Eq. 6 [17].

$$BDP = BtlBw \times RT_{prop} \quad (6)$$

Periodic probing keeps both estimates up to date. These quantities are also used for normalization,

which keeps the health score comparable across paths of different capacity.

Computer-generated congestion control showed that a protocol can be designed from an explicit model of network uncertainty and an explicit congestion-control objective, and that the resulting protocol can outperform human-designed ones in the scenarios evaluated [18]. Similarly, for path selection, the goal of low tail latency with bounded switching is stated explicitly, and the controller architecture of Fig. 5 is derived from that goal. A path selector is controlled by a decision engine fed by a scorer, and the scorer is in turn fed by telemetry collectors on both fabrics.

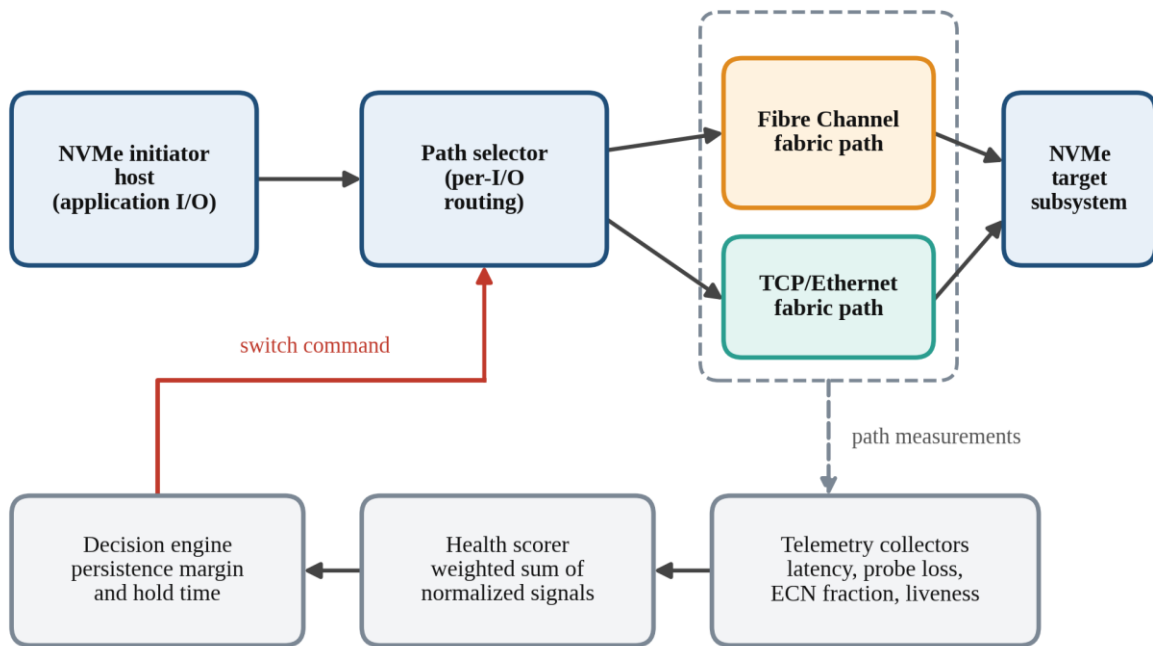


Fig. 5. Architecture of the adaptive multi-fabric failover controller.

In data centers, the gradient of the round-trip time (RTT), as measured by hardware timestamps, was found to be a good indicator of congestion, in contrast to the absolute value of the RTT [19]. The quantity G is the normalized gradient defined in Eq. 7,

$$G = \frac{\bar{d}}{RTT_{\min}}, \quad (7)$$

where $\bar{d}$ is the exponentially weighted moving average of the differences between consecutive RTT samples and $RTT_{\min}$ is the minimum RTT. A positive gradient indicates growing queues. The latency measured by the initiator is the completion latency, which can be obtained without support from switches.

In-network telemetry (INT) is the ability of each switch to add its queue length, transmitted-byte counter, and timestamp to passing packets, which enables exact per-link load computation. Based on this information, high precision congestion control (HPCC) computes the utilization U_j of link j (Eq. 8) and uses it to adjust the sending window [20].

$$U_j = \frac{q_j}{B_j T} + \frac{tx_j}{B_j}, \quad (8)$$

where q_j is the queue length, B_j is the bandwidth, T is the base RTT, and tx_j is the transmission rate

of link j . INT requires programmable hardware, which is not available in all fabrics, so support is uneven. Table IV compares the telemetry signals discussed above.

TABLE IV. COMPARISON OF PATH TELEMETRY SIGNALS

Signal	Mechanism	Timescale	Scope of detection	Fabric support	Source
Liveness state	Periodic control packets	Detection time of Eq. 2	Hard path failure	End-to-end	[13]
Active probe loss and latency	Server-to-server probing	Aggregated over probe windows	Partial degradation; silent drops	Hosts only	[10]
Mirrored packets	Rule-based mirroring and guided probes	Per packet	Drops, loops, imbalance	Switch rules	[12]
ECN mark fraction	Router marking echoed by receiver	Per window	Incipient congestion	ECN-capable switches	[14], [15]
Loss events	Duplicate acknowledgments and timers	Per loss event	Congestion and failure	None required	[16]
Bandwidth and propagation delay	Model-based estimation	Continuous probing	Capacity change	Hosts only	[17]
RTT gradient	Hardware timestamps	Per RTT	Queue growth	NIC timestamps	[19]
Per-hop link load	In-network telemetry	Per packet	Per-link utilization	Programmable switches	[20]

Note. Descriptions are synthesized from the cited sources; no new measurements are reported. NIC denotes network interface card.

C. Lossless Fabric Operation and the Health Score

The congestion-control algorithm for large-scale RDMA deployments operates as follows: the receiver sends a notification packet to the sender, the sender reduces its rate upon receiving the notification packet, and the sender then gradually returns to its normal rate [21]. The scheme was conceived for lossless Ethernet operating in priority flow control (PFC) mode, in which pause-based back-pressure is the principal protection mechanism. Path-health scoring therefore needs to interpret congestion signals according to the semantics of each fabric, and marking thresholds need to be calibrated per transport.

When RDMA was used over commodity Ethernet at scale, operational experience revealed risks of lossless operation, such as pause-frame storms, deadlocks, and livelocks, which were addressed with watchdogs and safety mechanisms [22]. These failure modes weaken a path without breaking it, so they cannot be detected by liveness detection. A composite health score $H_p(t)$ is therefore defined as a weighted sum of the normalized signals $x_{p,k}$ with weights w_k and reference thresholds θ_k , as given in Eq. 9, where p denotes the path, K is the number of signals, and the weights sum to 1.

$$H_p(t) = \sum_{k=1}^K w_k \left[1 - \min \left(1, \frac{x_{p,k}(t)}{\theta_k} \right) \right] \quad (9)$$

Failover from the active path p to an alternative path q is governed by a hysteresis rule: the score of q must exceed that of p by a margin δ in each of N consecutive scoring intervals of length Δ , as

given in Eq. 10, and failback to the original path is further delayed by a hold time T_h .

$$H_q(t - j\Delta) - H_p(t - j\Delta) > \delta, \quad j = 0, 1, \dots, N - 1 \quad (10)$$

Fig. 6 shows a synthetic trace of this rule, and Table V lists the illustrative parameterization of the failover model.

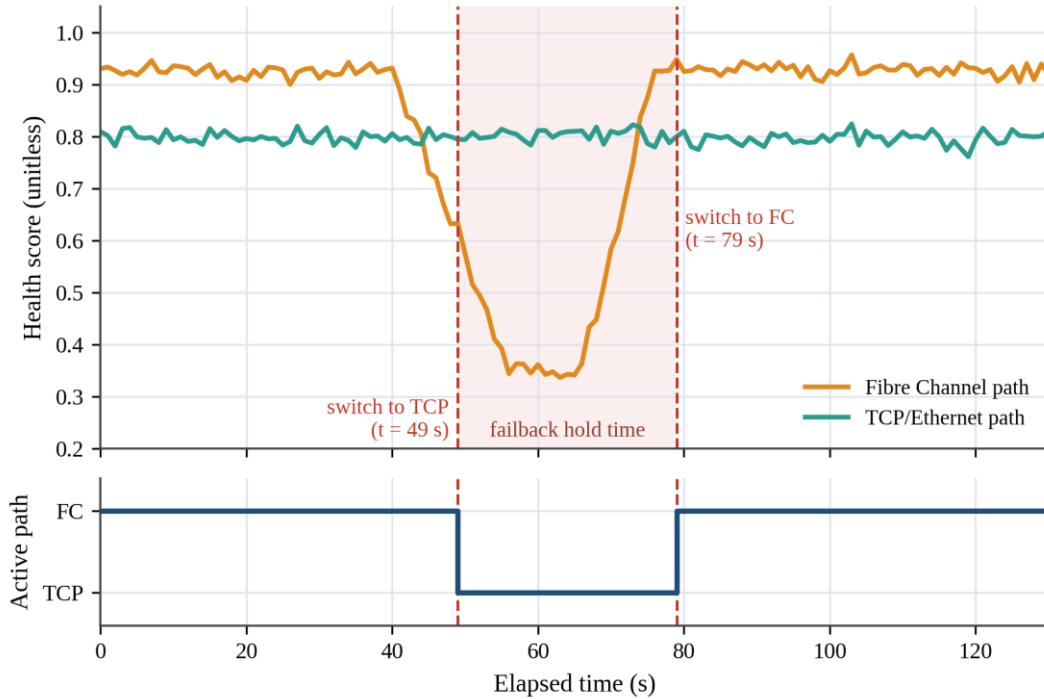


Fig. 6. Synthetic trace of the health-score rule (Eqs. 9 and 10), illustrating failover to the TCP path and delayed failback.

TABLE V. ILLUSTRATIVE PARAMETERIZATION OF THE FAILOVER MODEL

Parameter	Symbol	Value	Basis
Detect multiplier	m	3	Design value; semantics from [13]
Transmission interval	τ	50 ms	Design value; semantics from [13]
Detection time (Eq. 2)	T_{det}	150 ms	Computed
Decision delay	T_{dec}	20 ms	Design value
Switchover delay	T_{sw}	30 ms	Design value
Failover time (Eq. 3)	T_{fo}	200 ms	Computed
Scoring interval	Δ	1 s	Design value
Persistence count	N	3 intervals	Design value
Advantage margin	δ	0.10	Design value
Failback hold time	T_h	30 s	Design value
Smoothing weight	g	1/16	Reported in [15]

Note. Entries marked as design values are assumptions for illustration and are not measurements.

VI. MULTIPATH AND LOAD-DISTRIBUTION MECHANISMS

Re-architecting data center networks and stacks produced a protocol in which an overloaded switch trims the payload of a packet rather than dropping it entirely, receivers pull data at a controlled rate, and packets are sprayed over all available paths [23]. Trimming keeps the loss signal at the header level and is an example of a path reporting degradation without waiting for timeouts.

Leaf switches were implemented to exchange congestion feedback within a two-tier topology and to choose the least-congested uplinks, performing distributed congestion-aware load balancing at flowlet granularity [24]. The mechanism responds on round-trip timescales and also handles asymmetry caused by link failure, which demonstrates the potential of path-quality feedback for fine-grained steering.

Path utilization was propagated periodically across the network by probes, and each switch stored state only for the best next hop per destination [25]. The design shows that, when thousands of storage paths must be monitored, constant per-switch state is appropriate for steering because it simplifies the design of the telemetry.

Multipath TCP (MPTCP) was shown to increase the throughput and robustness of a data center by splitting a TCP connection into several subflows and applying coupled congestion control; with a small number of subflows, near-saturation of a fat-tree topology was achieved [26]. Subflow independence also provides resilience: if one subflow fails, the connection remains unaffected.

The standardized multipath extension to TCP provides for multiple addresses per connection and a data-level sequence mapping, which allows unacknowledged data that failed to flow over one subflow to be retransmitted over another [27]. If two paths carry redundant requests and fail independently, the probability that an I/O exceeds a latency bound t is approximated by Eq. 11,

$$P(L > t) \approx \prod_{i=1}^2 P(L_i > t), \quad (11)$$

where L is the I/O latency and L_i is the latency on path i . Table VI compares the multipath and load-distribution mechanisms.

TABLE VI. COMPARISON OF MULTIPATH AND LOAD-DISTRIBUTION MECHANISMS

Mechanism	Decision point	Granularity	Information used	Failure or asymmetry handling	Source
ECMP with Valiant load balancing	Switch hash	Flow	None about load	Static	[8]
NDP	Sender and switch	Packet	Trimmed headers	Loss signal at header level	[23]
CONGA	Leaf switch	Flowlet	Leaf-to-leaf congestion feedback	Adapts to asymmetry	[24]
HULA	Switch	Flowlet	Probed path utilization	Best next hop per destination	[25]
MPTCP	Host	Subflow	Per-subflow congestion state	Connection survives subflow failure	[26]
MPTCP (standardized)	Host	Subflow	Multiple addresses; data mapping	Retransmission on another subflow	[27]

Note. None of the mechanisms steers I/O between FC and TCP fabrics.

VII. CONCLUSION AND FUTURE DIRECTIONS

This review combined 27 sources and provided an analytical framework for failover between FC and TCP fabrics. The evidence supports four points. First, remote flash access adds only 11.7 μ s to 21 μ s of latency relative to local access. Second, in-fabric redundancy is only a partial solution to failure effects. Third, tail latency grows rapidly with fan-out, with a delay probability of 63 percent at 100 sub-requests. Fourth, complementary telemetry signals are available on timescales of milliseconds for liveness detection and of seconds for congestion and probe statistics. Together, these points suggest a two-tier design: a fast liveness path for hard failures and a slower, scored path for gradual degradation.

These conclusions are subject to several limitations. No reviewed source reports controlled failover measurements between the two fabrics, and the framework assumes that the signals available on the FC side are similar to those found for Ethernet. The independence assumption of Eq. 11 is optimistic given the reported 40 percent effectiveness of redundancy, since shared dependencies correlate failures. Table V presents design parameters, not measurements. Deployment is further limited by the telemetry overhead of a large probing system and by the reliance of INT on programmable hardware.

The framework needs to be validated in real deployments, the oscillation caused by noisy telemetry needs to be measured, and the overhead of redundant issuance must be bounded; the approximately 2 percent reported for hedged requests is a useful reference point. Hand-set weights and thresholds could be replaced by learned or synthesized policies, and completion-latency gradients could be used alongside liveness detection to provide early warning.

REFERENCES

1. M. Bjørling, J. Axboe, D. Nellans, and P. Bonnet, "Linux block IO: Introducing multi-queue SSD access on multi-core systems," in *Proc. 6th Int. Syst. Storage Conf.*, 2013, Art. no. 22, doi: 10.1145/2485732.2485740.
2. Q. Xu *et al.*, "Performance analysis of NVMe SSDs and their implication on real world databases," in *Proc. 8th ACM Int. Syst. Storage Conf.*, 2015, Art. no. 6, doi: 10.1145/2757667.2757684.
3. Z. Guz, H. H. Li, A. Shayesteh, and V. Balakrishnan, "NVMe-over-Fabrics performance characterization and the path to low-overhead flash disaggregation," in *Proc. 10th ACM Int. Syst. Storage Conf.*, 2017, doi: 10.1145/3078468.3078483.
4. J. Satran, K. Meth, C. Sapuntzakis, M. Chadalapaka, and E. Zeidner, "Internet small computer systems interface (iSCSI)," RFC Editor, RFC 3720, 2004, doi: 10.17487/RFC3720.
5. A. Klimovic, C. Kozyrakis, E. Thereska, B. John, and S. Kumar, "Flash storage disaggregation," in *Proc. 11th Eur. Conf. Comput. Syst.*, 2016, Art. no. 29, doi: 10.1145/2901318.2901337.
6. A. Klimovic, H. Litz, and C. Kozyrakis, "ReFlex: Remote flash $\approx$ local flash," in *Proc. 22nd Int. Conf. Archit. Support Program. Lang. Oper. Syst.*, 2017, pp. 345–358, doi: 10.1145/3037697.3037732.
7. M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proc. ACM SIGCOMM 2008 Conf. Data Commun.*, 2008, pp. 63–74, doi: 10.1145/1402958.1402967.
8. A. Greenberg *et al.*, "VL2: A scalable and flexible data center network," in *Proc. ACM SIGCOMM 2009 Conf. Data Commun.*, 2009, pp. 51–62, doi: 10.1145/1592568.1592576.

9. P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: Measurement, analysis, and implications," in *Proc. ACM SIGCOMM 2011 Conf.*, 2011, pp. 350–361, doi: 10.1145/2018436.2018477.
10. C. Guo *et al.*, "Pingmesh: A large-scale system for data center network latency measurement and analysis," in *Proc. 2015 ACM SIGCOMM Conf.*, 2015, pp. 139–152, doi: 10.1145/2785956.2787496.
11. J. Dean and L. A. Barroso, "The tail at scale," *Commun. ACM*, vol. 56, no. 2, pp. 74–80, 2013, doi: 10.1145/2408776.2408794.
12. Y. Zhu *et al.*, "Packet-level telemetry in large datacenter networks," in *Proc. 2015 ACM SIGCOMM Conf.*, 2015, pp. 479–491, doi: 10.1145/2785956.2787483.
13. D. Katz and D. Ward, "Bidirectional forwarding detection (BFD)," RFC Editor, RFC 5880, 2010, doi: 10.17487/RFC5880.
14. K. Ramakrishnan, S. Floyd, and D. Black, "The addition of explicit congestion notification (ECN) to IP," RFC Editor, RFC 3168, 2001, doi: 10.17487/RFC3168.
15. M. Alizadeh *et al.*, "Data center TCP (DCTCP)," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 4, pp. 63–74, 2010, doi: 10.1145/1851275.1851192.
16. M. Allman, V. Paxson, and E. Blanton, "TCP congestion control," RFC Editor, RFC 5681, 2009, doi: 10.17487/RFC5681.
17. N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-based congestion control," *Commun. ACM*, vol. 60, no. 2, pp. 58–66, 2017, doi: 10.1145/3009824.
18. K. Winstein and H. Balakrishnan, "TCP ex machina: Computer-generated congestion control," in *Proc. ACM SIGCOMM 2013 Conf.*, 2013, pp. 123–134, doi: 10.1145/2486001.2486020.
19. R. Mittal *et al.*, "TIMELY: RTT-based congestion control for the datacenter," in *Proc. 2015 ACM SIGCOMM Conf.*, 2015, pp. 537–550, doi: 10.1145/2785956.2787510.
20. Y. Li *et al.*, "HPCC: High precision congestion control," in *Proc. ACM Special Interest Group Data Commun.*, 2019, pp. 44–58, doi: 10.1145/3341302.3342085.
21. Y. Zhu *et al.*, "Congestion control for large-scale RDMA deployments," in *Proc. 2015 ACM SIGCOMM Conf.*, 2015, pp. 523–536, doi: 10.1145/2785956.2787484.
22. C. Guo *et al.*, "RDMA over commodity Ethernet at scale," in *Proc. 2016 ACM SIGCOMM Conf.*, 2016, pp. 202–215, doi: 10.1145/2934872.2934908.
23. M. Handley *et al.*, "Re-architecting datacenter networks and stacks for low latency and high performance," in *Proc. 2017 ACM SIGCOMM Conf.*, 2017, pp. 29–42, doi: 10.1145/3098822.3098825.
24. M. Alizadeh *et al.*, "CONGA: Distributed congestion-aware load balancing for datacenters," in *Proc. 2014 ACM SIGCOMM Conf.*, 2014, pp. 503–514, doi: 10.1145/2619239.2626316.
25. N. Katta, M. Hira, C. Kim, A. Sivaraman, and J. Rexford, "HULA: Scalable load balancing using programmable data planes," in *Proc. Symp. SDN Res.*, 2016, Art. no. 10, doi: 10.1145/2890955.2890968.
26. C. Raiciu *et al.*, "Improving datacenter performance and robustness with multipath TCP," in *Proc. ACM SIGCOMM 2011 Conf.*, 2011, pp. 266–277, doi: 10.1145/2018436.2018467.
27. Ford, C. Raiciu, M. Handley, O. Bonaventure, and C. Paasch, "TCP extensions for multipath operation with multiple addresses," RFC Editor, RFC 8684, 2020, doi: 10.17487/RFC8684.