Networking Technologies for Distributed Databases: From TCP/IP to RDMA and Programmable Fabrics

Elizbeth Espinal
Stevens Institute of Technology
Hoboken, New Jersey, USA

Abstract

Distributed databases are networked systems: every remote read, replicated write, quorum decision, failure-detection event, and reconfiguration crosses a communication substrate whose latency, throughput, loss behavior, and isolation properties shape database performance and availability. This survey organizes the supporting technologies into five layers—data-center and wide-area fabrics, transports, RPC and kernel-bypass mechanisms, acceleration and virtualization, and database coordination protocols. It compares TCP, QUIC, Ethernet, InfiniBand, RoCE, RDMA, user-space RPC, overlays, TLS, clock synchronization, and programmable switches, then relates them to representative systems including Spanner, CockroachDB, Aurora, FaRM, FaSST, and Raft-based stores. The central finding is that no network technology is uniformly best: reliable TCP-based stacks maximize portability and operational maturity; RDMA and kernel bypass minimize CPU and latency within controlled clusters; and WAN deployments benefit most from locality, topology-aware quorums, batching, and careful consistency choices. The survey concludes with selection guidance and research challenges involving tail latency, mixed lossless/lossy fabrics, multitenancy, observability, and safe in-network computation.

CCS Concepts

• **Information systems** → **Distributed database transactions**; • **Networks** → *Data center networks*; *Network performance analysis.*

Keywords

distributed databases, data-center networks, TCP, QUIC, RDMA, RPC, consensus, geo-replication, network virtualization

1 Introduction

Modern distributed databases partition and replicate state across many machines to obtain horizontal scale, fault tolerance, elasticity, and geographic reach. Those benefits make the network part of the database's critical path rather than a transparent delivery mechanism. A single transaction can involve client admission, a metadata lookup, reads from several shards, concurrency-control messages, replication to a quorum, durable logging, and a final acknowledgment. Each stage can add serialization delay, queueing, retransmission, or a remote round trip. When storage is based on flash or memory and execution is highly parallel, an extra message exchange or a short queue at a top-of-rack switch may dominate the end-to-end response time. Amazon Aurora's designers explicitly argued that the principal high-throughput constraint had shifted toward network traffic, motivating a log-centric protocol that sends compact redo records rather than full database pages [47].

This dependence is visible in several application classes. Interactive commerce, financial services, multiplayer systems, cloud control planes, and large web services often issue short database operations with strict tail-latency objectives. A request that fans out to multiple partitions is delayed by its slowest required response, so moderate variation at one host, NIC queue, or network path can become a visible percentile-latency spike. Write-heavy applications add replication traffic to that fan-out, while analytical scans, backup, compaction, rebalancing, and replica repair compete for the same links. Consequently, an application can be network-limited even when average link utilization appears low: packet-processing capacity, incast, burst synchronization, cross-region distance, or a few congested queues can determine throughput and availability.

The network is also a correctness boundary. A database cannot reliably distinguish a failed node from a delayed or partitioned node, and a transport that retransmits bytes does not by itself provide transaction atomicity, replica agreement, fencing, or bounded completion. Consensus, quorum replication, two-phase commit, leases, and failure detectors therefore convert network behavior into database semantics. Raft's leader-driven replicated log, for example, couples commit latency to majority communication and makes tail round-trip time and timeout calibration first-order concerns [39]. The CAP result further shows why a partition forces a choice between always accepting operations and preserving a single consistent view [18]. Network design must therefore be evaluated not only by bandwidth and median latency, but also by how it interacts with protocol safety, liveness, failover, and recovery.

The supporting technology landscape spans several layers. At the fabric layer, Ethernet Clos networks, InfiniBand, overlays, multipath routing, queue management, and isolation determine capacity and failure domains. At the transport and messaging layers, TCP, UDP, QUIC, RPC frameworks, batching, encryption, and kernel bypass trade portability against CPU cost and latency. RDMA, SmartNICs, and programmable switches move selected communication or coordination work closer to hardware, while service discovery, load balancers, clock synchronization, and topology-aware clients shape how database nodes are reached. These mechanisms are interdependent: a low-latency transport can still perform poorly with mismatched switch thresholds, unbounded RPC queues, unstable failure detection, or a replica layout that places every quorum across a wide-area path.

This survey studies that cross-layer design space through four questions. First, which network characteristics—latency, tail variation, bandwidth, message rate, ordering, loss behavior, and isolation—most strongly affect common database communication patterns? Second, when do general-purpose TCP/IP and RPC stacks provide the best engineering tradeoff, and when do kernel bypass or RDMA justify their operational complexity? Third, how should fabrics,

Table 1: Cross-layer taxonomy

Layer	Main options	Database effect
Fabric	Ethernet/Clos; InfiniBand	Capacity, path diversity, queueing
Virtual	VLAN/VXLAN; cloud VPC	Isolation, mobility, encapsulation
Transport	TCP; QUIC/UDP; RoCE	Reliability, ordering, congestion
Messaging	RPC; kernel bypass; batching	CPU cost, copies, tail latency
Coordination	Raft/Paxos; clocks; quorums	RTTs, consistency, failover

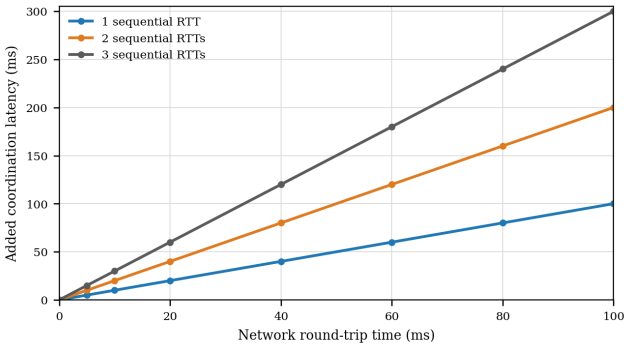

Figure 1: Analytical latency added by one, two, or three sequential network round trips. The model excludes processing, storage, retransmission, and queueing delay.

routing, congestion control, virtualization, security, and load balancing be configured around replication and transaction protocols? Fourth, how do geographic distance, cloud disaggregation, and multitenancy change the selection criteria? The analysis emphasizes transactional, key-value, and cloud-native systems and draws on standards, seminal designs, and operational studies through August 2026.

The paper makes three contributions. It presents a layered taxonomy that maps network mechanisms to database effects; compares mature and accelerated communication substrates using performance, correctness, deployment, and failure-recovery criteria; and provides practical selection and validation guidance. The scope is intentionally selective rather than encyclopedic: the goal is to identify capabilities that materially support correctness, performance, elasticity, security, and operability. Section 2 characterizes traffic and failure requirements; Sections 3–5 cover fabrics, transports, RPC, RDMA, and data-plane acceleration; Sections 6 and 7 connect networking to replication, geography, discovery, and load balancing; Sections 8 and 9 provide evaluation guidance and open research challenges; and Section 10 concludes.

2 Network Requirements and Communication Patterns

2.1 Latency, Throughput, and Tail Behavior

Database traffic combines many small control messages with large background transfers. Point reads, lock requests, Raft heartbeats, and commit acknowledgments are typically latency-sensitive and often only tens to hundreds of bytes. Snapshot installation, replica repair, backup, compaction, repartitioning, and analytical scans consume bandwidth for longer periods. The same cluster therefore experiences a mice-and-elephants workload in which bulk flows can inflate queues seen by short flows. DCTCP addresses this problem by using Explicit Congestion Notification (ECN) to react to the extent of congestion rather than waiting for loss; the original evaluation reported substantially lower buffer occupancy while retaining throughput [3]. A large-scale deployment study later confirmed reduced loss but also documented fairness, rollout, and implementation challenges, reminding designers that an algorithm's production behavior depends on the full host-and-switch ecosystem [11].

Mean latency is insufficient because distributed operations amplify stragglers. If a request waits for the slowest of several shards, or for any majority member among replicas, its percentile distribution reflects correlated queueing, retransmissions, pauses, and routing changes. Throughput also has multiple meanings: payload bandwidth, operations per second, packets per second, and CPU cycles per message. Fast networks can expose host-stack overhead, memory-copy cost, interrupt processing, serialization, and encryption as bottlenecks before links saturate.

Figure 1 isolates the network component of this effect. Each sequential coordination round adds one network round trip, so protocols that require multiple non-overlapped exchanges amplify latency even before queueing, logging, and execution are considered.

2.2 Failure Semantics and Partitions

The network may delay, duplicate, reorder, or drop packets; paths and endpoints can fail independently; and a slow node is difficult to distinguish from a partitioned one. Reliable transports hide ordinary loss but cannot guarantee bounded delivery or preserve a connection across every failure. The CAP result formalizes the impossibility of simultaneously guaranteeing consistency and availability in the presence of a partition [18]. Databases therefore add timeouts, epochs, idempotent request identifiers, fencing tokens, membership protocols, and recovery logs. The transport's retry behavior must be coordinated with application deadlines: long hidden retransmissions can turn an intended fast failover into a tail-latency event, whereas overly aggressive timeouts can trigger elections and duplicate work during transient congestion.

3 Fabrics, Routing, and Virtualization

3.1 Ethernet and Data-Center Fabrics

Switched Ethernet with an IP underlay is the dominant general-purpose data-center substrate. Leaf-spine or Clos topologies provide multiple equal-cost paths and high bisection bandwidth, while routing spreads flows across those paths. Commodity fat-tree [2], VL2 [20], and Google's Jupiter [44] established how Clos-derived fabrics and centralized control can scale cluster capacity. For a distributed database, the fabric's important properties are not merely nominal link rate but oversubscription, path diversity, convergence time, maximum transmission unit, ECN configuration, and queue

isolation. Replica placement should align with independent failure domains—racks, power zones, and availability zones—yet every added distance increases latency. Topology-aware clients and consensus groups can prefer nearby replicas while retaining remote copies for durability.

Congestion control determines whether bursts from synchronized replication or recovery create incast. Loss-based TCP remains broadly compatible, while ECN-based schemes such as DCTCP keep queues shorter in managed data centers [3], [11]. pFabric prioritizes packets to approach minimum flow-completion time [4], whereas receiver-driven Homa [37] and delay-based TIMELY [36] and Swift [30] target low tail latency under high utilization. Reverie targets buffer sharing between TCP and RoCE classes [1]. The lesson for database operators is that a transport cannot be tuned in isolation: switch thresholds, priority flow control, host pacing, NIC queues, and workload burstiness form one control loop.

In shared clusters, distributed databases also compete with coexisting applications for link bandwidth, switch buffers, and NIC queues. Bulk data transfers, machine-learning communication, backups, and analytical workloads can increase queueing delay and packet loss even when the database's own traffic remains unchanged. Such interference is especially harmful to latency-sensitive replication and consensus messages: delayed acknowledgments can increase transaction latency and replica lag and, in severe cases, trigger unnecessary timeouts or leader elections. Consequently, isolated measurements of average bandwidth may conceal substantial degradation in tail latency under realistic multi-application workloads. Traffic isolation, bandwidth guarantees, priority-aware scheduling, admission control, and sufficient capacity headroom are therefore important for maintaining predictable database performance in shared clusters [1, 4, 11, 28].

3.2 InfiniBand, RoCE, and Lossless Operation

InfiniBand and RDMA over Converged Ethernet (RoCE) support remote direct memory access with low CPU overhead and microsecond-scale communication. InfiniBand supplies an integrated lossless fabric, whereas RoCE places RDMA semantics over Ethernet; RoCEv2 is routable over IP/UDP but usually relies on carefully engineered congestion and loss controls. IEEE Priority Flow Control can suppress per-class loss [21] but may introduce head-of-line blocking or congestion spreading. Consequently, modern deployments combine traffic-class isolation with schemes such as DCQCN for RoCEv2 [51]. MP-RDMA demonstrates that exploiting path diversity can improve robustness and utilization, although NIC state and packet reordering complicate the design [34].

3.3 Overlays and Cloud Network Virtualization

Multi-tenant clouds interpose virtual switches, policy engines, tunnels, and software load balancers between database nodes. VXLAN encapsulates Layer-2 segments over a Layer-3 network and uses a 24-bit virtual network identifier, enabling large numbers of isolated overlays [35]. Encapsulation consumes MTU headroom and may alter hashing, observability, and failure modes. Google's Andromeda shows how a production cloud virtual network can dynamically map flows to different processing paths to balance feature richness, isolation, and hardware-like performance [9]. For databases, virtual networking enables elastic addressing, policy, and zone-spanning deployment, but added dataplane state can create jitter and gray failures. Private routing, stable endpoint identities, health-aware load balancing, and explicit MTU tests are therefore operational requirements.

4 Transport and Messaging Substrates

4.1 TCP as the Portable Baseline

TCP remains the default substrate for database clients, replication streams, and RPC because it offers reliable, ordered byte streams, congestion control, and ubiquitous support; RFC 9293 consolidates the current core specification [15]. Its strengths are operational maturity and compatibility across hosts, clouds, and wide-area networks. Its costs include connection establishment, per-connection state, head-of-line blocking within a stream, kernel crossings, copying, and sometimes opaque retransmission delays. Connection pooling and long-lived channels amortize handshakes, while batching, framing, and request multiplexing reduce syscall overhead. Databases should set application deadlines and keepalives deliberately rather than inheriting generic defaults.

4.2 UDP and QUIC

UDP exposes datagrams with minimal transport policy and is useful when an RPC layer wants to control retransmission, ordering, and congestion. Its simplicity also shifts correctness obligations to the application. QUIC occupies a middle ground: it runs over UDP but supplies secure, flow-controlled streams, low-latency establishment, and connection migration [22]. Independent streams avoid transport-level head-of-line blocking across unrelated requests, and user-space evolution is attractive for cloud services. However, database adoption remains less mature than TCP, and QUIC's 0-RTT mode is inappropriate for non-idempotent writes unless replay is prevented at the application layer. QUIC is most compelling for mobile or cross-Internet clients, multiplexed gateways, and environments where path migration matters; inside stable database clusters, its advantages must outweigh operational novelty and CPU cost.

4.3 RPC, Kernel Bypass, and Serialization

RPC frameworks add message framing, method dispatch, deadlines, backpressure, and code-generated serialization over transports. Their design can dominate small-operation cost. eRPC demonstrated that a general-purpose user-space RPC library on commodity Ethernet could achieve high message rates and low-latency Raft replication without requiring RDMA, using techniques such as session management, zero-copy packet I/O, and explicit congestion handling [25]. MICA similarly showed that NIC-level steering and a lightweight networking stack can be integral to key-value performance [33]. These systems illustrate the value of kernel bypass through mechanisms such as DPDK, io_uring-style asynchronous I/O, or vendor user-space drivers, but they also introduce CPU pinning, huge-page, NUMA, security, and operational constraints.

Serialization and batching are equally important. Compact binary encodings reduce bytes and parsing, while batching amortizes fixed per-packet, per-RPC, encryption, and consensus costs. Excessive batching increases queueing delay, so latency-sensitive

systems commonly use adaptive size-or-time thresholds. Backpressure must extend from the receiver through RPC queues to the client; otherwise a fast transport merely moves congestion into memory, causing unbounded queues and timeout storms.

4.4 Secure Channels

TLS protects confidentiality, integrity, and peer authentication for client and node-to-node traffic. The current TLS 1.3 specification is RFC 9846, which supersedes RFC 8446 and retains warnings that early data lacks ordinary non-replay guarantees [41]. Mutual TLS is widely used to authenticate cluster members, but certificate rotation, connection churn, and cryptographic CPU costs require engineering. Session resumption, hardware acceleration, connection pooling, and record sizing can reduce overhead. Encryption should be evaluated with realistic small-message workloads: link bandwidth may be abundant while per-record processing limits operations per second. Network policy and encryption complement rather than replace database authorization and application-level idempotency.

5 RDMA and Data-Plane Acceleration

5.1 One-Sided and Two-Sided RDMA

RDMA allows a NIC to place data directly into registered remote memory; RDMAP standardizes protected remote read and write services over direct data placement [40]. One-sided operations bypass the remote CPU but expose memory-layout and concurrency concerns, while two-sided SEND/RECEIVE resembles messaging. HERD shows that a carefully chosen mix of RDMA verbs can maximize key-value throughput [26]. FaRM used direct access and fast messaging for transactional shared memory [13], [12], while NAM-DB showed scalable distributed transactions on RDMA [49]. FaSST instead favored two-sided datagram RPC for simplicity and scale [27]. An IPDPS study of a hybrid DRAM/SSD key-value store further showed how non-blocking RDMA interfaces can overlap storage access with communication [42].

The best primitive varies by transaction phase. DrTM+H found that neither one-sided nor two-sided RDMA wins universally and combined them across concurrency-control phases [48]. This result cautions against treating RDMA as a drop-in faster socket. Memory registration, queue-pair scaling, remote key management, failure recovery, and protection-domain design are part of the database architecture. RDMA is most attractive for controlled, low-latency clusters with stable membership, in-memory state, and expert operations; TCP or user-space RPC is often preferable when portability, encryption, multi-tenancy, or WAN reach dominates.

5.2 Virtualized RDMA, SmartNICs, and Switches

Cloud deployment complicates direct hardware access. FreeFlow virtualizes RDMA for containers in software while preserving isolation, migration, and policy control [29], illustrating how an indirection layer can reconcile acceleration with orchestration. SmartNICs and DPUs can offload encryption, virtual switching, packet steering, and sometimes storage services. Programmable switches go further by processing coordination state at line rate. NetChain placed a replicated key-value coordination service in the network dataplane and achieved sub-round-trip operations [24]. Such approaches reduce latency but face constrained memory, limited arithmetic, update consistency, failure semantics, debugging difficulty, and tight coupling to hardware. A safe architecture keeps the authoritative recovery path in software and treats in-network state as a carefully verified acceleration layer.

6 Replication, Consensus, and Geography

6.1 Network Cost of Strong Consistency

Consensus protocols convert unreliable communication and failures into a stable replicated order. Raft organizes leader election, log replication, and membership change around a strong leader [39], while Paxos gives the foundational majority-agreement construction [32]. In the common case, a write reaches the leader, is transmitted to followers, and becomes committed after acknowledgment from a majority; client-visible latency therefore includes the leader path and a quorum round trip. Two-phase commit adds cross-shard messages and can block after coordinator failure; Paxos Commit relates atomic commit directly to consensus [19]. Calvin establishes a deterministic global order before execution [46], TAPIR moves ordering into the transaction protocol [50], and coordination-avoidance analysis identifies invariants that need no synchronous coordination [5].

Network-aware optimization should preserve the protocol's safety assumptions. Common techniques include placing leaders near writers, selecting geographically compact voting quorums while retaining distant non-voters, pipelining log entries, coalescing acknowledgments, follower or leaseholder reads, and routing transactions to data-local gateways. Hedged reads can reduce tail latency but consume capacity and complicate load. Failure detectors must be calibrated to network conditions: short election timeouts improve failover yet can destabilize a cluster during congestion or a regional pause.

Figure 2 illustrates the availability benefit of a larger majority group under a deliberately simplified independent-failure model. Real deployments may violate independence because replicas share racks, power, routing, or regions; failure-domain-aware placement is therefore as important as replica count.

6.2 Wide-Area Replication

Inter-region round-trip time is constrained by distance, so software cannot make synchronous global commits behave like local memory. Systems instead choose among local strong consistency, global strong consistency, asynchronous replication, and tunable quorums. Bigtable scales structured storage through tablet partitioning [7]; Cassandra combines Dynamo-style decentralization with a Bigtable-like model [31]; and Megastore confines synchronous wide-area transactions to fine-grained entity groups [6]. Dynamo emphasizes availability and eventual reconciliation [10]. Spanner provides synchronous replication and externally consistent transactions with Paxos and TrueTime [8]. F1 reports that cross-datacenter replication raises write latency [43], while CockroachDB uses geo-aware placement for resilient distributed SQL [45].

Precise clocks do not eliminate communication but can reduce coordination for leases, snapshot reads, and timestamp ordering

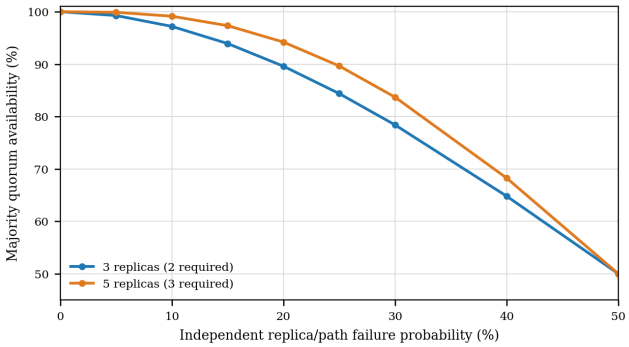

Figure 2: Majority quorum availability for three- and five-replica groups under independent, identical replica/path failure probability. Correlated failures are excluded.

Table 2: Technology selection summary

Context	Preferred substrate	Primary caution
General OLTP	TCP + pooled RPC + TLS	Tail retries and queueing
In-memory cluster	RDMA or kernel-bypass RPC	NIC state and recovery
Geo-distributed	TCP/QUIC + topology-aware quorum	RTT dominates commits
Multi-tenant cloud	VPC/VXLAN + mTLS	Jitter, MTU, noisy neighbors
Bulk repair	Paced TCP/multipath	Foreground interference
Coordination cache	RPC; optional switch offload	Limited dataplane semantics

when uncertainty is explicitly accounted for. Huygens showed that software techniques can synchronize clocks to tens of nanoseconds in a controlled data center by filtering queueing noise and exploiting pairwise consistency [17]. Across regions, databases must still tolerate larger and variable uncertainty. Clock APIs should expose bounds rather than pretend timestamps are exact, and correctness must survive clock-source failures.

6.3 Disaggregated and Cloud-Native Storage

Cloud-native databases increasingly separate stateless compute from replicated storage. This improves elasticity and failover but turns local storage operations into network traffic. Aurora reduces that traffic by sending redo records to a purpose-built, multi-tenant storage service rather than mirroring full pages [47]. The architecture demonstrates a general principle: protocol-level reduction of bytes and round trips may outperform merely increasing link speed. For distributed persistent memory, an SC paper proposed hardware-supported RDMA primitives that move remote durability operations into the platform while preserving an explicit persistence boundary [14]. Disaggregated designs should distinguish foreground log writes from background repair and compaction, reserve bandwidth for recovery, and avoid correlated overload when many compute nodes reconnect after a fault.

7 Discovery, Routing, and Control Planes

7.1 Service Discovery and Endpoint Selection

A database cluster is useful only if clients can locate healthy endpoints and route work to the correct shard or replica. DNS, configuration stores, membership services, virtual IPs, and client-side catalogs all solve parts of this problem. DNS is ubiquitous but its caching and time-to-live semantics can delay failover. Layer-4 load balancers hide topology from clients and can drain or replace backends, while database-aware drivers can route reads to followers, send writes to leaseholders, and avoid a second forwarding hop. The tradeoff is control-plane state: smarter clients need timely metadata and must behave safely when that metadata is stale.

Consistent hashing and range maps reduce remapping when nodes join or leave, but rebalancing still creates bulk network traffic and cache churn. Metadata should be versioned, signed or authenticated, and distributed independently of the data path so that a partial control-plane outage does not corrupt routing. Requests that reach the wrong replica should fail or redirect with an epoch number rather than execute under ambiguous ownership. Databases can also expose locality labels—region, zone, rack, and node class—to let drivers optimize latency without assuming that proximity implies authority.

7.2 Load Balancing and Connection Continuity

Database load balancing differs from stateless web balancing because connections may contain sessions, transactions, prepared statements, or replication streams. Per-packet spraying can reorder traffic, while per-flow hashing can strand a large flow on a congested path. Load balancers therefore need stable connection affinity, health-aware backend selection, and graceful drain behavior. Maglev combines ECMP, consistent hashing, and connection tracking at production scale [16]; Beamer preserves established connections during backend churn using state already present at servers [38]. Any database proxy must preserve client identity, propagate deadlines, avoid silently retrying non-idempotent transactions, and publish its own saturation metrics.

7.3 Membership and Out-of-Band Management

Membership, orchestration, and certificate services form a control plane whose failure can disable recovery even while the data plane is intact. Production designs should separate liveness from readiness, distinguish application overload from network unreachability, and avoid using a single in-band channel for both replication and emergency control. Administrative access, telemetry export, and fencing benefit from independent paths or at least reserved queues. The database should continue serving safely with cached membership during a temporary control-plane interruption, while refusing unsafe reconfiguration that cannot obtain a quorum of authoritative state.

8 Selection Method and Operational Guidance

Technology selection should start from measurable service objectives and failure assumptions rather than peak benchmark throughput. First classify traffic by message size, fan-out, read/write mix, locality, durability point, and burst behavior. Next identify where

latency is spent using end-to-end traces that correlate database stages with host and network metrics. Then test candidate stacks under failures, not only steady state: packet loss, queue buildup, link rerouting, NIC reset, certificate rotation, leader transfer, replica catch-up, and regional partition. Finally, evaluate p50, p99, and p99.9 latency together with CPU per operation, useful bytes per second, recovery time, and correctness.

For most deployments, a well-tuned TCP stack with pooled connections, compact RPC, batching, deadlines, TLS, and topology-aware placement is the strongest baseline. QUIC is attractive at mobile or Internet-facing edges and where stream multiplexing or path migration is valuable. Kernel bypass is justified when profiling shows the host stack consuming a material share of CPU or latency. RDMA is justified when the workload is dominated by small remote-memory operations in a controlled cluster and the team can operate lossless or near-lossless fabrics, memory registration, and NIC failure recovery. Programmable-network acceleration should be reserved for narrow, high-value functions with a software fallback.

Operationally, databases need observability at every layer. Useful signals include per-RPC deadline and retry counts, connection resets, retransmissions, ECN marks, queue occupancy, NIC drops, path changes, TLS handshakes, leader-to-follower RTT, quorum composition, and replica lag. Coordinated tracing is more informative than isolated dashboards because the same symptom may originate in a shard hot spot, CPU pause, virtual-switch policy, or fabric congestion. Capacity plans must reserve headroom for failure: a cluster that is healthy only when every link and replica is available will enter a recovery spiral after the first fault.

8.1 Benchmarking and Validation

A defensible evaluation separates open-loop offered load from closed-loop client behavior. Closed-loop tests can hide overload because clients wait longer and submit fewer requests; open-loop tests reveal queue growth but must record dropped or expired work. Benchmarks should preserve realistic key skew, transaction fan-out, message sizes, connection counts, and durability settings. Network microbenchmarks are useful for calibration, yet the final comparison must include the database protocol because a transport that wins a ping-pong test may lose after serialization, logging, contention, and replica recovery are included.

Failure testing should measure not only availability but the work created by recovery. A link cut may trigger leader elections, client reconnects, cache misses, snapshot transfer, and compaction simultaneously. Replaying these events under controlled fault injection reveals whether backpressure and bandwidth classes protect foreground traffic. Results should report confidence intervals and percentile latency over long enough windows to include periodic maintenance. Configuration, topology, NIC firmware, switch policy, and encryption mode are part of the experimental method and should be documented for reproducibility.

9 Open Challenges

Tail latency remains the central cross-layer challenge. Faster links shorten serialization time but do not remove queueing, incast, retransmission, CPU scheduling, or synchronized recovery bursts. Current research explores richer telemetry, pacing, multipath transport, and new congestion signals; CCC, for example, uses delay signals to approach telemetry-assisted congestion-control performance in high-speed clusters [23]. Production experience with DCTCP [11] and mixed TCP/RDMA buffer research [1] show that deployment complexity and traffic coexistence matter as much as isolated algorithm results. Databases could expose transaction criticality, deadline, or recovery class to the transport, but doing so safely requires preventing priority abuse and feedback instability.

A second challenge is making acceleration composable with multitenancy and security. RDMA's direct access conflicts with elastic migration, fine-grained policy, encrypted memory, and simple revocation; virtualized RDMA partially addresses these tensions [29]. SmartNIC and switch offloads distribute state into additional failure domains whose versions and telemetry must be managed. Formal verification, transactional control-plane updates, and portable programming models are needed before in-network database functions can become routine.

Third, wide-area systems need better cost-aware coordination. Data transfer is metered in many clouds, and carbon, sovereignty, and regional service constraints influence replica placement alongside latency. Adaptive quorums and learned routing may save time or cost, but reconfiguration itself must be safe and explainable. QUIC's user-space evolution offers an experimentation path, yet replay, encryption overhead, and interaction with database transactions require careful study.

Finally, observability must improve for encrypted and virtualized networks. Operators need causal evidence without weakening confidentiality: which hop or queue delayed a quorum, whether retransmission or application backpressure caused a timeout, and whether an overlay policy disagreed with the underlay. Privacy-preserving telemetry, cross-layer tracing standards, and deterministic fault injection would help connect database symptoms to network causes. The research opportunity is not simply a faster transport, but a dependable contract between database protocols, host stacks, NICs, virtual networks, and fabrics.

10 Conclusion

Distributed database performance and availability emerge from the interaction of network mechanisms with replication and transaction protocols. Ethernet, TCP, overlays, and TLS provide the portable operational foundation; ECN, topology-aware routing, batching, and kernel bypass improve that foundation; RDMA and programmable devices offer exceptional performance within narrower operational envelopes. Across regions, physical latency makes locality and consistency design more important than link speed. The most robust strategy is therefore cross-layer and evidence-driven: reduce messages and bytes, control queueing, place quorums deliberately, preserve explicit failure semantics, secure every channel, and adopt acceleration only where measurements justify its complexity.

References

[1] V. Addanki, W. Bai, S. Schmid, and M. Apostolaki. 2024. Reverie: Low Pass Filter-Based Switch Buffer Sharing for Datacenters with RDMA and TCP Traffic. Proc. USENIX NSDI, pp. 651–668.

[2] M. Al-Fares, A. Loukissas, and A. Vahdat. 2008. A Scalable, Commodity Data Center Network Architecture. Proc. ACM SIGCOMM, pp. 63–74. doi:10.1145/1402958.1402967

[3] M. Alizadeh et al. 2010. Data Center TCP (DCTCP). Proc. ACM SIGCOMM, pp. 63–74. doi:10.1145/1851275.1851192

[4] M. Alizadeh et al. 2013. pFabric: Minimal Near-Optimal Datacenter Transport. Proc. ACM SIGCOMM, pp. 435–446. doi:10.1145/2486001.2486031

[5] P. Bailis, A. Fekete, M. J. Franklin, A. Ghodsi, J. M. Hellerstein, and I. Stoica. 2014. Coordination Avoidance in Database Systems. Proc. VLDB Endow., vol. 8, no. 3, pp. 185–196. doi:10.14778/2735508.2735509

[6] J. Baker et al. 2011. Megastore: Providing Scalable, Highly Available Storage for Interactive Services. Proc. CIDR, pp. 223–234.

[7] F. Chang et al. 2006. Bigtable: A Distributed Storage System for Structured Data. Proc. USENIX OSDI, pp. 205–218.

[8] J. C. Corbett et al. 2012. Spanner: Google's Globally-Distributed Database. Proc. USENIX OSDI, pp. 251–264.

[9] M. Dalton et al. 2018. Andromeda: Performance, Isolation, and Velocity at Scale in Cloud Network Virtualization. Proc. USENIX NSDI, pp. 373–387.

[10] G. DeCandia et al. 2007. Dynamo: Amazon's Highly Available Key-Value Store. Proc. ACM SOSP, pp. 205–220.

[11] A. Dhamija et al. 2024. A Large-Scale Deployment of DCTCP. Proc. USENIX NSDI, pp. 201–220.

[12] A. Dragojević et al. 2015. No Compromises: Distributed Transactions with Consistency, Availability, and Performance. Proc. ACM SOSP, pp. 54–70. doi:10.1145/2815400.2815425

[13] A. Dragojević, D. Narayanan, M. Castro, and O. Hodson. 2014. FaRM: Fast Remote Memory. Proc. USENIX NSDI, pp. 401–414.

[14] Zhuohui Duan, Haodi Lu, Haikun Liu, Xiaofei Liao, Hai Jin, Yu Zhang, and Song Wu. 2021. Hardware-Supported Remote Persistence for Distributed Persistent Memory. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '21)*. Association for Computing Machinery, New York, NY, USA, Article 91, 14 pages. doi:10.1145/3458817.3476194

[15] W. Eddy and Ed. 2022. Transmission Control Protocol (TCP). RFC 9293, IETF, Aug.

[16] D. E. Eisenbud et al. 2016. Maglev: A Fast and Reliable Software Network Load Balancer. Proc. USENIX NSDI, pp. 523–535.

[17] Y. Geng et al. 2018. Exploiting a Natural Network Effect for Scalable, Fine-Grained Clock Synchronization. Proc. USENIX NSDI, pp. 81–94.

[18] S. Gilbert and N. Lynch. 2002. Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services. ACM SIGACT News, vol. 33, no. 2, pp. 51–59. doi:10.1145/564585.564601

[19] J. Gray and L. Lamport. 2006. Consensus on Transaction Commit. ACM Trans. Database Syst., vol. 31, no. 1, pp. 133–160. doi:10.1145/1132863.1132867

[20] A. Greenberg et al. 2009. VL2: A Scalable and Flexible Data Center Network. Proc. ACM SIGCOMM, pp. 51–62.

[21] IEEE. 2011. IEEE Standard for Local and Metropolitan Area Networks—Amendment 17: Priority-Based Flow Control. IEEE Std 802.1Qbb-2011. doi:10.1109/IEEESTD.2011.6032693

[22] J. Iyengar, M. Thomson, and Eds. 2021. QUIC: A UDP-Based Multiplexed and Secure Transport. RFC 9000, IETF, May.

[23] W. Jiang et al. 2026. CCC: Re-architecting Delay-Based Congestion Control in Datacenter Networks. Proc. USENIX NSDI, pp. 1553–1570.

[24] X. Jin et al. 2018. NetChain: Scale-Free Sub-RTT Coordination. Proc. USENIX NSDI, pp. 35–49.

[25] A. Kalia, M. Kaminsky, and D. Andersen. 2019. Datacenter RPCs Can Be General and Fast. Proc. USENIX NSDI, pp. 1–16.

[26] A. Kalia, M. Kaminsky, and D. G. Andersen. 2014. Using RDMA Efficiently for Key-Value Services. Proc. ACM SIGCOMM, pp. 295–306. doi:10.1145/2619239.2626299

[27] A. Kalia, M. Kaminsky, and D. G. Andersen. 2016. FaSST: Fast, Scalable and Simple Distributed Transactions with Two-Sided (RDMA) Datagram RPCs. Proc. USENIX OSDI, pp. 185–201.

[28] Yao Kang, Xin Wang, and Zhiling Lan. 2022. Study of workload interference with intelligent routing on dragonfly. In *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 1–14.

[29] D. Kim et al. 2019. FreeFlow: Software-Based Virtual RDMA Networking for Containerized Clouds. Proc. USENIX NSDI, pp. 113–126.

[30] G. Kumar et al. 2020. Swift: Delay Is Simple and Effective for Congestion Control in the Datacenter. Proc. ACM SIGCOMM, pp. 514–528.

[31] A. Lakshman and P. Malik. 2010. Cassandra: A Decentralized Structured Storage System. ACM SIGOPS Oper. Syst. Rev., vol. 44, no. 2, pp. 35–40. doi:10.1145/1773912.1773922

[32] L. Lamport. 2001. Paxos Made Simple. ACM SIGACT News, vol. 32, no. 4, pp. 51–58. doi:10.1145/568425.568433

[33] H. Lim, D. Han, D. G. Andersen, and M. Kaminsky. 2014. MICA: A Holistic Approach to Fast In-Memory Key-Value Storage. Proc. USENIX NSDI, pp. 429–444.

[34] Y. Lu et al. 2018. Multi-Path Transport for RDMA in Datacenters. Proc. USENIX NSDI, pp. 357–371.

[35] M. Mahalingam et al. 2014. Virtual eXtensible Local Area Network (VXLAN). RFC 7348, IETF, Aug.

[36] R. Mittal et al. 2015. TIMELY: RTT-Based Congestion Control for the Datacenter. Proc. ACM SIGCOMM, pp. 537–550.

[37] B. Montazeri, Y. Li, M. Alizadeh, and J. K. Ousterhout. 2018. Homa: A Receiver-Driven Low-Latency Transport Protocol Using Network Priorities. Proc. ACM SIGCOMM, pp. 221–235.

[38] V. Olteanu, A. Agache, A. Voinescu, and C. Raiciu. 2018. Stateless Datacenter Load-Balancing with Beamer. Proc. USENIX NSDI, pp. 125–139.

[39] D. Ongaro and J. Ousterhout. 2014. In Search of an Understandable Consensus Algorithm. USENIX ATC, pp. 305–319.

[40] R. Recio, B. Metzler, P. Culley, J. Hilland, and D. Garcia. 2007. A Remote Direct Memory Access Protocol Specification. RFC 5040, IETF, Oct. doi:10.17487/RFC5040

[41] E. Rescorla. 2026. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 9846, IETF.

[42] D. Shankar, X. Lu, N. S. Islam, M. Wasi-Ur-Rahman, and D. K. Panda. 2016. High-Performance Hybrid Key-Value Store on Modern Clusters with RDMA Interconnects and SSDs: Non-Blocking Extensions, Designs, and Benefits. In *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, Los Alamitos, CA, USA, 393–402. doi:10.1109/IPDPS.2016.112

[43] J. Shute et al. 2013. F1: A Distributed SQL Database That Scales. Proc. VLDB Endowment, vol. 6, no. 11, pp. 1068–1079.

[44] A. Singh et al. 2015. Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network. Proc. ACM SIGCOMM, pp. 183–197.

[45] R. Taft et al. 2020. CockroachDB: The Resilient Geo-Distributed SQL Database. Proc. ACM SIGMOD, pp. 1493–1509. doi:10.1145/3318464.3386134

[46] A. Thomson et al. 2012. Calvin: Fast Distributed Transactions for Partitioned Database Systems. Proc. ACM SIGMOD, pp. 1–12. doi:10.1145/2213836.2213838

[47] A. Verbitski et al. 2017. Amazon Aurora: Design Considerations for High Throughput Cloud-Native Relational Databases. Proc. ACM SIGMOD, pp. 1041–1052. doi:10.1145/3035918.3056101

[48] X. Wei, Z. Dong, R. Chen, and H. Chen. 2018. Deconstructing RDMA-Enabled Distributed Transactions: Hybrid Is Better! Proc. USENIX OSDI, pp. 233–251.

[49] E. Zamanian, C. Binnig, T. Kraska, and T. Harris. 2017. The End of a Myth: Distributed Transactions Can Scale. Proc. VLDB Endow., vol. 10, no. 6, pp. 685–696.

[50] I. Zhang, N. K. Sharma, A. Szekeres, A. Krishnamurthy, and D. R. K. Ports. 2015. Building Consistent Transactions with Inconsistent Replication. Proc. ACM SOSP, pp. 263–278. doi:10.1145/2815400.2815404

[51] Y. Zhu et al. 2015. Congestion Control for Large-Scale RDMA Deployments. Proc. ACM SIGCOMM, pp. 523–536.