GGN: Experiences in Designing and Deploying the Next-Generation Google Global Network

Mohammad Al-Fares, Richard Alimi, Arda Balkanay, Dennis Fetterly, Chi-Yao Hong
Nachikethas A. Jagadeesan, Bikash Koley, Priya Mahadevan, Subhasree Mandal, Warren Martins
Arjun Muralidharan, Namrata Pralhad Kadam, Aman Shaikh, Anees Shaikh, Rob Shakir
Arjun Singh, Sankalp Singh, Charith Wickramaarachchi, Min Zhu, Jonathan Zolla

Google
ggn-sigcomm@google.com

Abstract

Cloud and AI/ML workloads are posing unprecedented new requirements on the wide-area network: it must combine strict availability, massive growth, and feature agility. It became increasingly clear that traditional WAN designs were ill-equipped to adapt to these requirements.

We present Google's Global Network (GGN), a major architectural redesign of our WAN that evolves B2 and B4 into a single, modular, and highly available software-defined network. The architecture is designed around three pillars: (1) A modular design of functional domains with well-defined APIs; (2) a physically sharded and regionalized core for fault isolation and horizontal scaling; (3) a vendor-agnostic hardware strategy based on open standards. We share the multi-year deployment journey of GGN, including a safe, host-steered migration strategy, and demonstrate its ability to improve network availability and reaction time to failures, setting a foundation for a planet-scale modern WAN.

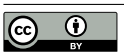
CCS Concepts

• **Networks** → **Wide area networks**; **Network reliability**; **Traffic engineering algorithms**.

Keywords

Wide-Area Networks, Network Reliability, Traffic Engineering, Software-defined Networking

ACM Reference Format:
Mohammad Al-Fares, Richard Alimi, Arda Balkanay, Dennis Fetterly, Chi-Yao Hong, Nachikethas A. Jagadeesan, Bikash Koley, Priya Mahadevan, Subhasree Mandal, Warren Martins, Arjun Muralidharan, Namrata Pralhad Kadam, Aman Shaikh, Anees Shaikh, Rob Shakir, and Arjun Singh, Sankalp Singh, Charith Wickramaarachchi, Min Zhu, Jonathan Zolla. 2026. GGN: Experiences in Designing and Deploying the Next-Generation Google Global Network. In *ACM SIGCOMM 2026 Conference (SIGCOMM '26), August 17–21, 2026, Denver, CO, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3789240.3829114



SIGCOMM '26, Denver, CO, USA

ACM ISBN 979-8-4007-2467-1/2026/08
https://doi.org/10.1145/3789240.3829114

1 Introduction

Global wide-area networks (WANs) operated by Cloud and content providers have experienced a fundamental shift in requirements over recent years. The growth of customer and enterprise traffic from Cloud services (relative to first-party services) has brought much stricter reliability and performance expectations, and the explosion of large-scale AI/ML training workloads introduced unprecedented demand to scale the network to deliver more capacity. These requirements are significantly different from the previous era of Internet-delivered services and media streaming, in which WANs were designed for global reach and low latency to users, and efficient inter-datacenter transfer of backend traffic.

For over a decade, Google's private WAN infrastructure was architected as two WANs: B2, a traditional carrier-grade IP/MPLS network [13], handling user Internet traffic, and B4, a global software-defined WAN (SDN) [17], carrying large volumes of inter-datacenter traffic. Dual WANs provided a way to separately optimize for key properties such as high availability and performance vs. massive capacity with efficient cost and utilization. Other hyperscalers subsequently also deployed split WAN architectures with similar motivations and designs [10, 20].

The dual backbone architecture enabled significant growth in external user traffic, and cost efficiency for even greater growth of inter-datacenter traffic.[1] However, it also introduced challenges such as redundant connectivity and feature development costs, stranding due to non-fungible capacity, and the single point of failure associated with a global control plane. Other hyperscalers have reported similar problems with operating two disparate WANs [20]. It is clear that the split WAN approach is no longer able to meet the requirements of the modern WAN. Cloud customers, and the extreme scale of ML workloads extending beyond a single datacenter, demand a network architecture that inherently provides smaller failure domains and near-zero tolerance for multi-region failures, and the ability to continually grow capacity, maintain low latency, and allow for fast technology evolution. In our case, these challenges motivated a reexamination of Google's dual WAN strategy and a clean-slate design for a unified, global network architecture.

This paper introduces **Google's Global Network (GGN)**, which evolves B2 and B4 into a single, cohesive SDN backbone. GGN is architected to address the diverse requirements stemming from Cloud, AI, and our first-party services, with four primary goals:

[1] As of July 2026, Google's network spans 43 regions and 200+ countries and territories.

- **Eliminate Global Failures:** Ensure that a failure's blast radius is inherently limited to a single region,[2] while continuing to deliver all high-priority traffic.
- **Massive Scalability:** Support rapid traffic growth with a scalable network architecture, beyond what can be achieved by traditional vertical hardware scaling alone.
- **Faster Network Evolution:** Enable agile introduction of new hardware platforms and network capabilities to meet changing requirements.
- **Well-defined SLOs:** Provide customers with clear, enforceable Service Level Objectives (SLOs) for packet loss, bandwidth, and latency.

GGN achieves these goals through three architectural pillars: (i) a physically-sharded and regionalized core enabling fault isolation and horizontal capacity scaling, (ii) a modular network architecture allowing independent evolution of each functional domain, and (iii) a vendor-independent hardware strategy based on open standards to accelerate integration with new technology platforms.

The main contributions we present in this paper are:

- The architecture of GGN, a blueprint for building a highly available, planet-scale WAN based on principles of modularity and hardware optionality (§3).
- The design, implementation, and deployment of GGN's reliability mechanisms: sharding for containing global failures (§4), and regionalization for isolating regional failures (§5).
- A technical strategy and multi-year experience for safe migration from our existing WANs to GGN using host-based traffic steering (§6).
- Key observations and lessons learned from real-life evaluation of GGN's performance and resilience (§7).

We believe the experience in designing GGN and evolving to a unified backbone has applicability beyond our specific use case. The results from deploying GGN, and observing it reliably carrying user traffic during major real-world failures, demonstrate the effectiveness of sharding and regionalization for achieving high availability in global backbones. Our approach for safely migrating production traffic from one network architecture to another can also be a recipe for similar evolution in other WANs.

In the appendices we describe: a suite of novel control plane optimizations developed for GGN to enhance scalability, availability, and efficiency (§A), a historical audit of network outages and how the GGN architecture would have addressed them (§B), a sharding granularity efficiency analysis (§C), and finally, our technology introduction velocity and vendor qualification strategy (§D).

2 Background and Motivation

The decision to re-architect our WAN was driven by the growing limitations of the legacy split-WAN model and a fundamental shift in customer and product requirements.

2.1 Legacy Architecture Limitations

The B2 and B4 networks, while successful in their own right, created significant challenges when operated in parallel.

[2]A *region* is defined as an independent geographical failure domain backed by Cloud service agreements governing availability [5].

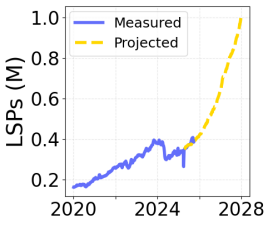


Figure 1: B2 LSPs. Growth is straining MPLS-TE scale.

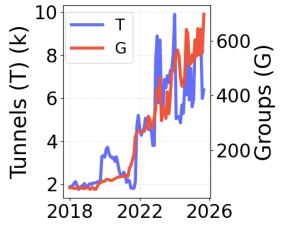


Figure 2: B4 Tunnels. Growth is straining HW table size limits.

Operational Overhead and Divergence. Each network had its own control plane, management stack, and operational procedures. B2 relied on distributed protocols like IS-IS and RSVP-TE on vendor hardware, while B4 used SDN controllers [11] on custom hardware. Adding new features (e.g., IPv6 support, new QoS classes) required separate, often substantially different, development and deployment efforts, doubling the engineering cost and slowing down velocity.

Scaling and Reliability Challenges. Both backbones faced scaling limitations: The expansion of the global network exposed scalability limitations in B2's distributed MPLS-TE control plane (Fig 1). B4's centralized TE offered superior scalability, but created a massive failure domain: a bug in the global controller could impact traffic worldwide [13]. B4's growth is also straining the underlying hardware's table limits (Fig. 2). B4's architecture also relied on vertical scaling—upgrading to larger, more powerful hardware (e.g., from 100GE to 400GE merchant silicon switch chips), and horizontal scaling—expanding a site by adding supernodes.[3] By themselves, these two approaches are not able to meet the massive capacity growth requirements of Cloud and AI/ML.

Lack of Capacity Fungibility. A critical limitation was the inability to share capacity between B2 and B4. During a subsea cable fault or any other large-scale event, one network could be heavily congested while the other had spare capacity. Dynamically shifting traffic between networks is constrained by fundamental architectural mismatches, specifically the incompatibility of supported traffic types, the misalignment of connected edges, and the disparity in offered latencies. This led to inefficient resource utilization and a lower overall network resilience.

2.2 Evolving Customer Requirements

Our network evolved from serving primarily internal Google services to a vast ecosystem of Google Cloud enterprise customers and large-scale AI/ML workloads.

Predictability and Control. Cloud customers treat our network as an extension of their own. They demand predictable performance with clearly defined SLOs for bandwidth, latency, and availability, akin to traditional MPLS/E-LINE/pseudo-wire services. They require control over their traffic's geographic path for data sovereignty and compliance, and increasingly need advanced features like on-demand bandwidth and wire-speed encryption. The black-box nature of B2 is ill-suited for these requirements.

[3]A *site* is the basic network fabric unit in B4's architecture, now adopted more widely in GGN. A *supernode* is the basic abstraction of an SDN domain, and used as a building block of a site [14].

Higher Availability and Smaller Blast Radius. Unlike many internal Google applications designed for resiliency with global load balancing, Cloud customers are highly sensitive to network disruptions. An outage affecting multiple regions is unacceptable. The global failure domain of B4's control plane posed a direct business risk. The need to drastically reduce the "blast radius" of any failure became a primary architectural driver.

Velocity and Competition. The Cloud/AI market is intensely competitive. The ability to rapidly deploy capacity, turn up new regions, and deliver new features is a critical competitive differentiator. The operational complexity and duplicated engineering effort of the B2/B4 model hindered our development and deployment velocity. Unifying the architecture was essential for agility.

3 GGN Architecture

GGN's architecture is a fundamental departure from the monolithic designs of B2 and B4. It is founded on the principles of modularity, isolation, and standardization to achieve our goals of dependability, scalability, and velocity. The network is decomposed into a set of independent *functional domains* with constrained, well-defined APIs at their boundaries. This design allows each domain to evolve independently, containing failures within its boundary and enabling parallel development.

The primary functional domains, illustrated in Fig. 3, are:

- **Core Domains (Global and Regional):** Responsible for transiting traffic between edge domains. The Global Core connects all regional cores, while a Regional Core connects edges within a single region.
- **Edge Domains (Compute, Customer Service, Internet):** Responsible for originating and terminating traffic. The Compute Edge (e.g., a Jupiter cluster fabric [25]), Customer Service Edge (CSE, for Cloud Interconnect), and Internet Edge (IE, for peering) each serve distinct purposes and connect to the Core domains.

The critical boundary between these domains is the **Edge-Core API**. This is not a single technology but a set of principles and protocols governing interaction. It enforces strict isolation, ensuring an edge domain cannot destabilize the core or other edges. The API consists of two main parts: a *routing API* (based on eBGP) for exchanging minimal reachability information, and a *bandwidth API* (based on BwE[4]) for communicating capacity allocations from the core to the edges. This narrow, robust API is key to enabling modularity and safe, independent evolution of each domain.

3.1 A Scalable and Reliable Core

Given the vast volume of traffic transiting the core domains, a failure within them has the largest potential blast radius. To mitigate this, GGN's core is architected with two primary reliability features: *sharding* and *regionalization*.

Core Sharding. Rather than partitioning the existing B4 network, we implement sharding by deploying three *new* parallel, independent global networks, with B4 defined as the first shard.[5]

[4] *BwE* is our Bandwidth Enforcement system at the source hosts, with dynamic throttling based on service priority and network conditions [21].
[5] The design supports adding more shards as needed. B2 is slated for decommissioning once the GGN migration is complete.

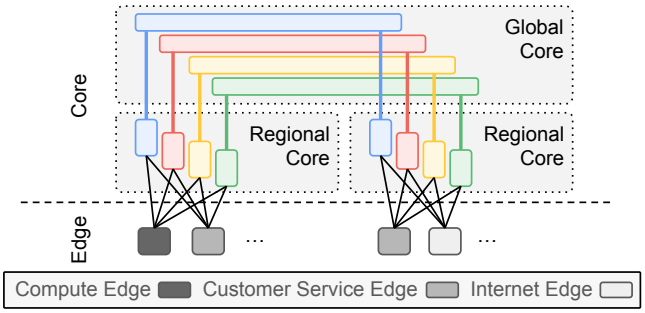


Figure 3: High-level GGN Architecture. The Edge-Core API enforces domain isolation. Edge domains connect to a sharded core.

These networks are parallel and physically-disjoint at L2 and above, with independent data and control planes. Traffic is spread across all available shards, providing $N+1$ redundancy against a full-shard failure, in conjunction with failure planning as described in §4.3.4. This is our primary defense against global outages.

Regionalization. Regionalization partitions the network topology along geographical lines. Each region operates its own independent TE and BwE SDN controllers, limiting the impact of control plane bugs or misconfigurations to traffic within that single region.

These two concepts are the cornerstone of GGN's dependability, and discussed in detail in §4 and §5. The core itself is an evolution of B4's SDN principles, as its centralized TE and host-integrated bandwidth enforcement provided the ideal foundation for building the sophisticated traffic steering required for sharding.

3.2 Hardware Optionality

A fundamental pillar of GGN is moving from bespoke, in-house switching hardware (like B4's Saturn/Stargate) toward a multi-vendor ecosystem based on commodity "lean routers." This strategy, enabled by what we call the **WAN Building Block (WBB),** offers several advantages:

- **Resilience:** A diversified strategy protects against single-vendor bugs, exploits, or supply-chain disruptions.
- **Velocity:** We can leverage vendor innovation and new hardware capabilities (e.g., higher port speeds, new features) much faster than designing our own.
- **Focus:** Google engineers can concentrate on our key differentiators in software (the control and management planes), rather than on hardware development.

WBB makes this possible along two dimensions:

(1) Standardized Device Management and Control: WBB emphasizes a shift towards standardized, open APIs for device management and control. We rely heavily on a suite of gRPC-based services: **gNMI** for telemetry and configuration (e.g., OpenConfig models), **gNOI** for operational commands (e.g., reboots, link-qualification), and **gRIBI** for programming the forwarding plane. These APIs are a core part of the OpenConfig standard [4].

(2) Standardized Hardware Catalog: WBB defines a Hardware Catalog with multiple *Abstract SKUs*. An abstract SKU is essentially a device *class* with a defined envelope of required features and

characteristics (e.g., required set of forwarding capabilities, minimum forwarding throughput, maximum power draw, etc.). For each SKU, networking vendors would offer actual devices that fit within that class, and such devices are considered interchangeable from our deployment and operational perspectives. We tailor SKUs to sites with different capacity requirements (e.g., WBB-XL SKU with 16-slot modular form-factor switches for the largest data centers, fixed form-factor WBB-S for peering sites, etc.).

The primary motivation for optionality is network reliability: We define a deployment policy such that no single vendor bug may impact all shards. In addition, such optionality allows us to be more resilient to vendors' supply-chain disruptions or inability to fulfill material volume. This is increasingly critical given the forecasts for GGN's growth, driven in large part by ML/AI workloads. This combination of commodity hardware and open, programmatic interfaces is crucial for achieving the agility GGN requires.

3.3 Priority Classes and SLOs

In GGN, traffic is routed and protected according to traffic priority classes. To better motivate some of the following discussion, we describe the different traffic priority classes and their target availability SLO dimensions (Table 1). We define a flow as a ⟨*src endpoint*, *dst endpoint*, *traffic class*⟩ tuple. The higher the flow's traffic class, the tighter the SLO guarantee. For each flow, we define SLO targets for loss (connectivity), bandwidth and latency, and our computations are performed at 1-minute granularity. A minute is deemed a "bad minute" if *any* of the three targets for packet loss, bandwidth, and latency has not been met.[6]

Table 1: Traffic Priority Classes and Availability SLOs.

Priority Class	Traffic Description	SLO (%)
Tier 0 / User-facing	Highest priority, latency-sensitive. Critical user and infrastructure traffic.	99.99
Tier 1	Minimal loss-tolerating internal traffic.	99.9
Tier 2	High-bandwidth, latency-insensitive traffic.	99
Tier 3	Bulk transfers, copy traffic.	50

(a) Traffic Priority Classes

Dimension	Criterion
Packet Loss	Loss is under a defined loss-SLO for that flow's tier (dependent on factors like locality, etc.).
Bandwidth	Delivered flow packets fulfill approved demand profile.
Latency	Within tiered multiplicative factor of baseline shortest-path (lower-priority tiers are more latency-tolerant).

(b) SLO Components. All three must be met in any given minute.

We use both active and passive measurements for availability: active UDP probes measure availability at the IP layer (L3), and RPC request-responses measure the application layer (L7). L7 availability serves as a good proxy for application performance. We measure both L3 and L7, as they can differ due to "gray failures." [15]

[6]Note that this is stricter than the industry standard; operators like Microsoft OneWAN [20] and Meta's EBB [16] define this for connectivity/loss only.

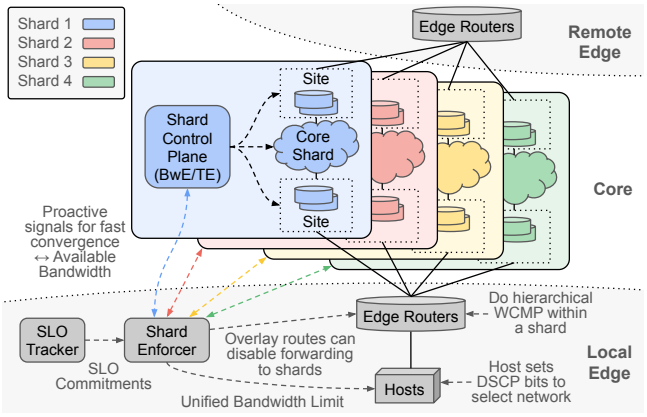


Figure 4: Core Sharding Architecture. Edge domains connect to all shards. Traffic is split at the source fabric (Local Edge), and recombined at the destination (Remote Edge).

4 Core Sharding

The most significant architectural change in GGN is the sharding of the core network; essentially creating parallel, independent global networks. The primary motivation is to eliminate the possibility of a single control plane failure causing a global outage, a known risk with B4's monolithic design [1, 13]. In addition to reliability benefits, sharding also provides a path for horizontal scaling of the WAN, allowing deployment of more network capacity without increasing the scaling requirements for components such as our global TE server or domain controller.

Each shard operates as an independent, full-fledged network with its own data plane (routers, links) and control plane (TE, BwE, routing). Shards are disjoint at L2 and above, with no inter-core interactions. Edge domains connect to all shards (Fig. 4) via independent L1 links, and edge routers are responsible for traffic distribution across them. Connecting each edge domain to all shards ensures that any endpoint pair meets our reliability target of N-way redundant control planes (where $N = 4$ in our current deployment), and simplifies capacity allocation of the Edge-to-Core interconnects.

4.1 Cross-Shard SDN Control

For cross-shard traffic distribution and failure handling, the system relies on the newly-introduced **Shard Enforcer (SE)**, a policy-based SDN component within the BwE control plane of each edge domain. The SE maintains a unified view of available bandwidth across all shards, interacting with per-shard control planes to determine shard reachability based on the local BGP state within the source edge domain. It translates this policy intent into iBGP advertisements to program edge routers—which allows disabling of specific shards at a destination-cluster granularity during outages—while simultaneously informing host-side BwE subsystems to enforce corresponding limits. This architecture augments traditional host-based enforcement with cross-shard traffic steering at network edge routers, ensuring that traffic distribution adheres to global policy while enabling the fast local reaction times provided by BGP (§4.2). Control plane telemetry-based reachability determination has known limitations; specifically, it cannot detect data

plane outages under certain failure modes. Consequently, we are actively evaluating augmenting this with data plane telemetry to improve detection accuracy.

SE design separates *intended* shard weights from *programmed* shard weights. In general, the intended weights are used by the SDN control plane, including global BwE and TE, to perform allocation within the shard, and programmed weights are sent to the edge routers via iBGP to modify the split of traffic across the individual shards. The following examples show the benefits of an SDN-based design for controlling the failure of individual shards.

Proactive, Policy-Aware Convergence. By including a controller that is deeply aware of our network's policies and offered SLOs and can integrate with the shard control planes, we can design a system that makes more appropriate and detailed tradeoffs which would not be feasible to express in traditional BGP routing policies. E.g., global SDN systems have some inherent convergence delay. The addition of an SDN controller allows us to proactively inform the per-shard control planes of the upcoming change in demands *before* actuating the change in routing, avoiding introducing unnecessary transient congestion or throttling.

Our two most critical traffic classes (Table 1), namely Tier-0 and Tier-1, have very small outage budgets (~4 and ~40 mins monthly, respectively). Less time-sensitive workloads can sustain several hours of downtime monthly. Given the complexity of global outages, which may take hours to resolve, the first two classes must both fit on remaining healthy shards in a failure. Other classes do not need such protection, though still benefit by opportunistically receiving a portion of unused capacity on remaining healthy shards.

For the first two classes, our design allows us to optimize for partial failures. We optimize for immediate failover during outages impacting out highest-priority Tier-0 traffic, possibly requiring other shards to converge after the traffic is moved. However, for traffic with a larger outage budget, we can inform the other shards of an upcoming change to demand, resulting in proactively creating additional tunnels via TE. For partial outages impacting only our lowest priority traffic, we can opt to not fail over traffic at all, preferring to simply throttle the traffic until the outage is resolved.

Transient Behavior. The design allows the system to avoid transient throttling during failure convergence. BwE operates on a logical model of the network forwarding state and traffic demands to determine the appropriate bandwidth limit to apply for a set of network flows. Without a controller like SE, BwE would observe the transient state where a single shard has severely reduced capacity, but is equal in forwarding weight to the other healthy shards. This would result in significant traffic throttling until the updated network state, which excludes the bad shard from the flow's path, is observed by BwE. It's important to avoid throttling in this state, as one possible failure mode for a system such as BwE is creating *incorrectly low* limits which result in unnecessary throttling, while in reality there is sufficient physical capacity for the traffic. In production, we have observed this behavior stem from control plane or telemetry bugs, as well as misconfigurations. SE mitigates this by calculating limits based strictly on the flow's final, post-failover allocation, while the edge routers are being programmed with the updated state.

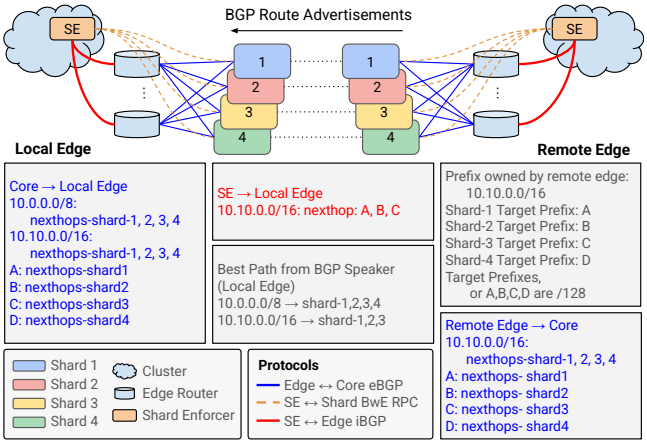


Figure 5: Routing between Edge, Shards and SE. In this example, SE disables shard-4 for `10.10.0.0/16` cluster-prefix using target-prefix-based indirect advertisements.

Blast Radius Control In addition to having a per edge-router deployment of SE along with associated progressive roll-out policies with backstops, the proactive signal of demand changes to the per-shard SDN controllers allow us to confirm that the remaining shards can accommodate a shard failure and that post-failure allocation will improve SLO compliance with respect to the current state before removing the shard from programmed routes, even if a large number of SE instances attempt to fail a shard at the same time. Additionally, we align the SE with capacity planning policies (§4.3.4) and only allow a single shard to be failed for a flow. Subsequent loss of connectivity on a shard can be handled via underlay routing. Multi-layer input and output validations are added against automated multi-shard fail over by SE (protection against unplanned cascading SE actions).

4.2 Edge Routing

Each edge router receives reachability information for remote edges from the sharded WAN. This is achieved by assigning each remote edge a shard-specific address, called a *target prefix*. This design is illustrated in Fig. 5, where A, B, C, and D are examples of remote edge target prefixes. An edge connected to four shards receives a total of four target prefixes. The prefixes are advertised by the originating edge over each WAN shard and propagated to all other edges via BGP. In short, these target prefixes represent the reachability for a given edge over a given shard.

Building on this base reachability signal, the SE signals which shards to use for traffic egressing from the local edge routers toward the remote edges. This information is signaled using BGP, where SE translates its intent into BGP advertisements. SE has a global view of all shards and can make better decisions on shard selection for a particular source and destination pair in response to a partial failure. For a complete loss of reachability, in-band mechanisms like PRR will act quickly, and BGP routing will converge to ensure the underlay routes around this blackhole. The converged BGP routes will then signal the reachability loss to SE, which can make a more informed decision (e.g., to re-enable a shard that had been removed

previously due to partial failure). Therefore, primary reachability for remote edges is still propagated via BGP.

SE uses BGP to announce remote machine prefixes (owned by edge clusters) to the local edge routers. In Fig. 5, `10.10.0.0/16` is the machine prefix range owned by the remote edge. In these announcements, the corresponding target prefixes are used as the BGP next-hops. Both machine and target prefixes provide reachability information for the remote edge. However, they are distinct: machine prefixes are end-host addresses, whereas target prefixes are virtual addresses used as next-hops by SE to achieve shard selection. A remote edge can have one or more machine prefix ranges but only one target prefix per shard. Upon receiving these BGP advertisements from SE, the local edge routers perform BGP recursive resolution on the target prefix next-hop to find the physical next-hops required to reach the remote edges. These physical next-hops point to an SE-selected shard. As shown in Fig. 5, SE advertises A, B, and C to the local edge as next-hops for the prefix owned by the remote edge (`10.10.0.0/16`). BGP then recursively resolves next-hops A, B, and C to the trunks of Shards 1, 2, and 3.

When a remote edge becomes unreachable over a specific shard due to BGP route withdrawals, BGP reacts within seconds by withdrawing advertisements for that remote edge's target prefixes from the local edge. BGP route withdrawals can occur due to a BGP session failure, a bug in the remote-side configuration, or a control plane issue. BGP's recursive resolution then eliminates the shard because the next-hops are no longer resolvable once they have been withdrawn from BGP advertisements—even if the SE is still signaling to use all shards. The SE then converges to the new reachability information once it receives the updated signal.

If SE decides to turn off a shard due to congestion or planned network maintenance, it advertises the reachability for a remote edge with next-hops as the target prefixes assigned to all shards except for the shard experiencing the failure. BGP recursive resolution then results in physical next-hops that point to the healthy shards. SE also provides its intent for per-shard traffic distribution using BGP link bandwidth. This link bandwidth is attached to target prefixes when advertised by SE. The edge router uses hierarchical WCMP (Weighted Cost Multipath [30]) to apply this intent at the physical trunk to achieve the traffic distribution intended by BwE. BGP reaction to SE's intent is on the order of seconds.

The sharded WAN advertises remote machine prefixes–assigned to all edges–along with target prefixes to the local edge routers. These serve as backup routes, utilized only when the SE connection to the edge routers is down. Our Edge Routing SDN controllers [11] also provide defense in depth for SE. Input validation checks, such as confirming a sufficient number of next hops, are applied to incoming route advertisements from SE, which can be rejected if determined to be unsafe. Furthermore, BGP policies at the edge routers are hardened to avoid any unexpected advertisements.

4.3 Design Decisions and Tradeoffs

The GGN architecture transforms potentially catastrophic global failures into manageable, contained events with limited and SLO-budgeted impact. The following motivated this architecture.

4.3.1 Isolation Between Shards. We evaluated the tradeoffs for several choices for where to isolate our network shards. We considered a hierarchy of components, where isolating components lower in the hierarchy also implies isolating those above: (1) global and regional SDN control plane (e.g., BwE, TE), (2) L2 links, (3) routers, and (4) L0/L1 fiber infrastructure and line systems.

Our decision was to strictly isolate through the router layer and above and opportunistically isolate L0/L1 infrastructure, both for the internal Core topology and Edge-to-Core interconnects.[7] This results in a design with separate, parallel networks and L2+ components, and a separate control plane for each shard.

We considered several tradeoffs with this decision:

Isolation from Physical and Software Failures. Events such as a fiber-cut, switch failure, or bad switch software rollout result in a common input (e.g., topology) change to each of the logical controllers. If such an event triggers a previously unknown bug in the global controller, all shards' control planes could be impacted concurrently. As a result, we want to isolate as much physical infrastructure and software as possible. This drove us to the decision of strictly isolating through at least the router level, as we wanted to ensure that software rollouts and maintenance events on individual routers could impact only a single shard.

Improved Control Plane Isolation. We also considered sharing L3 links between shards. However, this precludes strict isolation of bandwidth limits. E.g., if one logical instance of BwE were to generate limits that exceed its logical slice of a physical link, the physical link will experience congestion, creating loss for all logical shards. With isolated links, the congestion will be limited to flows on the impacted shard. If that congestion is severe enough to cause significant packet loss, traffic can be removed from the shard, isolating the remaining BwE instances from the failed one.

Control Plane Scalability. By isolating routers and links, our design can spread the same amount of capacity (devices, links) across independent control/routing domains, allowing our global controllers to see a smaller topology scale for the same amount of physical capacity.

4.3.2 SDN Controller vs. BGP-Based Traffic Distribution. At the edge, we employ a layered architecture for traffic distribution and reliability. The BGP layer ensures reachability, handles shard failover, and provides path diversity for Protective ReRoute (PRR) [27].[8] Due to its rapid sub-second response, PRR is a primary mechanism for protecting against shard failures. Unlike traditional BGP-only multi-shard networks, such as Meta's EBB [10], our introduction of local SDN control system (§4.1) to optimize cross-shard load balancing offers several advantages over conventional solutions:

Policy-Aware Failover. Our controller can be aware of fine-grained policies that are infeasible to represent in routing protocols like BGP. E.g., we can communicate not just available bandwidth, but evaluate whether the impacted traffic is at risk of not meeting our advertised SLOs, avoiding unnecessary shard failover if so. Similarly, switch hardware resource usage may increase as we handle combinations of shard failures for different destination edges. The sharding controller is aware of these limitations, and can use available resources more effectively under a large number of failures.

[7] For some elements, such as sub-sea cables, the cost of complete physical isolation may be too high to warrant the risk reduction gained.

[8] PRR is an end-host mechanism that leverages path diversity, responding within a few RTTs to a flow's failure signals by updating its FlowLabel. Intermediate switches then re-hash onto a path that avoids the outage.

Safe and Proactive Global Convergence. The global routing systems within each shard are complex systems that can require time to converge for large shifts in demands. By integrating an SDN controller for shard failover, we can proactively signal upcoming demand changes to each per-shard control plane, allowing that shard to converge (e.g., create additional tunnels, adjust bandwidth allocations) before committing to traffic shifts via shard failover. Having a controller to mediate the signal from each shard also allows us to select a better result even when multiple shards may be degraded. E.g., if one shard has already failed, we can opt to ignore a signal indicating reduced available bandwidth on a second shard, reducing the likelihood of cascading failures across shards.

4.3.3 Host- vs. Network-Based Traffic Steering. Edge border devices (e.g., the Fabric Border Router (FBR) for Compute Edge) decide how to split traffic across shards. This network-centric approach is a key design choice. To differentiate traffic destined for shard 1 (B4) from traffic eligible for the other shards, we utilize bits in a reserved header field, which originating hosts mark accordingly. The FBRs maintain separate VRFs for each: forwarding unsharded traffic only to shard 1, and sharded traffic across the other shards. For the latter, this is split using equal-cost multipath (ECMP) by default, but it is controlled by SE (§4.1), which can adjust the weights to steer traffic away from unhealthy or congested shards.

Note that host-based marking is not strictly required for adopting the sharded GGN architecture (e.g., for network operators where such marking may not be possible). In Google's case, it was a useful mechanism for controlled, gradual migration where one of the shards already exists.

4.3.4 Safety Policies. The operational independence of shards enables a much safer change management process. Software and configuration changes are rolled out sequentially—one shard at a time—with a mandatory soak period for health verification before proceeding to the next shard. This policy, enforced by our production safety systems such as CAPA [22], ensures that a faulty change can impact at most one shard, preventing it from causing a multi-region or global incident. This "one shard at a time" principle is a cornerstone of GGN's operational reliability.

Planning for Failure. The core is planned for $N - 1$ shard availability. For traffic requiring protection, capacity is provisioned such that if one shard fails, the remaining shards can absorb its traffic without violating SLOs. E.g., with four shards, a 100Tbps demand would be met by provisioning 33.3Tbps on each shard. This introduces a planned overbuild, but this "spare" capacity is used to carry lower-priority, best-effort traffic that can tolerate being dropped during a failure, thus maximizing overall efficiency. By implementing strict network safety policies that disallow concurrent operations across shards, we can plan for failure of just a single global shard to improve efficiency.

Reacting to Shard Failure. When a shard fails, a multi-layered response ensures graceful degradation:

(1) **Sub-second Data Plane Reaction:** At the ingress, local health checks or mechanisms (e.g., PRR) can rapidly detect a black hole on one path (shard) and re-hash affected flows to the remaining healthy paths.
(2) **Seconds-Scale Edge Routing Reaction:** Reachability is communicated via BGP between edges across each shard. If there is no routing reachability for a remote edge over a given shard, the local edge routers will deprefer that shard without needing a response from higher-level components like BwE. This is a faster response for larger-scale outages than the convergence delay of higher-level components.
(3) **Minutes-Scale SDN Control Plane:** At a slower time-scale, a newly-introduced SDN component detects a loss of reachability, and signals the global BwE and TE controllers in the remaining healthy shards ($\approx$ 2 min). These controllers observe the increased traffic demand and compute a new, globally optimal allocation of paths and bandwidth. At this timescale, the system can also address more subtle failures, such as partial outages that reduce available bandwidth. These failures manifest as congestion within a shard rather than as traffic black holes or routing reachability issues. The SDN component can execute shard failover to mitigate bandwidth availability in an SLO-aware manner. In the event of a partial outage, the reaction time is $\approx$ 3 min for Tier-0 traffic. Conversely, for Tier-1 traffic, the distributed SDN component staggers the failover over a $\approx$ 5–15 min window.

Edge Network Limitations Our SDN controller uses BGP to communicate its intent to edge routers; however, BGP comes with some complexity, as it requires extra features to express various SDN intents. Currently, the SDN controller does not express intent per QoS class due to the lack of BGP multi-VRF support at the edge routers. We mitigate this issue by ensuring that our network is capacity planned to absorb shard failure for traffic classes based on their SLO, and by leveraging per-QoS host enforcement in BwE.

The SDN controller also has limitations on the granularity of the intent due to hardware table space constraints at the local edge. E.g., to avoid creating a large number of WCMP groups at the local edge, the SDN controller can only turn off one shard at a time, and can only program a unique state for 50% of the remote edges. For practical purposes this is sufficient, as concurrent failures across multiple shards are rare and we expect shard failure to either impact most prefixes from the perspective of a local edge (e.g., global outage, local Edge-to-Core failure), or be limited to a smaller number of remote edges (e.g., remote Edge-to-Core or remote region failure).

The SE also allows us to intelligently mitigate both of these limitations, e.g., via SLO-aware failover or selecting the most critical destinations if we have an unexpected number of shard failures.

5 Regionalization

While sharding protects against cross-shard failures, regionalization limits the geographical blast radius of an outage. The core principle is **region isolation**: each region has its independent control plane. This ensures that failures outside a region, whether from a control system malfunction or bad inputs irrelevant to the region, do not impact traffic within the region. This is particularly important to Google's Cloud customers as it reduces the risk of regionally-correlated failures of their compute and storage. Regionalization also reduces the scaling pressure on the global core, significantly reducing the number of nodes the global core must manage.

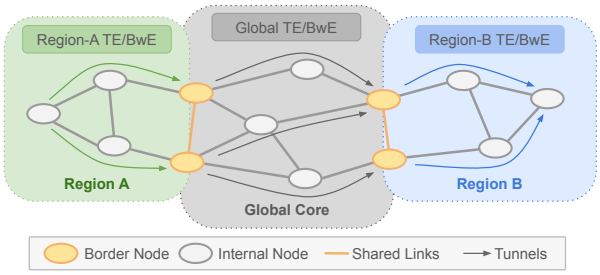


Figure 6: Regionalization Architecture. Regional and global cores share nodes at the border, and links between border nodes are shared and logically-carved. Inter-region flow pathing is co-owned by regional and global controllers.

5.1 Design

Regionalization partitions the network across geographical lines. Each region is sharded as described in §4, and each sharded region has its own TE and BwE controllers. These controllers have visibility into, and control over, the topology and traffic only within their own region (Fig. 6).

Pathing and bandwidth allocation for intra-region traffic (e.g., between two clusters in the same region) is wholly controlled by the regional TE/BwE controllers. The pathing of inter-region traffic is co-owned by the regional and global core controllers. Each core TE controller controls the pathing of this traffic within the slice of the network it observes and controls. Similarly, the regional and global core BwE controllers determine bandwidth allocations for the inter-region flow on the slice of the network it controls, which are then merged together at the host to determine the overall bandwidth allocation.

5.1.1 Network Design and Inter-Region Controller Architecture. To get the desired isolation properties, we must minimize the dependency between the regional and global controllers. This motivates our design of a hard border separating the regional and global cores: the topology is organized in a hierarchical tree structure, where the regional cores (leaves) hang off the global core (root).

We considered two alternate designs which were both rejected as they failed to provide the necessary isolation guarantees: **(1) Two-layered Controllers.** Here, a lower layer intra-region controller manages traffic within the region, and an upper-layer inter-region controller operates on the abstract region-level graph and produces inter-region pathing. The intra-region controller receives pathing intent from the upper layer inter-region controller and translates it into actual pathing programmed on the network. This was rejected as it compromised isolation, since the upper-layer controller could still influence (and poison) the intra-region controllers. **(2) Global Controller fully controls Inter-Region Traffic.** While a dedicated intra-regional controller controls pathing of intra-region traffic, the global controller has visibility into the entire network and controls the end-to-end pathing of inter-region traffic. This was also rejected on the same basis: it requires the global controller to program nodes deep within the region, leaving open the risk that it could impact the intra-region control plane. This also scales poorly as the global core still contains the entire network.

5.1.2 Border Node Separation. The boundary between regional and global domains is formed by a set of *border nodes* in each region. These nodes are physically shared between the regional and global controllers, i.e., both controllers can program these nodes. We have safeguards in place to prevent the global controller from poisoning the regional controller via its programming of these nodes and vice versa. For example, overload protection protects from a rogue controller flooding the shared node with programming operations. In the event of conflicting intent, the border nodes strictly prioritize the regional controller intent, protecting regional traffic from outside interference. Having separate physical regional and global border nodes would have the benefit of even stricter isolation guarantees; however, this requires additional WAN capacity and comes at a significant cost premium.

5.1.3 Physical vs. Logical Carving of Border Links. Links connecting two border nodes can be used for both intra-region and inter-region traffic. To maintain isolation, these physical links are logically carved into separate virtual slices each with dedicated bandwidth allocations. Specifically, a portion of the link's capacity is dedicated to the regional controller, and the remainder is dedicated to the global controller. This logical separation, enforced by the respective controllers, prevents contention and ensures that a global controller cannot inadvertently consume capacity intended for intra-region traffic or vice versa. This approach provides the flexibility to adjust the capacity split via configuration changes alone without requiring any physical changes, which proved vital during the network's evolution and migration. This logical carving is automatically determined during the network planning process based on projected intra- and inter-regional demands and adjusted monthly.

The alternate approach of physically carving the border links increases isolation. However, it was rejected for two reasons: the loss of ability to adjust the capacity split to accommodate changes in intra- and inter-region traffic patterns, and high deployment complexity not commensurate with the increased isolation benefits.

5.1.4 Load Balancing across the Border. A critical design decision is how to load-balance inter-region traffic across border nodes. We require strict isolation between global and regional controllers, which prohibits online communication between them and mandates that load-balancing decisions rely solely on locally available inputs. We selected an algorithm based on the capacities adjacent to the border nodes, constrained by the logical carving (§5.1.3), to maximize total cross-border flow. This approach is effective as it not only satisfies the isolation requirement but also produces stable traffic handoff patterns and predictable pathing. Moreover, this stability is achieved without sacrificing flexibility, as the algorithm can adapt to dynamic changes in border node capacities. This adaptability is enabled by packet sampling at the border nodes, which provides the controller with the necessary data on inbound traffic volume.

5.1.5 Controller Deployment Model. The regional BwE/TE controller jobs run in compute clusters co-located within the region they manage across at least three disaster domains. This ensures they can continue to operate independently if the region is cut off from the global WAN, and aligns with the zone-isolation failure domain. A standard leader election mechanism is used [8].

6 Deployment

The deployment and migration to GGN is a brownfield effort, executed on a live network carrying all of Google's production traffic. A "big bang" cutover was not an option.

6.1 Traffic Migration and Availability SLOs

Our traffic migration strategy centered on a gradual, host-steered migration, spanning multiple years (Fig. 8). We prioritized safety and meeting the SLO of our highest priority traffic class, i.e., Tier-0 user-facing ("UF") traffic. Meeting SLO targets is especially critical for UF traffic in order to ensure that services experience similar network performance on GGN post migration as they did on B2. Our migration was carried out in steps:

(1) **Physical Interconnect:** First, we built physical connectivity between the legacy networks and the new GGN core components, creating connections from B2 edges to B4/GGN.
(2) **Phased Migrations to B4:** We staggered the migration of UF traffic from B2 to shard 1, to establish our sharded WAN's capability and readiness to carry UF traffic.
(3) **Ride the Growth for New Shards:** With shard 1's capability established through the migrations from B2, we next leveraged organic traffic growth to populate the new shards, so new traffic demands were directly provisioned on them, which allowed for a cost-effective build-out.
(4) **Host-Based Steering:** Host-based packet marking provided fine-grained control over migration. We can enable sharding on a per-application, per-destination, or even per-flow basis by instructing hosts to mark their flows accordingly to be sharded or not. This allowed for a highly controlled, incremental rollout.

We built automated systems that performed multiple safety checks before migrating traffic to GGN shards. These include: (a) Adequate capacity for committed bandwidth on the respective shard, (b) Availability SLOs both at L3 and L7, and (c) Latency SLOs between source and destination endpoints on GGN. Our TE system allowed us to reserve capacity on GGN for flows *before* they were migrated, ensuring a smooth transition without initial congestion.

Prior to GGN, all capacity planning for UF traffic was only performed on B2. We added support to our capacity planning systems to forecast traffic demand and build network capacity on GGN under various failure scenarios in order to meet the 4 9s of availability for all UF traffic that we planned to migrate from B2 to GGN. A key risk-mitigation strategy was to plan and build network capacity on both B2 and GGN shard 1 for UF traffic, so that in case of any performance issues or outages, we could immediately revert the UF traffic to be routed on B2 instead. We deployed new capacity to GGN shard 1 in a phased manner to accommodate all the flows planned for migration. We migrated traffic in phases: we would migrate a small volume weekly from a single $\langle src, dst \rangle$ cluster pair at a time, with our automated measurement systems continuously monitoring network health and availability. Our gradual migrations gave confidence in shard 1's ability to meet the availability SLOs of UF flows. We then increased both the geographical scope and volume of traffic we would migrate weekly to shard 1. To enable these large scale migrations, we continuously deployed additional capacity to shard 1. We planned capacity and onboarded new traffic directly to shards 2-4, without requiring backup capacity on B2.

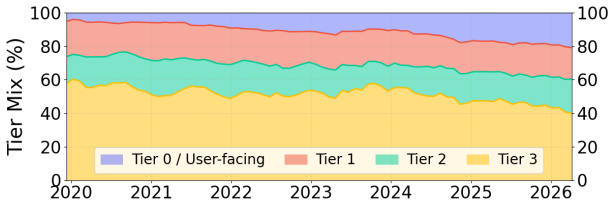


Figure 7: Traffic Priority Classes in GGN. As reliability improvements are gradually rolled out, this allowed GGN to increasingly carry Tier-0/UF traffic (from 5% to 21%).

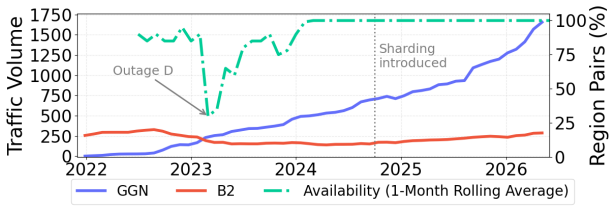


Figure 8: Tier-0/UF Volume and GGN Availability. Volume is normalized to GGN in 2022. Availability measures % of region-pairs meeting UF SLO. The availability dip in 2023 is outage D (§7.3.2) and predates sharding, which would have protected against it.

6.2 Network Scale and Sharding Deployment

As shard 1 (B4) was an extant global network, enabling the sharded GGN architecture required (a) deploying new shards (2-4), and then (b) safely migrating traffic to use all shards. Creating these new shards requires physical deployment on a global scale. This sharding is currently ongoing, and we expect fully sharding the entire network to take multiple years.

Figs. 7-11 show overall UF volume growth, network growth, sharding and WBB adoption. We expect the sharding adoption to ramp up sharply in the future. Fig. 9 also shows the number of sharded supernodes deployed so far (in shards 2-4) compared to shard 1. Initially, these sharding sites serve a regional or metro-level aggregation function.[9] We are building their WAN interconnects gradually, either by acquiring new WAN assets, or by re-balancing the existing L1 infrastructure from shard 1 to serve shards 2-4 instead. We are prioritizing earlier sharding deployments for mega-campuses where ML/AI training is conducted. As of July 2026, shards 2-4 account for 35% of overall GGN capacity, and up to 40% of the traffic volume within one of our largest ML/AI regions being sharded. Note also the rapid adoption of sharding: as Fig. 10 shows, each of the new shards 2-4 is ~5 times of the size B4 was in 2019 as a baseline (already one of the largest networks in the world at the time), with GGN overall being ~45 times as big. This build is both massive and rapidly accelerating: on average this is roughly equivalent to building a new B4 every ~2 months, and more recently as frequently as every ~2 weeks.

[9] A *metro* is a logical grouping of regions such that round trip times (RTT) within a metro would not exceed 5ms.

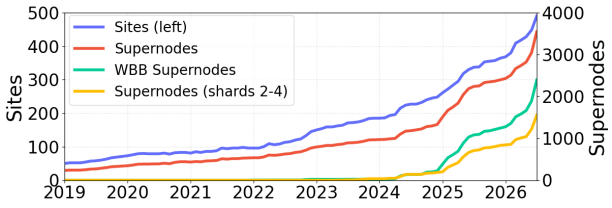


Figure 9: GGN Site and Supernode Growth. Note the rapid WBB and sharding adoption, and ~10x overall site growth since 2019.

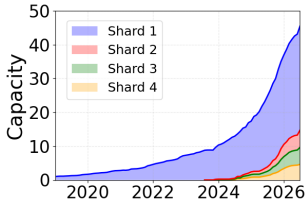


Figure 10: Capacity by Shard (Stacked). Normalized to shard 1 (B4) in 2019. GGN is ~45x as big as B4 was in 2019.

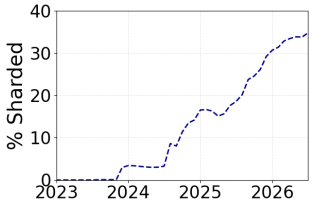


Figure 11: Sharded Capacity. 35% of overall GGN is on shards 2-4 today.

Due to our use of ECMP for sharded traffic (§7.2.3), we attempt to keep our deployed capacity across all of the new shards symmetric to ensure that our load balancing remains efficient. Future work will enable WCMP across shards to enable asymmetric deployments.

6.3 Regionalization Deployment

Regionalization must be executed on a live production network, and must be gradual and carefully managed to avoid transient congestion or traffic blackholing. Shards 2-4 are greenfield deployments and regionalized from their inception. This creates a natural synergy with the core sharding deployment, as the host-based steering mechanism used to migrate traffic to these shards also increases the volume of traffic protected by regionalization. In contrast, regionalizing shard 1 (B4) is more complex as it has a pre-existing global controller. Our strategy is to gradually transform this controller to internally mimic the behavior of a regional and inter-regional controller. This multi-stage process involves introducing logical carving of border links, enforcing regional pathing constraints, and splitting end-to-end flows at the border into regional and inter-regional segments. Once this transformation is complete, control of the regional flows is transferred to a dedicated regional controller, and the regional topology is then safely removed from the global controller's purview. To date, we have regionalized the control plane for 53% of the network's regions, protecting 12% of the total intra-region traffic. Since Jan. 2024, the volume of intra-region traffic on the network has grown by 406% and the volume of traffic with regionalization protection has increased by 947%.

7 Lessons Learned

Given the global scale of B2/B4's footprint, the transition to GGN is a multi-year effort involving careful planning, staged migration, and continuous validation. This migration, while still in progress, is having a substantial positive impact on availability. We describe some of our experiences so far, including four case-studies of real-world outages and how GGN architecture handled them.

7.1 Meeting Availability SLOs

By 2023, many critical GGN design features had only been deployed in select regions (e.g., regionalization, PRR, and control plane and availability improvement features (§A)). Thus, not all region pairs were consistently meeting the stringent 99.99% availability required to carry UF traffic. In 2024, we identified a subset of region pairs that were already consistently meeting our SLOs and prioritized their migration to gain early experience with carrying UF flows on GGN. The remaining region-pairs could not consistently meet our 99.99% availability SLOs since we did not have the network capacity to support PRR in those region pairs; additionally we had not yet rolled out Active Path Diversity and Automated TE-to-Routing Fallback (§A). We progressively rolled out the aforementioned features globally through 2023, which dramatically and consistently improved availability. Over the last two years, every region pair in GGN has been consistently meeting >99.99% availability SLO (Fig. 8). We then accelerated migration of existing UF flows, and onboarded *new* UF flows to GGN directly.

7.2 Sharding Protection and Challenges

Core sharding is GGN's primary defense against global outages. We validated its effectiveness through extensive simulation and controlled production tests [7], and actual outages. In a simulated full-shard failure, high-priority traffic experiences a brief period of disruption (sub-second to seconds) as data plane and edge control plane mechanisms steer traffic to healthy shards. The aggregate bandwidth for these flows is fully restored within minutes as the global BwE controllers converge on a new allocation. Lower-priority traffic, for which we do not provision $N-1$ capacity, is gracefully degraded via BwE's policy-based throttling, preserving the SLOs of critical services. This confirms that the sharded architecture successfully transforms a potential multi-hour global outage into a brief, SLO-budgeted disruption.

We next describe sharding's protection against two actual unplanned outages from production.

7.2.1 Outage A: Routing Underlay Response. An operational error took down routing for an entire site on Shard 3. We observed the layered reaction to the loss of reachability. The system first removed the offending shard via fast reaction by the routing underlay and methods such as PRR, which was then identified by SE, enabling it to converge its solution to remove the offending shard.

In Fig. 12, we plot loss of probes representing the experience of sharded traffic, and also demonstrate our visibility into issues with specific shards, via probes that leverage our routing design (§4.2) to target a specific shard, allowing us to observe issues within a shard even when SE has removed it from the path of user traffic. We also plot the state of the shard failures initiated by SE. We observed a brief period of minor loss for user traffic when Shard 3 fails. This loss fully resolves as SE disables this shard for the impacted flows. Soon after the outage is resolved within the shard, SE re-enables it for all flows with no impact to user traffic.

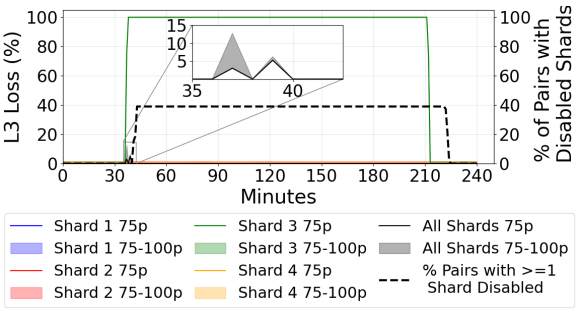


Figure 12: Outage A. A routing issue on a single shard is mitigated to a minor disruption for sharded traffic (inset). BGP and PRR mitigate the issue. SE then disables the shard until observed to be stable.

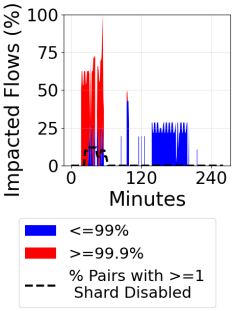


Figure 13: Outage B. SE overrides broken routing to fail a shard.

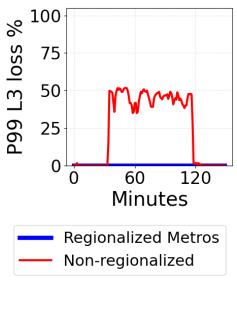


Figure 14: Outage C. P99 L3 loss % for intra-region UF traffic. No loss for regionalized.

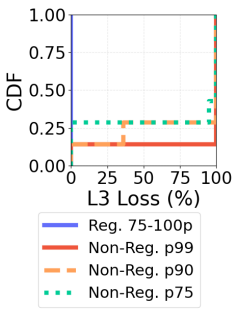


Figure 15: Outage D. % L3 loss across regions. No loss for regionalized.

7.2.2 Outage B: SE-Optimized Response. A capacity outage triggered an overload for shard 1's TE controller. This resulted in programming delays that prevented the system from fully converging to optimize tunnels around the reduced capacity. As a result, we did not see extensive blackholing loss, but observed traffic throttling initiated by BwE to fit traffic within the reduced available capacity. Fig. 13 shows the severity of BwE throttling and SE's reaction. We see that during periods of severe throttling impacting our high-availability (Tier-0 and Tier-1) flows, SE initiates a failure of the impacted shard in a number of locations (min 15-60). When the TE system converges and sufficient bandwidth is available, the traffic is placed back on all shards. We also see that during periods of throttling impacting lower availability (Tiers-2 and Tier-3) traffic (min 140-200), SE does not react, as this traffic has sufficient outage budget available to absorb the outage. In summary, this outage demonstrates the value of SE, as it gracefully reacts to partial capacity outages in a manner in line with our business policies.

7.2.3 Cross-Shard Load Balancing Challenges. Constrained by edge-router scaling limits, our architecture currently employs ECMP to distribute traffic *across* shards, with fail-over managed at a per-destination-cluster granularity. This approach introduces operational challenges when hardware heterogeneity or deployment delays (e.g., staggered upgrades) lead to capacity asymmetry across shards. Since ECMP assumes uniform capacities, these result in sub-optimal utilization of the total available bandwidth. We currently tolerate these transient efficiency losses while evaluating more flexible traffic distribution methods, including WCMP and flow pinning strategies for subset of shards for better utilization.

Hardware heterogeneity and varying loads in our production network often compromise ideal ECMP behavior. Data plane optimizations [24] can further cause traffic distribution to diverge; e.g., we observed a 6% imbalance between shards in one location. This divergence introduces inaccuracies in control plane modeling, complicating flow management across the sharded core. We are addressing this by incorporating these imbalances into our models while phasing out underperforming device versions.

7.3 Regionalization Protection and Lessons

We describe two actual outages where regions with a regionalized control plane were protected from control plane bugs and bad inputs that impacted other regions.

7.3.1 Outage C: Protection from Bad Configuration. A faulty pathing policy was propagated to the global TE controller, causing it to retract optimal tunneled paths for intra-region traffic in some regions. This traffic then reverted to underlay routing, which defaulted to shortest-path. This suboptimal pathing led to network congestion and significant packet loss for UF traffic, an issue that persisted for 90 mins until the faulty configuration was reverted. One such region experienced a sharp increase in L3 loss (Fig. 14). Regions with control plane regionalization, which manage their own intra-region pathing, did not experience any packet loss.

7.3.2 Outage D: Protection from Corrupted Input. A day-one bug in the topology exporting pipeline pushed a bad topology missing many sites to the global TE controller. The global controller programmed paths based on this faulty data, leading to brief but severe widespread packet loss. This was the cause of the main dip in 2023 availability (Fig. 8). Regions with a regionalized control plane were completely protected from the impact of the corrupted global data (Fig. 15). Note that this outage predates the introduction of sharding, which would have further protected against its impact for locations not yet regionalized.

7.3.3 Network Planning Challenges. Adapting the network topology for regionalization incurs an efficiency overhead that requires provisioning additional capacity. The initial integration with network planning relied on a post-hoc fit check which provided feedback on whether the capacity provisioned was sufficient. However, as ML/AI-driven growth quickly outpaced projections, the associated costs spiraled, rendering this approach economically unviable. This highlighted the need to integrate regionalization constraints directly within the network planning optimization algorithms, which has now been incorporated. Moreover, we adopted a specialized load balancing strategy for inter-region traffic terminating at border nodes in the larger regions. Together, these interventions significantly reduced the costs, restoring economic viability.

7.4 Cost Analysis

WBB Adoption. We compared the total cost of ownership (including development, equipment, power, warranty, etc.) of WBB-based fabrics against a hypothetical next-generation fabric we would have developed. We found that a WBB-based approach, amortized over 5 years, was essentially cost-neutral (~2% saving in net present value), while providing critical business, architectural and reliability benefits (e.g., vendor optionality, fast HW-based failover handling).

Core Sharding. We compared the cost of physical sharding vs. growing a single large backbone. With assumptions regarding the sharding design (§4), e.g., protection only for traffic tier-0 and tier-1, and safety policies in place to prevent concurrent shard failures, we found a sharded WAN brings the benefits of global outage protection on a physically sharded topology with only a 2-5% cost increase.

Regionalization. While our design provides strict isolation guarantees, they introduce a trade-off in reduced network efficiency: (a) There is a loss of fungibility between regional and global capacity shares due to logical carving of the border links. (b) Capacity-based load balancing, which relies solely on local inputs without communication between peering controllers, can diverge from optimal end-to-end pathing. The implementation of the regionalized architecture incurred a 1% overhead on the total network cost.

8 Related Work

Virtually all major cloud providers face similar challenges in building and operating planet-scale WANs, i.e., exponential scaling requirements to face ML/AI workloads, traffic engineering to meet efficiency and availability requirements, and the complexity of operational management of a network with global footprint. We briefly discuss the approaches of Microsoft's **OneWAN** [20], Meta's **Express Backbone (EBB)** [10, 16], and Alibaba's **Global WAN (AGN)** [28, 29] and its next-generation **eCore** WAN [9].[10]

Microsoft's OneWAN shares a similar motivation with GGN: unifying a split WAN architecture (SWAN for inter-data-center, CORE for Internet) into a single, SDN-based backbone. Microsoft's **Blastshield** [19] is its decentralized Traffic Engineering system, and employs a geographically-partitioned control plane managed by independent controllers. While Blastshield's controllers are responsible for programming routers only within their domain, they ingest network-wide inputs to compute global pathing solutions. GGN's regionalization enforces stricter isolation guarantees: each regional controller consumes inputs exclusively from its own region. Regionalization also reduces the global core topology size, leading to lower path computation time for cross-region paths, while Blastshield ingests global-level topology and demand inputs.

Meta's EBB employs a multi-plane (shard) architecture for improving reliability with planes enabling failure isolation and A/B testing. Traffic within an EBB plane is managed using MPLS-based segment routing, programmed using a global per-plane controller (while still not fully distributed, GGN's per-shard TE has more delegated regional control). EBB's controllers leverage an Unequal-Cost Multi-Path (UCMP) strategy on the edge to account for dynamic link failures and maintenance, as well as switch-local precomputed capacity-aware Fast-Reroute (FRR) that prioritizes high-priority traffic. Similar to GGN, EBB's controller computes primary and backup paths per src/dst pair, and traffic is symmetrically planned and spread across planes in steady-state

Alibaba's Global WAN (AGN) is also a private global WAN that interconnects its data center networks, and peers with external ISPs. The DCN and WAN interconnect using eBGP, with the WAN being a single AS that uses iBGP/IS-IS to internally redistribute routes learned from outside. This is very similar to the routing underlay of GGN. As AGN proved ill-suited to meet new demands for similar reasons of scale and complexity, this necessitated a re-design and a migration to the simplified eCore architecture. eCore uses a multi-plane architecture with SRv6-based Traffic Engineering [12] and smaller domains, and shares many similarities to GGN. It also faced many of the same challenges, including the safety of traffic migration from AGN to eCore. Unlike GGN, eCore emphasizes router HW uniformity (an in-house developed fixed-form white-box) for standardized deployment and management.

In summary, although these hyperscaler WAN networks have been designed and developed independently, they exhibit a similar evolution with a focus on redundant shards (or "planes") for availability and scale, smaller control plane domains for better fault-isolation, and a simpler protocol ecosystem for better reliability and management.

9 Conclusion and Future Work

GGN represents a paradigm shift in Google's WAN architecture, specifically built to meet the new stringent availability, scalability, and velocity requirements of Google Cloud and AI. By embracing core principles of sharding for global fault isolation, regionalization for containing local failures, and hardware optionality through open standards, we have built a planet-scale backbone that provides a dramatic step-function improvement in reliability and operational agility over legacy backbone designs. Our experience also demonstrates a recipe for migrating a live, planet-scale network to a new architecture without impacting customers.

Looking ahead, we are accelerating our deployments and expanding the sharding and regionalization footprints globally. We are also actively working to improve the granularity of sharded load-balancing for better efficiency and deployment flexibility, expanding the scope of Protective ReRoute (PRR) and Smart Fast ReRoute-like functionality (FRR) [18], among others. We believe that this global network, built with commodity vendor hardware and managed with open standards and extensible software, is a foundation for networking innovation in the next decade and beyond.

Acknowledgments

Building GGN is an ongoing, multi-year collaborative effort spanning hundreds of talented engineers. We sincerely thank our colleagues at Google helping bring it to reality. We specifically thank Kondapa Naidu Bollineni, Yun Freund, David Gong, Konstantin Iakhontov, Jay Kaimal, Rob Peasegood, and Yang Zhang. We also thank Jeff Mogul, David Wetherall, and our anonymous SIGCOMM PC reviewers for their valuable feedback and time. This work does not raise any ethical issues. Gemini was used for copyediting and to render the figures.

[10]To our knowledge, Amazon's AWS Global WAN does not externally publish peer-reviewed details on its WAN architecture.

References

[1] 2019. Google Cloud Networking Incident #19009. https://status.cloud.google.com/incident/cloud-networking/19009

[2] 2025. Kubernetes Network Emulation. https://kubernetes.io/docs/home/.

[3] 2025. Ondatra: Open Network Device Automated Test Runner and API. https://github.com/openconfig/ondatra.

[4] 2025. Openconfig. https://github.com/openconfig

[5] 2026. Google Cloud Documentation: Geography and regions. https://docs.cloud.google.com/docs/geography-and-regions

[6] 2026. OpenConfig Featureprofiles. https://github.com/openconfig/featureprofiles

[7] Heather Adkins, Betsy Beyer, Paul Blankinship, Ana Oprea, Piotr Lewandowski, and Adam Stubblefield. 2020. *Building Secure and Reliable Systems.* O'Reilly Media.

[8] Mike Burrows. 2006. The Chubby lock service for loosely-coupled distributed systems. In *OSDI 2006.*

[9] Dennis Cai, Roy Jiang, and Manish Mukherjee. 2025. eCore: Architecture and Whitebox Evolution of Alibaba's Service-Oriented DCI Network. In *OCP Global Summit 2025.* San Jose, CA. Alibaba Cloud & Cisco Systems. https://www.segment-routing.net/images/OCP-summit25-eCore_Alibaba.pdf

[10] Marek Denis, Yuanjun Yao, Ashley Hatch, Qin Zhang, Chiun Lin Lim, Shuqiang Zhang, Kyle Sugrue, Henry Kwok, Mikel Jimenez Fernandez, Petr Lapukhov, Sandeep Hebbani, Gaya Nagarajan, Omar Baldonado, Lixin Gao, and Ying Zhang. 2023. EBB: Reliable and Evolvable Express Backbone Network in Meta. In *SIGCOMM 2023.*

[11] Andrew D. Ferguson, Steve Gribble, Chi-Yao Hong, Charles Killian, Waqar Mohsin, Henrik Muehe, Joon Ong, Leon Poutievski, Arjun Singh, Lorenzo Vicisano, Richard Alimi, Shawn Shuoshuo Chen, Mike Conley, Subhasree Mandal, Karthik Nagaraj, Kondapa Naidu Bollineni, Amr Sabaa, Shidong Zhang, Min Zhu, and Amin Vahdat. 2021. Orion: Google's Software-Defined Networking Control Plane. In *NSDI 2021.*

[12] Clarence Filsfils, Pablo Camarillo, John Leddy, Daniel Voyer, Satoru Matsushima, and Zhenbin Li. 2021. Segment Routing over IPv6 (SRv6) Network Programming. RFC 8986. https://www.rfc-editor.org/info/rfc8986

[13] Ramesh Govindan, Ina Minei, Mahesh Kallahalla, Bikash Koley, and Amin Vahdat. 2016. Evolve or Die: High-Availability Design Principles Drawn from Googles Network Infrastructure. In *SIGCOMM 2016.*

[14] Chi-Yao Hong, Subhasree Mandal, Mohammad Al-Fares, Min Zhu, Richard Alimi, Kondapa Naidu B., Chandan Bhagat, Sourabh Jain, Jay Kaimal, Shiyu Liang, Kirill Mendelev, Steve Padgett, Faro Rabe, Saikat Ray, Malveeka Tewari, Matt Tierney, Monika Zahn, Jonathan Zolla, Joon Ong, and Amin Vahdat. 2018. B4 and after: managing hierarchy, partitioning, and asymmetry for availability and scale in google's software-defined WAN. In *SIGCOMM 2018.*

[15] Peng Huang, Chuanxiong Guo, Lidong Zhou, Jacob R. Lorch, Yingnong Dang, Murali Chintalapati, and Randolph Yao. 2017. Gray Failure: The Achilles' Heel of Cloud-Scale Systems. In *HotOS 2017.*

[16] Faisal Iqbal, Vitaly Neganov, Brian Chang, Rong Rong, Yuanjun Yao, Marek Denis, Qin Zhang, Alexandru Manea, Anton Marchenko, Ulas Kozat, Aditya Akella, and Ying Zhang. 2026. Express Lane to Efficiency and Reliability: Multi-Dimensional Control in Meta's Express Backbone Network. In *NSDI 2026.*

[17] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, Jon Zolla, Urs Hölzle, Stephen Stuart, and Amin Vahdat. 2013. B4: Experience with a globally-deployed software defined WAN. In *SIGCOMM 2013.*

[18] Alexander Krentsel, Nitika Saran, Bikash Koley, Subhasree Mandal, Ashok Narayanan, Sylvia Ratnasamy, Ali Al-Shabibi, Anees Shaikh, Rob Shakir, Ankit Singla, and Hakim Weatherspoon. 2024. A Decentralized SDN Architecture for the WAN. In *SIGCOMM 2024.*

[19] Umesh Krishnaswamy, Rachee Singh, Nikolaj Bjørner, and Himanshu Raj. 2022. Decentralized cloud wide-area network traffic engineering with BLASTSHIELD. In *NSDI 2022.*

[20] Umesh Krishnaswamy, Rachee Singh, Paul Mattes, Paul-Andre C Bissonnette, Nikolaj Bjørner, Zahira Nasrin, Sonal Kothari, Prabhakar Reddy, John Abeln, Srikanth Kandula, Himanshu Raj, Luis Irun-Briz, Jamie Gaudette, and Erica Lan. 2023. OneWAN is better than two: Unifying a split WAN architecture. In *NSDI 2023.*

[21] Alok Kumar, Sushant Jain, Uday Naik, Anand Raghuraman, Nikhil Kasinadhuni, Enrique Cauich Zermeno, C. Stephen Gunn, Jing Ai, Björn Carlin, Mihai Amarandei-Stavila, Mathieu Robin, Aspi Siganporia, Stephen Stuart, and Amin Vahdat. 2015. BwE: Flexible, Hierarchical Bandwidth Allocation for WAN Distributed Computing. In *SIGCOMM 2015.*

[22] Bingzhe Liu, Colin Scott, Mukarram Tariq, Andrew Ferguson, Phillipa Gill, Richard Alimi, Omid Alipourfard, Deepak Arulkannan, Virginia Jean Beauregard, Patrick Conner, P. Brighten Godfrey, Xander Lin, Joon Ong, Mayur Patel, Amr Sabaa, Arjun Singh, Alex Smirnov, Manish Verma, Prerepa V Viswanadham, and Amin Vahdat. 2024. CAPA: An Architecture For Operating Cluster Networks With High Availability. In *NSDI 2024.*

[23] John Lunney, Sue Lueder, and Gary O' Connor. 2016. *Postmortem Culture: Learning from Failure.* https://landing.google.com/sre/book.html

[24] Mubashir Adnan Qureshi, Yuchung Cheng, Qianwen Yin, Qiaobin Fu, Gautam Kumar, Masoud Moshref, Junhua Yan, Van Jacobson, David Wetherall, and Abdul Kabbani. 2022. PLB: congestion signals are simple and effective for network load balancing. In *SIGCOMM 2022.*

[25] Arjun Singh, Joon Ong, Amit Agarwal, Glen Anderson, Ashby Armistead, Roy Bannon, Seb Boving, Gaurav Desai, Bob Felderman, Paulie Germano, Anand Kanagala, Jeff Provost, Jason Simmons, Eiichi Tanda, Jim Wanderer, Urs Hölzle, Stephen Stuart, and Amin Vahdat. 2015. Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network. In *SIGCOMM 2015.*

[26] Abhishek Verma, Luis Pedrosa, Madhukar Korupolu, David Oppenheimer, Eric Tune, and John Wilkes. 2015. Large-scale cluster management at Google with Borg. In *EuroSys 2015.*

[27] David Wetherall, Abdul Kabbani, Van Jacobson, Jim Winget, Yuchung Cheng, Brad Morrey, Uma Parthavi Moravapalle, Phillipa Gill, Steven Knight, and Amin Vahdat. 2023. Improving Network Availability with Protective ReRoute. In *SIGCOMM 2023.*

[28] Fangdan Ye, Da Yu, Ennan Zhai, Hongqiang Harry Liu, Bingchuan Tian, Qiaobo Ye, Chunsheng Wang, Xin Wu, Tianchen Guo, Cheng Jin, Duncheng She, Qing Ma, Biao Cheng, Hui Xu, Ming Zhang, Zhiliang Wang, and Rodrigo Fonseca. 2020. Accuracy, Scalability, Coverage: A Practical Configuration Verifier on a Global WAN. In *SIGCOMM 2020.*

[29] Yifei Yuan, Fangdan Ye, Yifan Li, Jingkai Zhang, Mengqi Liu, Yuyang Sang, Ruizhen Yang, Duncheng She, Zhiqing Ye, Tianchen Guo, Xiaobo Zhu, Xinji Tang, Li Jia, Zhongyu Guan, Lingpeng Su, Ci Wang, Ruiyang Feng, Shuo Wu, Zhonghui Xie, Cheng Jin, Peng Zhang, Qing Ma, Xianlong Zeng, Dennis Cai, and Ennan Zhai. 2025. New Evolution of Hoyan: Enhancing Scalability, Usability, and Accuracy for Alibaba's Global WAN Verification. In *SIGCOMM 2025.*

[30] Junlan Zhou, Malveeka Tewari, Min Zhu, Abdul Kabbani, Leon Poutievski, Arjun Singh, and Amin Vahdat. 2014. WCMP: weighted cost multipathing for improved fairness in data centers. In *EuroSys 2014.*

Appendices

Appendices are supporting material that has not been peer-reviewed.

A Control Plane Optimizations

Building a unified, sharded, and regionalized WAN at Google's scale required significant innovation in our control plane systems. We highlight several key optimizations that were critical to the success of GGN.

A.1 Active Path Diversity

To improve resilience against single-link failures, we enhanced our TE solver to provide active path diversity. For high-priority traffic, instead of a single primary path and a cold backup, the solver now computes multiple, physically disjoint primary paths. Traffic is actively load-balanced across these paths. In the event of a link failure, only one of the active paths is affected. Host-based fast failover mechanisms like PRR can then immediately re-hash traffic onto the surviving active paths within a few round-trip times, providing sub-second recovery without waiting for control plane convergence. In order to meet the latency SLO targets, especially for UF traffic, we also limit the latency stretch of active paths.

A.2 Automated TE to Routing Fallback

To protect against catastrophic TE controller failures, we developed Automated TE-to-Routing Fallback. This system continuously monitors end-to-end path health using active probing of both TE and routed paths. If it detects persistent high packet loss on TE-steered paths, it automatically triggers a fallback at the ingress router, withdrawing the TE-programmed paths and allowing traffic to flow over the simpler, more robust underlay routing (BGP/ISIS). This provides a crucial safety net against unforeseen bugs in the complex TE system. Note that unlike other TE systems from other hyperscalers, GGN's TE and routing systems are specifically engineered to have no dependency on each other.

A.3 Tree Tunnels for Scalability

B4's TE system created a full mesh of tunnels between all site-pairs, leading to $O(N^2)$ scaling for the number of tunnels and forwarding entries. To address this, we developed *Tree Tunnels*. This optimization merges multiple linear tunnels that share a common path segment towards a destination into a single, tree-structured tunnel. By programming forwarding state on a tree rather than individual paths, we dramatically reduce the number of required tunnels and FIB entries at transit and destination nodes by up to 90%, providing crucial scaling headroom.

A.4 NU-BwE Conditional Enforcement

Migrating all user-facing traffic to GGN for enforcement was challenging, as many services lacked accurate, up-to-date bandwidth commitments. Leaving this traffic unenforced posed a risk to network stability. Network Unaware BwE (NU-BwE) provided a solution by enabling conditional enforcement. For flows without adequate commitments, NU-BwE applies a safe, generously overridden approval, allowing them to be brought under enforcement without impacting legitimate traffic. This enabled a safe and complete migration of all user traffic to the new enforcement regime.

A.5 NCS-Free Architecture

Traditionally, each SDN domain in a site was managed by controllers running on dedicated, co-located Network Control Service (NCS) machines. This model became a scaling bottleneck and a significant operational expense. The NCS-free project re-architected our controllers to run on Google's standard, shared Borg infrastructure instead [26]. This involved solving challenges related to remote connectivity, higher-latency control loops, and distributed leader election, but ultimately resulted in a more scalable, cost-effective, and operationally simpler deployment model.

B Historical Outage Analysis

Given Google's large-scale, complex, distributed systems and their rate of change, outages are inevitable. As part of Google's blameless postmortem culture, all incidents trigger a postmortem report. This report details the incident's impact, immediate remediation, root cause, and follow-up actions for improving the architecture and preventing such an incident from recurring [13, 23].

Based on this corpus, we performed a cross-sectional study of all historical outages on our WAN network between 2021-2025, and considered whether they would have been protected against by the GGN architecture presented in this paper (Fig. 16). Note we have discounted incidents where there was a (a) capacity shortfall (as this indicates a mismatch between the physical topology's capacity or redundancy vs. demand), (b) thermal/power events, or (c) Edge issues, as this paper focuses on the GGN Core.

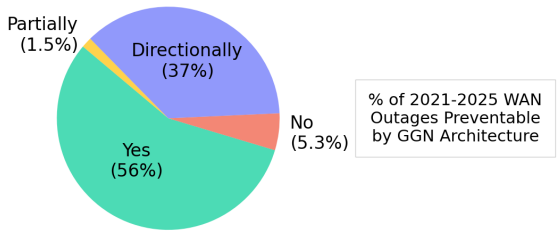


Figure 16: Classification WAN Outages, 2021-2025 This classifies whether historical incidents would have been addressed by the GGN architecture.

The majority were incidents that would have been addressed by the GGN architecture had it been in place at the time (either through sharding, regionalization, vendor-optionality, Protective ReRoute (PRR) or Smart Fast ReRoute-like functionality (FRR) [18], or other methods described in this paper).

Around 37% of the outages are *directionally* addressed, meaning that they would be addressed through phased improvements to the GGN architecture that are planned or in active development, but not yet in place and fall under future work. Examples of this category include expanding FRR-like handling scope into the GGN

edge domains, sharding the management plane, and expanding hardware redundancy and optionality scope, among others.

Some of the "No" category examples include faults in shared optical line-systems that we're opting not to fully isolate at this time given the cost/benefit analysis.

C Load Balancing Efficiency

A fundamental design trade-off in core sharding involves the granularity of traffic distribution at the network edge. To quantify these trade-offs, we evaluated the performance of various distribution strategies. For the purposes of this discussion, we present the two strategies: a simple threshold-based ECMP approach (*ECMP*), and a *Ratio Heuristic* that approximates ratios across flow allocations while respecting WCMP table size constraints in routers. Evaluations were conducted using an internal risk analysis framework across thousands of simulated daily snapshots, incorporating historical hardware failure models and software-driven allocation asymmetries.

Our findings indicate that the necessity of fine-grained WCMP depends heavily on the failure mode. In scenarios dominated by hardware failures where allocations remain relatively symmetric, the simpler *ECMP* approach performs comparably well to the theoretical optimum. However, the value of the *ratio heuristic* becomes evident during some software-driven outages, such as phased rollouts or corrupted inputs, which introduce significant allocation asymmetries. In these cases, approximating allocation ratios deliver better performance than ECMP-based solutions.

Table 2 summarizes the results, which demonstrate that while ECMP is sufficient for hardware failure scenarios, ratio-aware traffic distribution is critical for maintaining SLOs during some complex software failures. Considering implementation complexity and cost, we adopted an ECMP-based traffic distribution mechanism featuring per-flow shard failure protection. This intermediate step allows future evolution toward a finer ratio-based WCMP solution as we gain operational experience at scale.

Table 2: SLO Adherence of Traffic Distribution Strategies

Failure	ECMP	Ratio Heuristic
H/W	Meeting SLO	Near optimal
S/W	Tier-0 / Tier-1: 100% meeting SLO Tier-2 / Tier-3: 56% meeting SLO	≈ 100% meeting SLO for all tiers

D Technology Introduction in GGN

D.1 Technology Introduction Velocity

Optionality has transformed how GGN introduces new technology. By investing in vendor-neutral APIs and open-source unit tests, we have cut development-to-pilot durations by 75% and eliminated any potential single-supplier bottleneck. This approach allows us to capitalize on the engineering velocity of the entire industry; if one vendor leads in a new technology generation, we can pilot and deploy it immediately. §D.2 describes our vendor qualification.

Standardized APIs also enable us to treat the data plane as a programmable platform for Google-developed applications deployed as containers. This effectively decouples feature velocity from traditional multi-year vendor release cycles. By internalizing the development of critical functionality, we eliminate the need for complex multi-vendor implementation and interoperability testing while still maintaining a robust, per-device failure domain.

D.2 Vendor Qualification Strategy

The transition to vendor device-based WAN Building Blocks (WBBs) for GGN required focus and significant investment in how we qualify vendor platforms with Google's control and management planes' requirements and operational tooling. Our desire for multi-vendor optionality comes at the cost of interacting with several different implementations of WBB – which must be qualified to be behaviorally compliant with our requirements, as well as inter-operable within the GGN deployment. In order to support feature development velocity, we must be able to efficiently qualify both new software releases, as well as new hardware platforms for deployment. To achieve this, we invested in significant enhancements to a traditional "tiered" approach of qualification. We begin by qualifying devices with *functional* tests, that validate the correctness of the features supported by individual network devices (e.g., gRPC API compliance for gRIBI, gNMI, etc. or packet forwarding). *Integration* tests then validate our pair-wise interactions between Google services and these network devices (e.g., ensuring our software rollout automation correctly interacts with WBB devices). Our *solution* tests then validate end-to-end vertical integration against the entire gamut of internal systems akin to a production deployment.

Implementing a qualification pipeline that supports multiple vendors, supports our velocity goals, and does so while being cost-effective required significant investment. Particularly, we developed three key components:

(1) Open Network Device Automated Test Runner and API (ONDATRA).
(2) Open Source *Feature Profile* Tests.
(3) Kubernetes Network Emulation (KNE).

ONDATRA [3] is an open source framework for developing tests against network devices. Our approach centered around ensuring that tests could be written by network subject matter experts against different lab environments without code changes. The framework provides the necessary abstractions to interact both with devices under test (DUTs), and automated test equipment (ATE) (devices used as a reference implementation against which DUTs must interoperate). Rather than requiring complex domain-specific languages for test implementations, ONDATRA uses the Go programming language as the means to write tests, allowing test assets to be maintained and managed using common software development practices.

Through our device qualification experience, we observed that "left-shifting" testing activities as close to developers working on network devices elicited faster bug discovery, and thus less qualification effort. Using ONDATRA as the test framework, we invested in building an open source suite of functional tests, called **feature profile** tests [6], which can be executed by network equipment vendors within their own lab environments. This approach transitions our device specifications from typical lists of required standards, to automated tests which programmatically verify devices against

the specifications within the tests that we write. By developing these test suites, and requiring their execution *before* hand-over of hardware or software for WBB to Google, we significantly reduced the integration effort required at Google to support the multiple vendors in GGN.

Finally, using the **Kubernetes** network emulation framework [2] proved critical for velocity. Within all stages of our development and testing process the inconveniences of physical hardware testbeds slow down our efforts. In regular regression cycles, hardware failures within lab environments and contention for limited hardware resources can introduce flaky tests. Development velocity is slowed down when developers must wait for lab resources to be reset to a clean state, and their tests to execute. Within our solution test environments, scale is limited by the trade-off in cost, space and power utilization that limits the number of physical DUTs that we deploy. To overcome these challenges, we invested in means to provide emulated network topologies – consisting of DUTs, ATEs, and links – in a manner that directly allows tests to be developed against them and then subsequently run on hardware devices. To do this, we worked with network equipment vendors (of both DUT and ATEs) to provide containerized implementations of their software, and the necessary machinery to deploy them within Kubernetes clusters. This implementation allows developers to have their own testing topologies and rapidly iterate on test development. Tests with no hardware dependencies to be executed against virtual network devices to reduce testbed contention, and large scale topologies can be built by using virtual and physical nodes in combination. Today, test teams across our test hierarchies use KNE-based topologies as part of their regular development cycles.

Together, these investments in testing have significantly reduced our time-to-qualify vendor releases, and the number of software defects observed in our development cycles and production network infrastructure.