UNIVERSITY OF CALIFORNIA
SANTA CRUZ

NETWORK LOAD BALANCING FOR GEOGRAPHICALLY DISTRIBUTED DATACENTERS

A dissertation submitted in partial satisfaction
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE AND ENGINEERING

by

Lakshmi Krishnaswamy

August 2026

The Dissertation of Lakshmi Krishnaswamy
is approved:

_____
Dr. Katia Obraczka, Chair

_____
Dr. Abdul Kabbani

_____
Dr. Chen Qian

_____
Donald Smith
Interim Vice Provost and Dean of Graduate Studies

Contents

Figures and Tables

List of Figures

List of Tables

Abstract

Network Load Balancing for Geographically Distributed Datacenters

by

Lakshmi Krishnaswamy

As datacenters scale up and become more geographically distributed, wide-area network inter-datacenter traffic, which typically consists of data-heavy tasks, has become increasingly prevalent. Some of the noteworthy challenges raised by the coexistence and interaction between inter- and intra-datacenter traffic are the differences in their QoS requirements, link utilization, and round-trip times. To the best of our knowledge, these challenges have not yet been addressed by current datacenter load balancers. To highlight this gap, we conducted a comparative performance study of state-of-the-art datacenter load balancers. Through extensive simulations, we study how they perform under different network topologies and workloads, including intra-datacenter, inter-datacenter, and mixed intra- and inter-datacenter workloads that reflect how datacenters have evolved to keep up with their continuously changing driving application landscape. Our study shows that current load balancers are not able to adequately distribute load under inter-DC workloads as well as mixed intra- and inter-datacenter traffic coexistence. Motivated by our observations, we introduce Balancia, a transport agnostic, lightweight network load balancer that dynamically switches between per-flow and per-packet control in order to provide adequate performance for both intra- and inter-DC traffic given their different characteristics and quality-of-service (QoS) requirements. We show that, when compared against state-of-the-art load balancers, Balancia achieves close to 80%

reduction in the 99% tail flow completion times for inter-datacenter traffic in the presence of intra- and inter-datacenter workload coexistence.

To my parents, and family and friends

Acknowledgements

I want to extend my sincere gratitude and deepest and biggest appreciation for Dr. Katia Obraczka and Dr. Abdul Kabbani for all their constant support, guidance and help throughout my graduate journey. Their patience, care and guidance helped me not just with research but also with life more broadly. I also would like to extend my gratitude and sincere thanks to Dr. Chen Qian and Dr. JJ Garcia-Luna-Aceves for all their feedback and support which greatly helped shaped my research as well. Thank you Chris Parsa and Rick Graziani for helping me build and shape my teaching and constantly inspiring me with your teaching. I would like to extend my thanks to Hari Krishna Kuttivelil, Shesha, Nayan and Stephanie for their collaborations and work with the Network Simulation Bridge project. Thank you to all my fellow labmates for all the collaborations, discussions and constant support always. Thank you to Dr. Vamsi Krishna Tumuluru sir, Dr. Chandar sir and Dr. Manikandan sir from PES University, who were instrumental in being able to get into undergraduate research and start my graduate journey and I am truly grateful to them for this. Special thanks and gratitude to Aldrin Isaac from eBay and Madhu Paluru from Aviz Networks for their guidance and mentorship. I would also like to thank Dr. Sepehr Abdous for his support and guidance.

Next, I want to extend my heartfelt thanks and gratitude to my parents Uma and Krishnaswamy, and sister Madhumitha for everything. Thank you for always believing in me and supporting me throughout and always being by my side. All of this is only because of your care, support and constant encouragement, no matter what. Thank you to my uncle Dr. Venkatesh Rajendran who has been my inspiration and role model and the reason this was all possible, and thank you to my aunt Priya and my cousins Maanasvi and Raghav for being my second home and taking care of me and being there

always. I also want to thank my paternal uncles aunts and cousins for taking care of me and supporting me. I want to thank my late maternal and paternal grandparents who have always stood by me and supported me and loved me abundantly. My biggest thank you and gratitude to my husband Ayush Kundra who has been a pillar of support throughout and constantly encouraging and being there for me.

The text of this dissertation includes reprints of the following previously published material:

1. **L. Krishnaswamy**, K. Obraczka, and A. Kabbani, "What's Next for Datacenter Load Balancers — A Comparative Performance Study of the State-of-the-Art," in *2024 IEEE 13th International Conference on Cloud Networking (CloudNet)*, Rio de Janeiro, Brazil, 2024, pp. 1–9. doi: 10.1109/CloudNet62863.2024.10815922.

2. **L. Krishnaswamy**, K. Obraczka, and A. Kabbani, "Towards Next Generation Datacenter Load Balancers," *TechRxiv* (preprint), Nov. 2025. doi: https://doi.org/10.36227/techrxiv.176341363.36019194/v1.

3. **L. Krishnaswamy**, K. Obraczka, and A. Kabbani, "A Network Load Balancer for Geographically Distributed Datacenters," in *2026 IFIP Networking Conference (IFIP Networking)*, Lugano, Switzerland, 2026, pp. 1-9. doi: 10.23919/IFIPNetworking70592.2026.11579256.

The co-authors listed in this publication directed and supervised the research which forms the basis for the dissertation.

Chapter 1

Introduction

Geographic redundancy and physical diversity have become increasingly popular techniques cloud compute providers use. By allowing workloads to be distributed across these geographically distributed datacenters [11], cloud providers can not only maximize service availability but also reduce normative user-perceived response time. Recent works such as [12] highlight cloud providers that deploy multi-location datacenters connected by long-haul links within the same metropolitan area or region. Examples of such cloud applications include storage systems that span across datacenters and regions [13–15], and distributed AI/ML training that takes place across multiple datacenters to accommodate the scale of training [16, 17]. While inter-datacenter interactions are facilitated through high-bandwidth wide-area networks (WAN) [18], they raise a range of contrasting characteristics and requirements when compared to intra-datacenter (DC) traffic. DC traffic is characterized by round-trip times (RTT) on the order of $\approx 100\mu$s, while WAN traffic exhibits RTTs on the order of $\approx 100m$s. In addition, WAN networks are constrained by their interconnect cost, which means they are not as over-provisioned in terms of their bandwidth capacity compared to the DC networks,

thus having relatively high bandwidth utilization.

§ 1.1 Motivation

As datacenter fabric and workload diversity increase, meeting application Quality of Service (QoS) requirements while efficiently utilizing datacenter resources (e.g., without over-provisioning) becomes even more challenging. To the best of our knowledge, there is little work on load balancers that address the interaction between DC and WAN traffic. For instance, state-of-the-art datacenter load balancers such as Vertigo [7], and Preemptive Deflection [8] have been designed to handle microbursts, which are "microsecond scale congestion events" [7] that occur within a datacenter network. They have been shown to effectively reduce traffic tail latencies in the presence of microbursts. However, microbursts aren't characteristic of WAN traffic and, as shown by our performance results, a scheme tailor-made for microbursts does not work optimally for WAN traffic. Solutions such as in Annulus [18], Gemini [19] and Uno [20] have approached the problem from a purely end-to-end, congestion control perspective. They do not examine the challenges of path selection, or allocating DC and WAN workloads across multiple paths to deliver the required QoS for each type of traffic. More importantly, with the coexistence of DC-WAN traffic, it is important to have both end-to-end, as well as in-network decision making capabilities. Specifically, given the difference in time-scales for DC and WAN traffic, it is important to be able to sense and react at both the end-host as well as from inside the network.

§ 1.2 Main Contributions

With this motivation, as part of this dissertation we set out by first highlighting the current state-of-the-art network load balancers, and describe in detail their mechanics. We also provide detailed background on datacenter network topologies, workload modeling and experimental methodologies for running quantitative studies. Further, we highlight emerging efforts at redesigning the transport layer to support AI-era datacenters, which need to handle the coexistence between DC and WAN traffic generated by ML training workloads. We conduct thorough quantitative study to benchmark the state-of-art network load balancers. Based on the takeaways from our qualitative and quantitative study, we develop an adaptive, distributed load balancing mechanism that can adequately perform for DC-WAN mixed traffic while utilizing datacenter resources efficiently.

The main contributions of this dissertation are as follows:

- We introduce a taxonomy to map out the design space of datacenter load balancers and use our taxonomy to classify state-of-the-art datacenter load balancers highlighting their main features.
- We map the load balancing and congestion control design space by examining the different congestion signaling mechanisms, including which entity generates them, what ensuing actions are taken and who takes them.
- Using our map of the load balancing and congestion control design space as backdrop, we introduce a taxonomy of current datacenter load balancers and provide a detailed description of the state-of-the-art datacenter load balancing landscape.

- We also review current datacenter congestion control approaches, their interplay with load balancers as well as how they are evolving in the AI/ML era.
- We develop experimental models to represent WAN and DC-WAN mixed traffic and experimental methodology to run experiments. We also shared what we learned and recommendations for setting load balancer parameters.
- To better understand the performance of load balancing mechanisms, we present a detailed quantitative study exploring the effects of link speed, switch buffer capacity, and congestion notification signaling for intra- and inter-datacenter workload interactions.
- In light of the insights from our study, we introduce our adaptive load balancer and describe its design and implementation in detail, including its main components and mechanisms for congestion sensing and congestion-aware path-selection.
- We conduct a thorough evaluation study of our adaptive load balancer and present its performance against state-of-the-art load balancers.
- We plan to open-source our experimental models and make them available to the community.

§ 1.3 Other Accomplishments and Publications

Another main thrust of work, in addition to the effort to balance the load of the data center network, has been the development of "the Network Simulation Bridge"(NSB) [21]. NSB is a platform-, application- and network simulator-agnostic open-source tool that

allows researchers and developers to more easily integrate network simulation back-ends into their application front-ends, allowing for more holistic modeling of their systems. This work was first published in Q2SWinet'23, followed by a tutorial at ICDCS'25. With the NSF POSE Phase 1 award for NSB, we focused on building the ecosystem and open-source community around the tool, and also participated in the I-Corps training program in Winter'26. We are also actively working on submitting the current architectural design of NSB and its performance evaluations to ACM Middleware's industry track, followed by an extension of that work to ACM TOMACS journal.

I would also like to acknowledge that the thesis reflects a reprint of my prior publications and preprints. Along with these, we also have a manuscript under submission to IEEE Transactions on Cloud Computing journal. We are also working towards a submission to the ACM Middleware industry track, followed by an extension of that work to a journal venue.

Please find a list of all publications listed here :

1. H. S. Kuttivelil, S. Sreenivasamurthy, **L. Krishnaswamy**, N. Bhatia, and K. Obraczka, "Network Simulation Bridge: Bridging Applications to Network Simulators," in *Proceedings of the 19th ACM International Symposium on QoS and Security for Wireless and Mobile Networks (Q2SWinet '23)*, New York, NY, USA, 2023, pp. 39–46. doi: 10.1145/3616391.3622771.

2. **L. Krishnaswamy**, K. Obraczka, and A. Kabbani, "What's Next for Datacenter Load Balancers — A Comparative Performance Study of the State-of-the-Art," in *2024 IEEE 13th International Conference on Cloud Networking (CloudNet)*, Rio de Janeiro, Brazil, 2024, pp. 1–9. doi: 10.1109/CloudNet62863.2024.10815922.

3. H. Kuttivelil, **L. Krishnaswamy**, N. Bhatia, and K. Obraczka, "Network Simulation Bridge: Bridging Distributed Applications with Network Simulation Platforms," in *2025 IEEE 45th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, Glasgow, United Kingdom, 2025, pp. 416–419. doi: 10.1109/ICDCSW63273.2025.00084.

4. **L. Krishnaswamy**, K. Obraczka, and A. Kabbani, "Towards Next Generation Datacenter Load Balancers," *TechRxiv* (preprint), Nov. 2025. doi: https://doi.org/10.36227/techrxiv.176341363.36019194/v1.

5. **L. Krishnaswamy**, K. Obraczka, and A. Kabbani, "A Network Load Balancer for Geographically Distributed Datacenters," in *2026 IFIP Networking Conference (IFIP Networking)*, Lugano, Switzerland, 2026, pp. 1-9. doi: 10.23919/IFIPNetworking70592.2026.11579256.

§ 1.4 Thesis Roadmap

The rest of the dissertation is organized as follows : Chapter 2 provides background on network load balancing, and different congestion signaling mechanisms. Chapter 3 moves to a broader qualitative study that examines the datacenter load balancing and congestion control design space, along with state-of-the-art network load balancers. Chapter 4 focus on the quantitative study covering experimental methodology, workload modeling and experimental setup. Chapter 5 cover the results of the quantitative study. Chapter 6 introduces Balancia, our load balancer and describes its design. Chapter 7 describes Balancia2.0 an proactive congestion sensing load balancer that builds on

Balancia. Finally Chapter 8 concludes with takeaways and looking ahead at future research directions.

Chapter 2

Background

Datacenter network load balancers help provide application flows with adequate QoS by assigning traffic to different network paths to prevent congestion build-up, as well as to improve overall network utilization. Consequently, network load balancing algorithms need to be congestion-aware, which means that they need to make use of different signaling mechanisms to perceive the onset of congestion, i.e., perform congestion sensing, as well as employ techniques to either proactively, or reactively respond to the congestion onset, i.e., congestion management. At their core, load balancer algorithms manage congestion by deciding how application flows share network resources. They use different signaling mechanisms to decide: (1) how to assign application to the available multiple network paths, (2) what granularity level the assignment is performed (e.g., flow, packet, packet aggregates) and (3) where that decision is made. In this section, these different design axes of network load balancers are discussed, providing the necessary background to network load balancing state-of-the-art and the current gaps that our work addresses.

§ 2.1 Load Balancing Granularity

The granularity of the load balancing decision plays a major role in improving workload distribution and therefore in avoiding/mitigating congestion. There are three basic levels of granularity used by current load balancers, as follows:

Flow Level: In flow-based load balancing mechanisms [1, 2, 4, 22, 23], path assignment is done at a flow level, and thus each flow is assigned to a specific path. Upon sensing congestion, load balancers such as FlowBender [2] and PLB [22], reroute flows to less congested paths. The main advantage of such mechanisms is that it only requires end-hosts to participate in path selection and assignment without needing any in-network modifications. However, for low-entropy and high utilization workloads such as AI training workloads these mechanisms have been shown to be not as optimal [24].

Packet Level: Packet-level load balancers can distribute packets belonging to the same flow across multiple paths to effectively use redundant paths [5–8]. However, this may cause packets to arrive out of order which translates into additional overhead at the transport layer. Alternatively, packet-level load balancers such as Vertigo [7] and Drill [5] include a shim layer that can arrange out-of-order packets before passing them to the transport layer. This results in additional overhead. In load balancers, where enqueued packets are extracted to provision load balancing, then there is also the need for hardware support as well.

Flowlet Level: Flowlet-level load balancing achieves a middle ground between per-flow and per-packet path defletion mechanisms [3, 25–27]. Flowlets are defined as a group of packets belonging to the same flow and are detected when the packet gap between two successive packets exceeds a certain threshold, e.g., the maximal latency of all paths [3].

Thus, with flowlets, small bursts of packets can be assigned to a path, without incurring packet significant reordering overhead. However, the performance of flowlet-based techniques rely on precise setting of the inter-flowlet spacing [28].

§ 2.2 Congestion Signals

Congestion-aware network load balancers can infer congestion from a number of signals which can be grouped into two main categories: "end-to-end" or "in-network" signals. The most common form of end-to-end signaling is based on estimated round trip times (RTTs). In-network signaling often relies on Explicit Congestion Notification (ECN) [29] markings as well switch queue occupancy. While packet loss acts as the primary source of congestion inference, incurring a packet loss can be very detrimental, particularly for latency sensitive workloads. Consequently, load balancers try to avoid packet loss events by proactively detecting the onset of congestion.

Explicit Congestion Notification (ECN) [29]: ECN markings enable switches to inform the sending host of congestion onset, even before a packet loss occurs. Whenever a switch experiences congestion, it modifies its IP header to inform the sending host. This is done by setting a bit in the traffic class field of the IP header [9]. The receiver sends back this marked ECN bit to the sender in its ACK packet header. The sender then adjusts its sending rate in response to the congestion signal [9]. Works that make use of ECN markings for their congestion sensing mechanisms include [2, 4, 9, 20, 22].

Estimated Round Trip Time (RTT): Estimated RTT helps to understand end-to-end congestion. Based on how much the measured RTT deviates from the estimated RTT, the extent of congestion in the network can be inferred. This can be used as a signal

either through the help of timing acknowledgments when they arrive at the sender sent by the receiver upon receiving data packets or through explicitly sending out small data packets and measuring the corresponding RTT. Works that make use of estimated RTTs for their congestion sensing include [3, 4, 25].

Switch queue length: Another approach to infer in-network congestion is through switch buffer occupancy and switch queue lengths. Onset of congestion is detected when switches receive more packets than their capacity to enqueue them. Load balancers such as [3, 5–8] use this information to take action to prevent further congestion buildup.

Chapter 3

Network Load Balancing and Congestion Control - A Qualitative Study

§ 3.1 Datacenter Load Balancing and Congestion Control Design Space

Understanding the state-of-the-art of datacenter network load balancers requires a deep dive into the mechanics that underline load balancing decisions. This includes the types of congestion signaling mechanisms available, the timescale in which the decision needs to be made, along with who initiates the actions, and what these actions consist of. Although load balancing performance and effectiveness depend on the kind of congestion signaling received, it is also important to explore the effects that different transport protocols can have on load balancing decisions. In particular, there is a close interplay between different congestion control mechanisms and load balancing decisions, which can be subtle yet very impactful. Fundamentally, congestion control mechanisms operate purely at the transport level, with the goal of trying to combat

congestion at the end hosts, through decisions that also affect the end hosts, such as modifying congestion window size, and sending rate of the application. However, load balancing mechanisms are broadly designed to optimize traffic allocation across existing redundant paths between the source and destination pairs, such that the desired quality of service (QoS) is met and network resources are optimally utilized. The interplay of the two mechanisms happens due to these congestion signals often shared to make both congestion control and load balancing decisions. Table 3.1 maps the datacenter load balancing and congestion control design space. The "Signal Type" column lists different mechanisms that are used to signal congestion. For instance, Explicit Congestion Notification (ECN) markings [29] help inform the sending host of congestion before packet loss occurs. Whenever a switch experiences congestion buildup, it modifies its IP header to inform the sending host that congestion is building up. Probing is another popular congestion signaling mechanism, which can be "active" or "passive". Active probing typically requires sending small packets (or probes) into the network to understand whether congestion is building up based on the probes' return time, inferred through their Round-Trip Time (RTT) measurement. Passive probing approaches, instead of injecting explicit probes, use existing traffic in the network (e.g., TCP segments and their acknowledgments) to detect congestion, e.g., by measuring the RTT. The "Who generates/measures" refers to which network entity is responsible for generating/measuring the congestion signal. All switch generated congestion signals are in-network, while the host generated signals are end-to-end. "Who acts", refers to the network entity that uses the congestion signal to take an action to mitigate congestion build up. "What is the action" describes the kind of action taken by the network entity to balance load and mitigate congestion. With the switch queue length as congestion

signal, the action is a one-hop reroute, which is also referred to as "deflection" and happens at the packet level. However, this results in computational overhead, in order to determine which packets to deflect or drop.

Signal Type	**Who generates/ measures**	**Who acts**	**What is the action**
Explicit Congestion Notification (ECN) marking [29]	Switches	Host	Adjust sending rate/window size
Estimated RTT	Host	Host	Adjust Retransmission Timeout (RTO)
Transmission rate/window size	Host	Host	Reroute flow
Probing	Host	Host	Adjust sending rate/window size
Switch queue length	Switches	Switches	Drop/deflect packets

Table 3.1: Datacenter Load Balancing and Congestion control Design Space

§ 3.2 State-of-the-art of Datacenter Network Load Balancing

While extensive research exists in the area of network load balancers for datacenter networks, there has been little work focused on designing load balancers that explore the coexistence between DC and WAN traffic. We present a qualitative evaluation of some of the state-of-the-art load balancers for datacenter networks which motivates the quantitative analysis we present in Section 4 and Section 5. Understanding the design choices helps shed light on why their performance differ when operating in different, unseen environments.

The taxonomy in Table 3.2 provides a classification of state-of-the-art network-level datacenter load balancers by identifying their main distinguishing features. "Transmission Unit" refers to the unit of information used to make load balancing decisions. These can be at the "Flow" level, "Flowlet" level or "Packet" level. A flow refers to the continuous stream of packets belonging to the same application. Flowlets are defined as a stream of packets belonging to the same flow, where the packet gap between two successive packets does not exceed the maximal latency of all paths [30]. Thus, with flowlets, small bursts of packets can be assigned to a path, minimizing packet reordering overhead. With packet level decisions, load balancers can make more fine-grained decisions, however, with the drawback of incurring packet reorderings overhead. "Who decides" refers to which network entity makes the load balancing decision. This decision, which has to do with "path selection" for the "transmission unit" can be done at the end hosts or at the switches. Additionally, it can also be a joint decision by the host and switches. The "Additional Support" column indicates whether the load balancer

requires mechanisms from other layers, e.g., hardware changes in order to support the full functionality of the algorithm, or introduction of an additional shim layer to handle out-of-order packets before sending them to the upper layer. "Congestion Signal" specifies what congestion indicators the load balancer uses. Finally, "Path Selection" summarizes how the load balancer chooses the best path to distribute load evenly.

3.2.1 Equal-Cost Multi-Path Algorithm (ECMP) [1]

Equal-Cost Multi-Path (ECMP) is one of the earlier approaches to network-level load balancing. It uses a hash function to determine a flow's outgoing path. ECMP's hash function uses a packet's header information, i.e., source and destination IP address and port numbers, to generate a hash key. The key is then used to find the corresponding path in the hash table. This results in packets from the same flow taking the same path, since these packets contain the same packet header information and thus generate the same hash key and get routed through the same path. Overall, traffic is distributed equally across all of the possible paths since, a particular flow is equally likely to hash to any viable path. In other words, ECMP is a congestion-agnostic load balancer as it does not adjust to network load dynamics. ECMP is quite simple and is known to perform well in lightly loaded networks. Its routing decisions can be made quickly, with low compute overhead, and no need for special packet tagging, addressing packet reordering or specialized hardware. However, with ECMP, path collisions may result in uneven load distribution and frequent congestion events at relatively light utilization levels. Several load balancing works such as [2–8, 22, 23, 25–27, 31–33] have highlighted this major drawback with ECMP and devised more optimal load balancing designs.

3.2.2 FlowBender [2]

FlowBender is a flow-level load balancer that aims at guaranteeing congestion-aware decisions. Designed as an end-host-based solution that does not involve any switch modifications. As described in the "Congestion Signal" and "Path Selection" columns in Table 3.2, FlowBender uses explicit congestion notification (ECN) to detect congestion. Based on the ECN markings, FlowBender reroutes the flow by modifying the hash function used for path selection. A version of Flowbender, that is production ready, "PLB" [22] has been successfully deployed at Google and has been shown to significantly improve overall network utilization and reduce tail latencies. One of Flowbender's drawbacks is the possibility of rerouting a flow multiple times in the attempt to find a non-congested path.

3.2.3 CONGA [3]

CONGA is a distributed load balancing technique that makes load balancing decisions at the flowlet level. As seen in the discussion related to Table 3.2, flowlets are a stream of packets belonging to the same flow, which can be detected when the packet gap between two successive packets exceeds the maximal latency of all paths [30]. In particular, this maximal latency is a function of the network's link delays. CONGA makes use of VxLAN [34] technology to carry path utilization and infer global congestion information. CONGA is shown to perform optimally even in the presence of link failures. However, it works best for 2-tier Leaf Spine topologies since it relies on obtaining global congestion information. This is done with the help of each of the leaf switches maintaining a per-destination congestion table. With 2-tier topologies, each leaf switch

has a complete view of the direct path to the destination via the spine switches. With 3-tiered networks, this knowledge of congestion is not directly obtained, thus affecting the overall performance of CONGA. Further, as noted in Table 3.2, CONGA requires modifications to the switches' hardware, in particular the leaf switches, in order to maintain and update the congestion tables. This makes it difficult to deploy in production datacenters.

3.2.4 Hermes [4]

Hermes is another distributed load balancing technique. It gathers dynamic information about the network using active probing mechanisms and uses both RTT measurements and ECN markings to learn about congestion in the network. As noted in the "Added Support" column of Table. 3.2, one of Hermes' strengths is that it requires no additional support from other layers. Hermes only reroutes packets of a flow when congestion indicated by both RTT and ECN exceeds a set threshold level. However, this results in having to fine-tune a number of parameters, such as the thresholds to infer congestion and probing intervals in order to sense path conditions effectively and provide optimal load balancing decisions.

3.2.5 DRILL [5]

Distributed Randomized In-Network Localized Load Balancing (DRILL) employs a "micro" load balancing technique. More specifically, DRILL makes per-packet level decisions using local information available at each switch to distribute load as evenly as possible on a microsecond timescale. It makes load-aware forwarding decisions local to

each switch. This knowledge requires hardware support through switch modifications, as seen from Table 3.2. Additionally, this is prone to more packet reorderings which isn't desirable, as well as requiring stateful switch processing [22].

3.2.6 DIBS [6]

DIBS or Detour Induced Buffer Sharing uses per-packet deflection as its load balancing approach. As described in the "Path Selection" column of Table 3.2, whenever a switch receives more packets than it can enqueue, it detours the excess packets to a randomly selected port with enough buffer space. DIBS simple deflection technique performs well under light loads, but quickly degrades as load increases.

3.2.7 Vertigo [7]

Vertigo is one of the more recent network load balancers and focuses on mitigating microburst induced congestion [7]. Vertigo employs an in-network, host-assisted, selective deflection solution. While DIBS performs simple packet deflections, Vertigo was developed to deflect packets based on congestion information of flows. In particular, packets that potentially cause persistent congestion in the network are selected for deflection, thereby reducing the reordering that deflection in general can cause. However, this requires extracting packets already enqueued in the switch buffer, which isn't supported in current datacenter switches [8].

3.2.8 Preemptive Deflection [8]

To address Vertigo's limitation, Preemptive Deflection was introduced as its successor. As outlined in the "Path Selection column of Table 3.2, Preemptive deflection predicts and deflects packets. In particular, Preemptive deflection uses an admission control technique that evaluates if an incoming packet should be deflected or enqueued. This decision is made based on priorities assigned to the packets based on the current packet, enqueued packets, and the destination's queue occupancy. Based on this, packets that are more prone to cause long-lasting congestion are deflected.

3.2.9 REPS [9]

Recycled Entropy Packet Spraying for Adaptive Load Balancing and Failure Mitigation (REPS) is a per-packet load balancing algorithm designed to optimize network utilization along with rapid recovery from link failures. One of its biggest strengths, as seen in the "Path Selection" column of Table 3.2, is caching good, uncongested paths, in a circular buffer. REPS makes use of ECN markings to re-route flows to less congested paths. However, REPS requires out-of-order support from the transport layer.

3.2.10 Ethereal [10]

Divide and Conquer Network Load Balancing in Large-Scale Distributed Training (Ethereal) specifically targets deep neural network (DNN) training taking place in large GPU clusters. As recorded in Table 3.2, being a flow-level load balancer, Ethereal avoids using packet spraying as the de-facto mechanism for large-scale distributed training in production datacenters. Instead, it proposes and validates the use of source

routing based load balancing to optimize performance for large-scale distributed AI/ML training. Ethereal operates at two layers: load balancing decisions (path assignment and flow splitting) are made proactively at the application layer before transmission, while congestion control and failure recovery remain at the transport layer. In particular, their key design choices are motivated by the following assumptions: 1) The topology structure along with link capacities are known to Ethereal 2) The flow sizes of AI/ML training workloads are uniformly sized within each training step, and 3) Flows can be split upon arrival, into "subflows" to control load balancing from the application layer [10].

3.2.11 Discussion

Our qualitative exploration of the current datacenter load balancing state-of-the-art reveals that most current load balancers have been designed and optimized for specific workloads and datacenter topologies. As the diversity of datacenter applications increases resulting in the coexistence of workloads with different characteristics and requirements, having tailored, custom load balancing solutions will not scale as datacenter workloads and topologies evolve. In the next section, we discuss recent trends in congestion control and load balancing addressing modern and future datacenters.

Table 3.2: Taxonomy of State-of-the-Art Datacenter Load Balancers

Load Balancer	**Trans. Unit**	**Who decides**	**Additional Support**	**Congestion Signal**	**Path Selection**
ECMP [1]	Flow	N/A	No	Congestion agnostic	Hash function
FlowBender [2]	Flow	Host	No	Explicit Congestion Notification (ECN) marking	Input to the hash function modified based on ECN marking threshold
CONGA [3]	Flowlet	Edge switch	Hardware support	Sending rate	with global awareness of all paths
DRILL [5]	Packet	Switch local	Shim layer for out-of-order arrivals	Switch queue length	Local to each switch
DIBS [6]	Packet	Switch local	Hardware support	Switch buffer size	detour to a switch with buffer space
Hermes [4]	Flow	Host	No	RTT duration and ECN	Cautious rerouting only for congested paths
Vertigo [7]	Packet	Switch, with host-assistance	Hardware support, shim layer for out-of-order arrivals	Remaining flowsize from transport protocol and switch queue occupancy	Drop or deflect, based on flowsizes, and contribution to persistent congestion
Preemptive Deflection [8]	Packet	Switch	Shim layer for out-of-order arrivals	switch queue occupancy	Predict and deflect packets that can cause congestion, before entering the switch queue
REPS [9]	Packet	Host	No	ECN marking	Caching of good, uncongested paths using a circular buffer, and reusing these paths when congestion is encountered
Ethereal [10]	Flow	Source host	Minimal changes to NIC hardware	ECN marking, timeouts and Negative Acknowledgements (NACKs)	Pre-determined path selection at flow arrival time, based on source routing

§ 3.3 Taxonomy of State-of-the-Art Datacenter Load Balancers

§ 3.4 Evolving Trends in Datacenter Load Balancing and Congestion Control

With the growing deployment of multi-location datacenters and the increasing AI/ML workloads, we observe a growing trend towards distributing AI/ML training across datacenter networks [20]. This results in increased prevalence of inter-DC workloads. While to the best of our knowledge, there exists no prior work focused on designing load balancing solutions targeted at DC-WAN workloads. With the congestion control and load balancing design space from Section 3 as a backdrop, in Section 3.4.1 we review some recent efforts that focus on congestion control mechanisms to address the coexistence of DC and WAN traffic. In Section 3.4.2 we also review congestion control mechanisms developed for purely DC traffic, that starts looking into the possible interactions between congestion control and load balancing. We examine these recent efforts at congestion control, and their assumptions about an underlying load balancing algorithm and how that can impact the overall network performance. We conclude this section by discussing where these mechanisms fall short, and some open research questions they raise.

3.4.1 Congestion Control for DC-WAN interaction

Works such as Annulus [18], Gemini [19] and Uno [20] highlight the growing trend of inter-datacenter networks that results in DC-WAN traffic coexistence.

In Annulus [18], a dual control loop makes congestion control decisions to explicitly address the diverse QoS needs of mixed DC-WAN traffic.

Gemini [19] uses a combination of Explicit Congestion Notification (ECN) and delay signals to make congestion control decisions. These works address how congestion control mechanisms can adapt to the DC-WAN scenario. However, they do not examine the problem of optimally allocating DC and WAN traffic across multiple paths to deliver the required QoS for each type of traffic. This further highlights the gap in state-of-the-art approaches and the need to re-examine network load balancers for evolving datacenter environments. More recently, Uno [20] argues for the need to co-design congestion control and load balancing algorithms for intra-, and inter-DC communications. Uno offers a modular framework, consisting of unoCC and unoRC ; where the former is responsible for integrating congestion control for intra-, inter-DC traffic, while unoRC focuses on delivering reliable load balancing. Given that WAN traffic is bounded by its long propagation delays, even a single packet loss is significantly impactful and causes large latency penalties. Uno's design choice to providing reliability is at the network level, using erasure coding as part of their load balancing mechanism. However, with the use of erasure coding, comes the cost of additional compute overhead. It is important to note that Uno's modular design makes it easy to pair unoCC with alternative load-balancing schemes that are transport-agnostic and lightweight.

3.4.2 Datacenter Congestion Control in the AI/ML Age

Mechanisms such as SMaRTT-REPS [35] and MCC [36] acknowledge the impacts that network load balancing can have on transport congestion control, and devise ways to manage it from the congestion control design choices. For example, SMaRTT-REPS [35] couples a sender-based congestion control algorithm with an endpoint-driven per-packet load balancer. SMaRTT combines ECN marking and RTT measurements to react to congestion quickly, and introduces QuickAdapt, a mechanism that converges to fair-share bandwidth within roughly one RTT by measuring the bytes acknowledged during a congestion event. It works jointly with REPS [9] as its load balancing mechanism. MCC [36], is a congestion control mechanism designed specifically for AI workloads. They state a key observation they make that when using existing congestion control mechanisms with per packet load balancing schemes, the congestion mechanisms tends to overreact to per packet signals such as ECN markings [36]. Their approach to this, is a solution still targeted at the congestion control level, where they shift the granularity at which the congestion mechanism reacts. From all of these works, there are some interesting future research directions opening up. Specifically, if the transport layer can still be strictly end-to-end and just deliver reliability, while network load balancing can take care of infering congestion, adjusting sending rates and manage path allocations. Given these questions, it is important to acknowledge the broader efforts going into the redesign of the transport layer in itself, in order to better service for the AI-era workloads. The Ultra Ethernet Consortium (UEC) has released a v1.0 specification [37] defining a next-generation Ethernet transport for AI and HPC workloads. There is a lot of interesting discussion on packet trimming, per-packet spraying, and selective acknowledgment as part of the specifications. However, it leaves congestion control and

load balancing policies to individual implementations.

3.4.3 Discussion

Studying the state-of-the-art load balancing and recent trends in congestion control mechanisms uncovers the fact that congestion control mechanisms, which are often designed as a transport layer function, have been typically decoupled from and agnostic to network-level load balancing approaches.

MCC [36] is one of the first works to acknowledge that such an assumption can lead to conflicting performance goals from congestion control and load balancing, which may result in suboptimal outcomes. Uno [20] adopts an interesting design choice where reliability is also pushed towards the network, away from the transport layer. These key observations, namely: the decoupling between transport-level congestion control and network load balancing, along with the drawbacks of tailored network load balancing approaches raise the following problem: **Can we design a network load balancer that performs effectively for both intra- and inter-datacenter traffic, remains transport-agnostic, and adapts to the evolving trends in datacenter network architectures and workloads?** To help address this problem, we formulate the following questions:

- How do state-of-the-art network load balancers perform in scenarios where data-center workloads are distributed across multiple geo-distributed datacenters?
- How do current network load balancers perform when they co-exist and interact with congestion control mechanisms at the transport level?
- Given the design space of congestion control and load balancing, what are the tradeoffs of a purely end-host or purely in-network strategy and can we find a

balance, given how changing the endpoints is easier than making modifications to the network?

Chapter 4

Network Load Balancing and Congestion Control - A Quantitative Study

To answer the questions framed in Section 3.4.3 above, we conduct a quantitative study that evaluates the performance of the current state-of-the-art in datacenter network load balancing and their sensitivity to network parameters, and transport-level congestion control parameters.

More specifically, our study includes: 1) Reproducing performance of state-of-the-art datacenter load balancers using scenarios they were designed for to validate our experimental setup; 2) Modeling intra-datacenter and inter-datacenter workloads to evaluate how current datacenter load balancers will perform in such scenarios; 3) Understanding the sensitivity of current network load balancers to their own parameters, datacenter network topologies, and the choice of transport-layer congestion control functions.

4.0.1 Experimental Methodology

Experimental Infrastructure

All simulations were carried out using the OMNeT++ simulator version 5.6.2 [38] and INET network framework version 4.3.2 [39]. We use the open-source Vertigo codebase [7] to evaluate performance of the load balancers for validation with DC traffic. The CloudLab [40] platform was used to carry out the experiments.

Topology Modeling

Intra-Datacenter Topology We use two standard datacenter topologies, namely, Fat Tree and Leaf Spine [7]. Leaf Spine is a 2-tier fully connected topology where the bottom tier, or leaf switches connect to the end hosts on one end and create a fully connected mesh with the upper-tier, or spine switches. Fat Trees are more recent datacenter topologies organized as a 3-tier hierarchy, consisting of Top-of-Rack (ToR) switches, aggregate switches and core switches. A pod consists of a set of aggregate switches, ToR switches, along with the hosts connected to the ToR switches. The number of switches and hosts that are part of a pod is customizable. While the Leaf Spine network provides full connectivity, making it cost-effective and simple to deploy, the Fat Tree topology offers better scalability through the provisioning of additional pods as needed. These two topologies have been used by most of the works covered in Section 3.1. The DC network experiments were performed to validate our implementation of the state-of-the-art load balancers we consider in our study using their recommended operational parameters, network topology and traffic model. These results are discussed in Section 5.0.1.

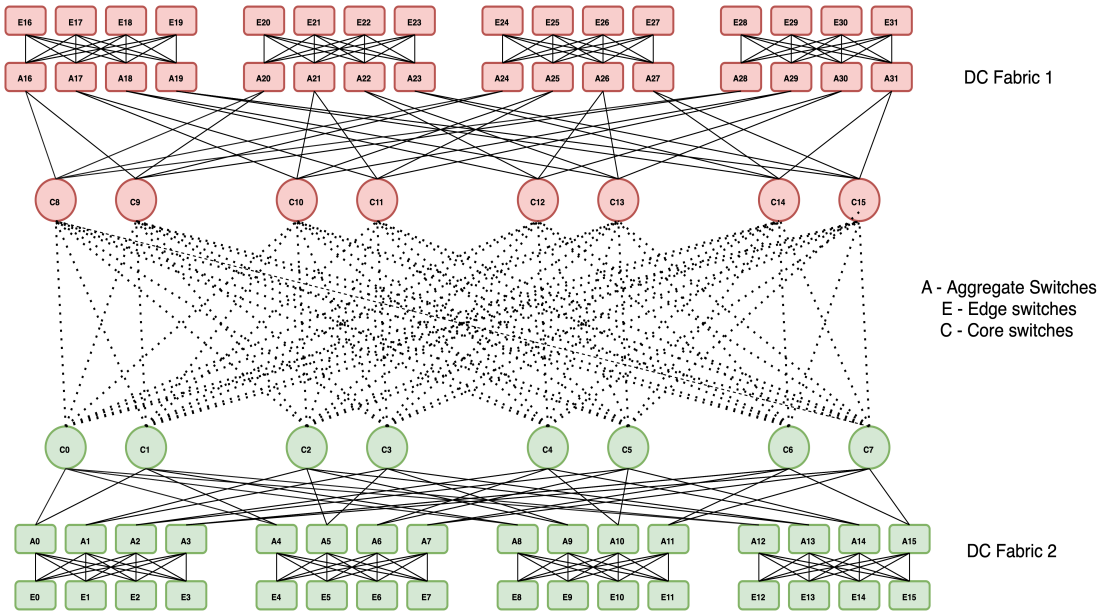

Figure 4.1: Inter-Datacenter Network Topology

Inter-Datacenter Topology Validating load balancer performance on geo-distributed datacenters calls for using a network topology model that considers distributed datacenters, with long haul interconnects. Additionally, an important practical consideration is to ensure that simulation experiments finish in reasonable time. With these goals in mind, we designed and modeled the inter-datacenter topology illustrated in Fig. 4.1, inspired by works such as Annulus [18] and Gemini [19], and our interactions with datacenter practitioners. In particular, Annulus [18] makes use of a “border router" to connect the datacenter fabric to the wide area network. In Gemini [19], all experiments were performed on hardware testbeds that contained exactly one inter-datacenter link. We propose the inter-datacenter topology model illustrated in Figure 4.1, which tries to adequately balance between using a realistic datacenter fabric and enabling our experiments to complete in reasonable time. In our inter-datacenter topology model

(see Figure 4.1), the network consists of two datacenter fabrics interconnected through their core switches. Each datacenter fabric has four pods, each of which with four edge switches, four aggregate switches and 16 servers, thereby the whole network consists of 128 servers in total. Core switches in one datacenter fabric are connected via inter-datacenter links (*also referred to as* WAN links) to the core switches of the other fabric. In order to have the WAN links oversubscribed, each core switch in a given datacenter fabric connects to every core switch in the other datacenter fabric, thus forming a fully connected mesh. We set all links within a datacenter fabric to a uniform speed of 100Gbps. The links between the core switches are set to 10Gbps. The links between the top-of-rack (ToR) switches and the server have a delay of 1μs , while the links between the aggregate switches and ToRs have a delay of 2μs. The WAN links, between the core switches are set to have a delay of 1ms [18]. The bottleneck links for DC traffic are the links between aggregate and core switches. The bottleneck links for WAN traffic are the inter-DC links, or WAN links. The flow inter-arrival times are scaled to vary the supplied load in all the experiments. In our comparative study of the different datacenter load balancers, we explore how having heterogeneous link speeds at the different levels of the topology affects performance.

4.0.2 Workload Modeling

Intra-Datacenter Workloads

We make use of popular realistic workloads in the literature such as web-search [41] and data mining [42], which have been used in works highlighted in Section 3.1. We also use synthetic workloads to validate the correctness of our experiments, which was

particularly important for the proposed Inter-datacenter network and for running purely inter-datacenter traffic (also referred to as WAN traffic) experiments, and DC-WAN mixed traffic experiments.

For the synthetic workloads, we use a simple bimodal distribution, that overcomes the issue of long run times while still capturing the heavy-tail distribution of typical datacenter network workloads. It consists of short flows that are $50KB$ in size and large flows that are $1MB$ in size. These were chosen, in order to maintain reasonable simulation times, while capturing the heavy-tailed distribution of DC workloads. In order to mimic the heavy-tailed distribution of DC workloads, 90% of the total flows are $50KB$, while the remaining 10% are of $1MB$. We generate around $51,200$ flows. This value was chosen for the number of flows, given that it was sufficient to be representative of the heavy-tailed distribution.

Inter-Datacenter Workloads

Also referred to as WAN workload, since they take place over the long haul, inter-datacenter (WAN) links. In order to capture the high utilization of WAN links in a inter-datacenter network, we use large flows of 10MB to mimic WAN Traffic. For these experiments, we generate around 12,800 10MB flows.

DC-WAN Mixed Workloads

To the best of our knowledge, very little work exists on modeling the coexistence of DC-WAN traffic. This is the coexistence of intra-datacenter, and inter-datacenter workloads. Thus, our workload modeling is also an important contribution of this work. To model the coexistence of DC and WAN traffic in geo-distributed datacenters,

we use both synthetic as well as realistic workloads. For the synthetic workloads, we use the bimodal distribution described under Section 4.0.2 to represent DC traffic within DC Fabric 1 and DC Fabric 2. For the WAN traffic, which are the flows transmitted over the WAN links between the two DC fabrics, we model a 1:5 flow ratio of WAN to DC traffic, which is based on recommendations from Annulus [18], Uno [20]. The flow inter-arrival times for DC and WAN traffic are drawn from independent Poisson distributions. The average arrival rate parameter of the Poisson distribution for each traffic type is determined based on the bottleneck link capacity, average flow size, and oversubscription ratio. For the realistic workloads, we sample flow sizes from the cumulative distribution function of web search workloads [41], to represent DC traffic within DC Fabric 1 and DC Fabric 2. For the WAN traffic, which is transmitted over the WAN links between the two DC fabrics, we model a 1:5 flow ratio of WAN to DC traffic, which is based on recommendations from Annulus [18].

4.0.3 Experimental Setup

Transport Protocol Parameters: DCTCP has been the recommended transport protocol for datacenter networks. It is also used in most related works such as [3],[6],[5],[7],[8]. Thus, we use DCTCP [43] as the transport protocol for all our experiments. As recommended in the DCTCP paper [43], the per-port switch buffer capacity is set to be equal to the bandwidth delay product of the specific datacenter network. Both the low ECN marking K_{min} and high ECN marking K_{max} thresholds are set to be equal [43]; to one-third of the per-port switch buffer capacity . The values used for these parameters in our experiments are described in Section 5 and summarized in Tables. 5.1, 5.3.

WAN workloads are characterized by long haul links and high bandwidth, in com-

parison to DC workloads. Additionally, TCP Cubic also does not make use of any in-network signaling for congestion conference, and purely relies on packet loss. This design where the only congestion signal is end-to-end, is in contrast to DCTCP, which uses in-network ECN markings as well. Thus in addition to DCTCP, we evaluated purely WAN workloads using TCP Cubic [44]. For both DCTCP and TCP Cubic, we set the initial congestion window size to be 10 packets. Given the high bandwidth-delay product associated with the high link speeds, in order to maximize throughput, window scaling is also enabled, thus preventing the advertised window from creating a bottleneck.

Load Balancer Specific Parameters: Because Vertigo and DRILL employ a reordering layer to mitigate the effects of packet reordering caused by their route deflection technique, they use a parameter known as the *ReOrdering Timer*. This timer is defined as the "maximum duration of time that the ordering component waits for a single delayed packet before starting to process out-of-order packets" [7]. Vertigo recommends setting this value to the time it takes for a single packet to traverse a network with almost full buffers. Thus, this recommendation is followed for the purely DC workload experiment. For the purely WAN and mixed DC-WAN workload experiments, a more detailed explanation is provided in Sections 5.0.2 and 6.2.2.

Performance Metrics: In our results, we report mean and 99% flow completion times (FCTs), both of which are defined in Section 5. We establish the *standalone Flow Completion Times (FCT)* by simulating a single flow in the network, and measuring the time taken for its completion, which doesn't account for queueing times. This helps to establish a baseline lower bound to validate the results.

Chapter 5

Quantitative Study Results

In this section, we examine the performance of state-of-the-art load balancers under three different workload scenarios, namely: purely DC workloads, purely WAN workloads, and DC-WAN mixed workloads. In the DC-WAN mixed workload experiments, we also dive deeper exploring different datacenter topologies, specifically their switch buffer capacity and different link speed configurations, and how they affect load balancer performance. A subset of the load balancers studied were chosen for our quantitative study. Our selection was based on the reported results in the chosen load balancers' original papers, which reported that these load balancers outperform other candidate load balancing choices [2, 5–7].

The results reported in the following sections show the mean and 99% FCT as a function of the load across the bottleneck links. Examining 99% FCT helps establish how long the last 1% of the workload takes to complete, which is critical especially for short flows and time-sensitive applications. The mean FCT helps to understand the overall effectiveness of a given load balancer.

5.0.1 DC Workloads

In these experiments, we validate the performance of the load balancers by simulating purely DC workloads within a single datacenter using the DC Fabric1 topology as illustrated in Figure. 4.1. We use the synthetic workloads described in Section 4.0.2. The values for all the parameters used in these experiments are summarized in Table. 5.1 and are set following the guidelines discussed in Section **??**. They also reflect the recommendations presented in Vertigo [7]. The *standalone FCTs*, as defined in Section **??**, for short- and long flows are summarized in Table. 5.2. Intra-ToR flows refer to flows between hosts under the same ToR switch, while intra-pod flows are flows within the same pod, and inter-pod flows refer to flows across different pods but within the same datacenter network. Each experiment was run two times by varying the random seed for generating different inter-arrival times and destination servers for the workloads. These runs were sufficient to observe the trends as reported in Vertigo [7]. In particular, we have added the standard deviation bars across these two runs for these experiments.

Table 5.1: Experimental Parameters: DC Workloads

Parameter	**Value**
Switch Per-Port Capacity	250KB
ECN Marking Threshold	56 packets
Initial Congestion Window Size	10 packets
Initial RTO	1s
Minimum RTO	10ms
Vertigo and DRILL ReOrdering Timer	0.000360s

Results and Discussion: The graphs in Figures 5.1 and 5.2 show the 99% tail performance, for short (50KB) and long (1MB) flows. We observe that Vertigo performs better than all the other load balancers for the short flows, as expected. Vertigo's

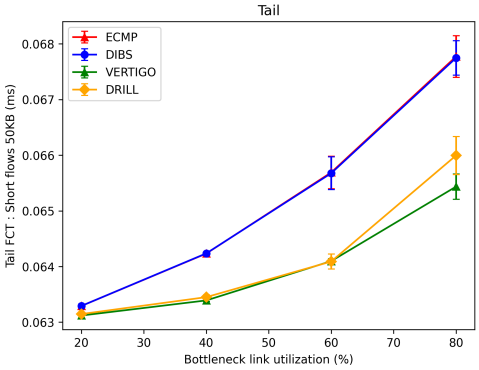

Figure 5.1: DC Workloads - 99% FCT for 50KB flows

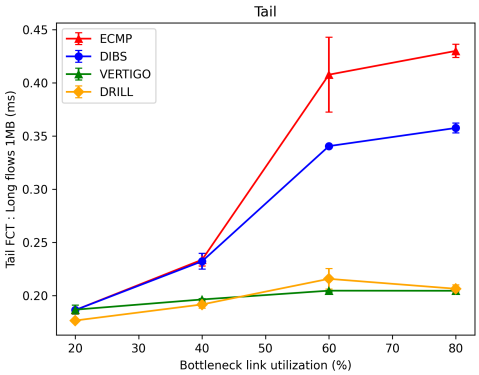

Figure 5.2: DC Workloads - 99% FCT for 1MB flows

Table 5.2: Standalone FCT Results: DC Workloads

Flow Type	**FCT**
50KB : Short Flows	
Intra-ToR flow	12.8 µs
Intra-pod flow	36.8 µs
Inter-pod flow	60.8 µs
1MB : Long Flows	
Intra-ToR flow	94 µs
Intra-pod flow	129 µs
Inter-pod flow	171 µs

superior performance for short flows is due to its ability to react to congestion by selectively deflecting packets that contribute to persistent congestion. For long flows, ECMP's and DIBS' performance degrades with load due to ECMP's congestion agnostic characteristics and DIBS' simple deflection mechanisms which causes excessive packet reorderings. Vertigo's superior performance is also observed in the case of long flows due to its selective deflection mechanism, which avoids packet drops, and thus improves FCT. **These results match the performance trends observed in previous work** [7] **and serve to validate our implementation of the network load balancers.**

5.0.2 WAN Workloads

In this set of experiments, we evaluate load balancer performance under purely WAN traffic across the two DC Fabrics in the WAN datacenter network illustrated in Figure. 4.1. To the best of out knowledge, there has been no prior datacenter load balancer study that reported evaluations with purely WAN workloads. The values for the parameters used in these experiments are listed in Table. 5.3. Given that the WAN link has a propagation delay of 1 ms, the initial RTO for these experiments is set to 100 ms. The

standalone FCT for these experiments is around 20.8 ms.

Table 5.3: Experimental Parameters: WAN Workloads

Parameter	**Value**
Switch Per-Port Capacity	$1MB$
ECN Marking Threshold	223 packets
Initial Congestion Window Size	10 packets
Initial RTO	$1s$
Minimum RTO	$100ms$
Vertigo1 and DRILL1 ReOrdering Timer	$0.002038s$
Vertigo2 and DRILL2 ReOrdering Timer	$0.000038s$

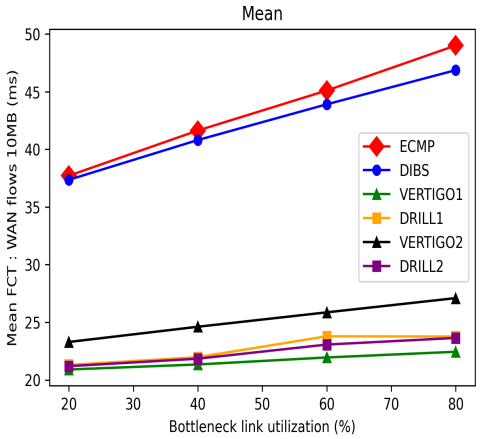

Figure 5.3: WAN Workloads - Mean FCT for 10MB flows

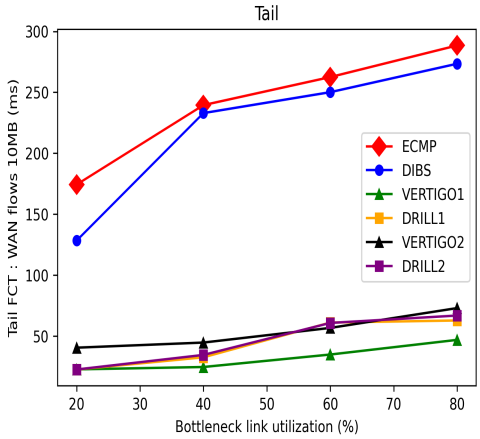

Figure 5.4: WAN Workloads - 99% FCT for 10MB flows

Results and Discussion: From Figures 5.3 and 5.4, we observe that ECMP's and DIBS' performance degrade as the supplied load across the core links increases. ECMP, being congestion agnostic, causes collisions due to multiple flows being assigned the same path under high load. DIBS' approach to packet deflection causes packet reordering and significant performance degradation under high loads. With DRILL and Vertigo, there are some interesting takeaways to be discussed. As the graphs in Figures 5.3 and

5.4 illustrate, **purely WAN workloads highlight Vertigo's and DRILL's sensitivity to the Reordering Timer, affecting how well they load balance WAN flows. It is also interesting to note that, when we disable the Reordering Timer, the performance of both Vertigo and DRILL suffer resulting in significantly higher FCTs.** This sensitivity to the Reordering Timer is also observed under DC-WAN mixed workloads as reported in the next section; this is due to mixed workloads' significant differences in RTTs and link speeds.

5.0.3 WAN Workloads with TCP Cubic

As noted in Section 4.0.3, TCP Cubic is a variant of TCP, developed to work best with networks having large bandwidth delay products, and traversing long haul links. Additionally, another feature of interest is its congestion control design, where it relies on just packet loss (end-to-end signal) for infering congestion. Given this design, there is full dependence on the network load balancers to detect congestion early on, and optimize path allocations. For these experiments, we evaluate the performance of the load balancer under purely WAN traffic in the two DC Fabrics in the WAN datacenter network illustrated in Figure. 4.1. We present in Table. 5.4, the relative reduction of mean FCT and 99% FCT (%) over ECMP, at different utilizations of WAN links. We use ECMP as the baseline to show the relative performances, as it is one of the more earlier, and simpler approaches.

Results and Discussion: TCP Cubic's congestion inference is purely based on packet loss, while DCTCP makes use of ECN markings as well. Thus, in a purely loss based congestion inference model, ECMP's congestion agnostic behavior leads to very high mean, and 99% FCTs, with all load balancers outperforming ECMP, as seen in Table. 5.4.

Table 5.4: Percentage reduction in FCT relative to ECMP

Load Balancer	**20%**	**40%**	**60%**	**80%**
Mean FCT				
DIBS	97.8	96.2	94.1	91.8
DRILL	98.9	98.5	98.1	97.5
Vertigo	99.1	98.8	98.6	98.4
99% FCT				
DIBS	55.3	38.9	36.2	30.6
DRILL	99.2	99.1	88.1	78.0
Vertigo	99.2	99.2	99.2	99.1

When comparing DRILL, DIBS, and Vertigo, Vertigo is seen to perform consistently better than rest of the load balancers, similar to results with DCTCP. Additionally, it is also observed that DRILL's performance suffers from significantly higher 99% tail FCT when compared with Vertigo. In particular, in DRILL, each switch has its own local congestion view, based on its buffer occupancy. However, TCP Cubic has no information about per-hop switch queues and thus, congestion is only inferred after a packet loss has occurred. While both DRILL and Vertigo decide purely in-network at the switch level, as seen in Table. 3.2, Vertigo also makes use of the knowledge about a flow's contribution to persistent congestion in order to determine when to drop or deflect a packet. **Thus, having additional congestion inference mechanisms makes Vertigo transport-agnostic, where it is also able to deliver optimal performance with protocols such as TCP Cubic which uses purely loss as a congestion indicator.**

5.0.4 Parametric Sensitivity Analysis

In this section, we present a detailed experimental analysis of load balancer sensitivity to network parameters, in particular switch buffer capacity (Section 5.0.4) and link speeds (Section 5.0.4). We have presented the 99% FCT values, at 80% supplied DC load and 80% WAN load for these analysis, since observing 99% FCT is critical to determine how the load balancers perform, especially at peak utilization - given that the slowest flows impact performance for all applications.

Impact of Switch Buffer Capacity

The main objective with these experiments is to understand the impact of using shallow and deep switch buffers for the datacenter topology illustrated in Figure 4.1, with the DC-WAN mixed workloads. With DC-WAN mixed workloads, the challenge in switch buffer design arises from the competing needs of DC-, and WAN traffic. Specifically, the bandwidth delay product of WAN traffic is significantly larger than that of DC traffic, due to the longer RTTs. As also noted in the work [20], while commodity switch buffers have increased in size during the years, they are still quite small, especially switches deployed within a single datacenter fabric. In order to explore the impact of switch buffer capacity on the performance of load balancers, we examine performance under two conditions, namely “homogeneous-” and ”heterogeneous switch capacity”. In particular, in the “homogeneous switch capacity” experiments, the buffer capacity of all intra-datacenter switches, i.e., edge-, aggregate- and core switches are uniformly set to 1MB per-port as described in Section 4.0.3.

For the workloads used in these experiments, we model our synthetic workloads as a bimodal distribution. As described in Section 4.0.2, For traffic that stays within the

DC fabric, we use 50KB and 1MB flows and use 10MB flows for traffic going across the WAN. This was designed to model the heavy-tailed distribution of intra-DC traffic, along with the high utilization of the WAN links. To compare against this setup, we use a heterogeneous switch buffer scenario, where the edge and aggregate switches in the DC fabric have a 250KB per-port capacity, while the core switches have per-port capacity of 1MB. The choice of 250KB per-port capacity was motivated by the design choices discussed in Section 4.0.2. We use the synthetic workloads described in Section 4.0.2. All parameters except for the switch buffer capacities used in these experiments are the same as those of purely WAN workloads which are summarized in Table 5.3.

Table 5.5: Shallow DC Buffers with Deep Core Buffers - 99%FCT using 1MB Flows

Load Balancer	**1MB Switch Capacity in DC Fabric**	**250KB Switch Capacity in DC Fabric**
ECMP	0.207755ms	0.266304ms
DIBS	0.207742ms	0.238908ms
DRILL	0.191276ms	0.217200ms
VERTIGO	0.203590ms	0.206585ms

In general, we observe that the use of heterogeneous switch buffers in the topology leads to an increase in FCT for all load balancers. Specifically, under DC-WAN mixed workloads, we observe increased packet drops at the edge and aggregate switches within the DC fabric. This supports the expectation that shallow switch buffers lead to more packet drops. As such, the main advantage of using shallow buffers is its cost effectiveness. However, achieving adequate performance when using shallow switch buffers requires careful design of congestion control mechanisms. Although deep switch buffers show quantitative evidence of better performance with reduced FCTs, they are expensive and thus not cost-effective. This underlines the **importance of having an**

adaptive load balancer that can work in hand with congestion control algorithms by selecting the right congestion notification signaling mechanism to react in a timely manner, in order to deliver adequate performance even with shallow switch buffers.

Impact of Heterogenous Link Speeds

The main objective here is to understand the impact of using heterogeneous link capacities in datacenter fabrics. In our experiments, we use the datacenter topology illustrated in Figure4.1 and the DC-WAN mixed workloads described in Section 4.0.2. Except for link speeds, all parameters used in these experiments are the same as those summarized in Table 5.3. The homogeneous link speed setup refers to when all the links in the DC fabric are set to 100Gbps, while the heterogeneous link speed setup has links between the aggregate- and core switches reduced to 50Gbps. The links between the core switches are set to 10Gbps in both sets of experiments. These parameter choices were motivated by the need to explore how different network provisioning of the DC versus WAN links affect load balancer performance, while ensuring that the oversubscription ratios at different levels in the network were still realistic of today's datacenters. Oversubscription ratio can be described as the ratio between the total downlink bandwidth and the total uplink bandwidth for a switch [45].

Table 5.6 shows the 99%FCT performance of the load balancers in the homogeneous and heterogeneous link speed experiments. We observe that ECMP and DIBS contribute to increased FCTs when the links between the aggregate and core switches are reduced to 50Gbps. With these reduced link speeds, any congestion build-up at the aggregate or core switches cannot be handled by ECMP, since it is a per-flow congestion-agnostic

Table 5.6: Sensitivity to Link Speeds - 99%FCT with 1MB Flows

Load Balancer	**Homogeneous Link Speeds in DC Fabric**	**Heterogenous Link Speeds in DC Fabric**
ECMP	0.207755ms	0.267708ms
DIBS	0.207742ms	0.267708ms
DRILL	0.191276ms	0.182182ms
VERTIGO	0.203590ms	0.187558ms

mechanism. DIBS performs simple packet deflections; it also falls short when congestion occurs at the network core. This is due to the underlying issue with packet deflections, where the average path length taken by the packets can increase, based on the deflections, especially when it is decided in an congestion-agnostic manner, like in DIBS. With DRILL and Vertigo, we observe reduced FCTs when links between the aggregate and core switches are reduced to 50Gbps. In particular a reduction of, about 5% with DRILL, and 8% with Vertigo. This is a quite interesting observation in comparison to ECMP and DIBS, where we observed that the FCTs increased when the link speeds were reduced to 50Gbps. The main difference that occurs with having heterogeneous link speeds, is where the congestion "hotspot" is, which can be defined as where the localized region of persistent congestion happens. Ideally, an optimal load balancer should be able to make fair allocations and reduce FCTs, regardless of where congestion takes place. **Thus, this further motivates the need to have an adaptive load balancer that that can respond to congestion wherever it occurs and distributes load to achieve adequate trade-off between application flow QoS and efficient network utilization.**

5.0.5 Discussion

Overall, the results of our quantitative study that compares the performance of current datacenter load balancers while considering trends in modern datacenter workloads and architectures point to the following observations and insights:

- Their performance is quite sensitive to the kind of networks they operate in and workloads they are subject to, and tuning their parameters is often a non-trivial task.
- The coexistence of intra- and inter-DC workloads requires both end-to-end as well as in-network decision making capabilities.
- Having different congestion inference mechanisms as part of the network load balancer helps design a transport-agnostic model, making it easy to integrate with any transport congestion mechanism.

Chapter 6

Balancia

§ 6.1 Design Description

In Section 3.4.3, we highlight the importance of a dual control and decision making mechanism for network load balancers to best provision for DC-WAN mixed traffic, given their different requirements and operation timescales. Dual control refers to the ability to sense congestion and load balance both at the end-host and assisted by the network in order to react effectively at different timescales. This was quantitatively highlighted through the study carried out in Section 5. Based on our qualitative- and quantitative studies, we developed Balancia, a novel network load balancer able to dynamically adapt to both intra- and inter-datacenter traffic. Balancia addresses the diverse QoS requirements of intra- and inter-datacenter workloads by proactively sensing congestion and making decisions accordingly at the appropriate time scales. It dynamically switches between different control granularities to provide adequate performance for both intra- and inter-datacenter traffic. At its core, Balancia is a load balancing technique for geographically distributed datacenters that can dynamically

switch between end-host and in-network load balancing decision making. Additionally, it provides in-network support for congestion sensing and management.

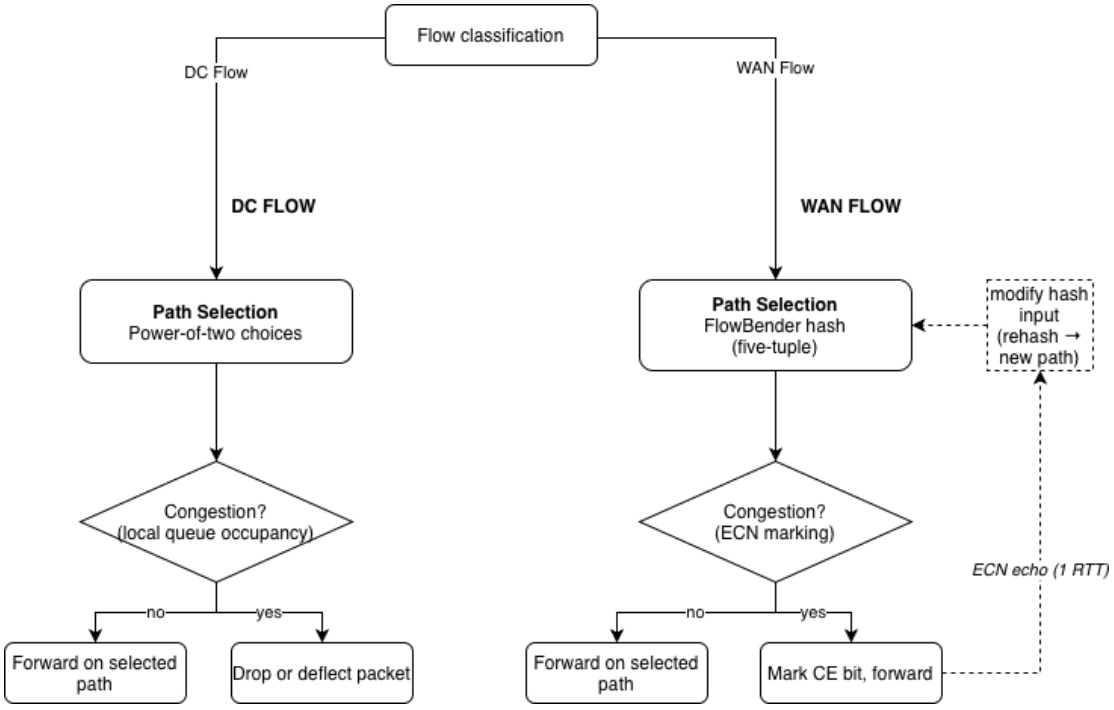

Figure 6.1: Balancia: Components and Workflow

Similar to other load balancing techniques, Balancia has three main components:

- Path Selection
- Congestion Sensing
- Congestion Management

Figure. 6.1 illustrates Balancia's decision flow - at the end-host, Balancia decides whether a flow is intra- or inter-datacenter bound. At a high level, Balancia's switches between Vertigo and FlowBender mechanisms for DC and WAN flows respectively.

Algorithm 1 Balancia's End Host Pseudo Code

```
1:  {Stage 1 - Path Selection by Flow Type}
2:  f ← flow of outgoing packet (p)
3:  if src(p), dst(p) intra-datacenter then
4:      class(f) ← DC
5:  else
6:      class(f) ← WAN
7:      if ack for f carries echoed CE mark then
8:          m_f ← m_f + 1 {marked acks};   t_f ← t_f + 1 {total acks}
9:          if m_f/t_f > T then
10:             modify TTL(f) {rehash: modifies hash input → new path}
11:         end if
12:     end if
13: end if
14: transmit p
15: {Shim Layer to Support Out-Of-Order Arrivals}
16: on arrival of p for flow f:
17: insert p into reorder buffer R_f keyed by seq
18: while R_f contains segment with seq = rcv_f and within duration of Reordering Timer
    do
19:     deliver it to transport;   rcv_f ← rcv_f + 1
20: end while
```

Path Selection : For intra-datacenter flows (DC FLOW), Balancia uses Vertigo's packet-level forwarding technique, inspired by the power of two choices [46] mechanism. Specifically, at each hop, the switch samples two candidate next-hops from the forwarding table and forwards the packet to the less occupied of the two, following the power-of-two-choices priniciple. This is outlined in lines 2-4 of Balancia's Switch pseudo code 2. Additionally, given that Vertigo was designed for DC flows, the sensitivity to Reordering Timer, which is part of the shim layer to support out-of-order arrivals does not play a considerable role. For inter-datacenter flows (WAN FLOW), Balancia maps the flow to a path determined by the hash computation of five fields, namely the source port, destination port, source IP address, destination IP address and

Algorithm 2 Balancia's Switch Pseudo Code

```
1:  {Stage 1 - Path selection and Forwarding Mechanism}
2:  if class(p) = DC then
3:      S ← sample 2 ports from Possible Ports P {power-of-two choices}
4:      {egressport}e ← arg min_{q∈S} OCCUPANCY(q)
5:  else
6:      h ← HASH(5-tuple, TTL(f), switchID)
7:      e ← P[ h mod |P| ] {per-flow path}
8:  end if
9:  {Stage 2 - Congestion sensing and response at e}
10: if OCCUPANCY(e) ≥ buffer_capacity then
11:     e' ← DEFLECT(p) {by contribution to persistent congestion}
12:     if e' = FULL then
13:         return drop(p)
14:     end if
15:     e ← e'
16: end if
17: {Stage 3 - ECN marking (WAN only)}
18: if class(p) = WAN and OCCUPANCY(e) ≥ ECN_threshold then
19:     set CE mark on p {echoed by receiver → host rehash}
20: end if
21: return forward(p, e)
```

the Time-To-Live (TTL) field, similar to FlowBender. Unlike Vertigo, this results in having the end-host make decisions on flow-level assignment. This is described in lines 6-7 of Balancia's Switch pseudo code 2.

In this manner, depending on the type of flow, i.e., intra- or inter-datacenter, Balancia dynamically switches between flow-level end-host or packet-level in-network decision making for path selection. Additionally, Balancia makes use of both end-to-end, as well as in-network congestion signals. Even inside the network, the load balancing switching regime still applies for DC and WAN flows.

Congestion Sensing and Management: We can see from Fig. 6.1, congestion is inferred through local switch queue occupancy for DC flows, and whenever the queue

exceeds set threshold, the packet is either dropped or deflected, depending on its contribution to persistent congestion in the network. This knowledge of persistent congestion is acquired from how much of the flow is remaining to be transmitted. Thus, by being able to infer and react in-network, Balancia is designed to be latency sensitive for DC flows. This is also noted under lines 9-16 of Balancia's Switch pseudo code 2. With WAN flows, as we can see from Fig. 6.1, congestion sensing takes place with the help of ECN markings. There is a set congestion threshold "T", which is determined empirically as described in FlowBender's original paper [2]. Whenever the ECN markings exceed this threshold, the input to the hash function is modified, thus re-assigning the flow to a new path, different from the current, congested path. This is described in lines 6-13 of Balancia's End-Host pseudo code 1.

These are particularly important decision and control mechanisms in order to react in-time for intra-datacenter flows, as well as wait for end-to-end signaling to react accurately in the case of inter-datacenter flows.

§ 6.2 Balancia Evaluation Results

This section describes the experiments carried out to evaluate Balancia under a variety of scenarios. We compare Balancia's performance against state-of-the-art load balancers under purely inter-datacenter workloads as well as under mixed intra- and inter-datacenter workloads. We also explore the impact of different datacenter topology designs, in particular switch buffer capacity.

6.2.1 Exploring Impact of Switch Buffer Capacities

In this set of experiments, we evaluate Balancia's performance with different switch buffer capacities. As noted earlier in Section 5.0.4, intra-datacenter workloads have much shorter RTTs (20μs)and are more latency sensitive when compared with inter-datacenter workloads which exhibit longer RTTs (2ms) and higher link utilization. This creates an interesting design challenge for switch buffer provisioning. We experimented with the following switch buffer configurations:

- Uniform switch buffer capacities of 1MB per port at the edge, aggregate and core switches as listed in Table 6.3. This value was chosen to reflect what has been used in works like [47],[18], [20] that have experimented with DC-WAN mixed traffic. However, this aren't the ideal switch buffer choices, due to the competing needs of DC and WAN traffic.

- Non-uniform or heterogeneous switch buffer capacities, where edge and aggregate switches are set to be shallow (250KB per port) and core switches are set to be deeper (7MB per port) as listed in Table 6.2. We explain the motivation behind using a heterogeneous switch buffer capacity fabric below.

Based on the takeaways from Section 5.0.4 and given that intra-datacenter workloads have tight latency requirements, having deep switch buffer - that are significantly larger than the bandwidth delay product - can cause buffer bloat and queue buildups that can further lead to transient congestion pockets. Thus, the aggregate and edge switch capacities have been fixed to be equal to the bandwidth delay product of the intra-datacenter network. Core switches, on the other hand, interconnect intra-datacenter fabrics, and their interconnects carry WAN traffic. Given the high throughput requirements of

inter-datacenter WAN traffic, having shallow buffers similar to the edge and aggregate switches will lead to quick buffer overflow due to the bandwidth mismatch. Additionally, with the increase of AI/ML training workloads =, works like [48], highlight their datacenter networks being designed with deep core switches.

For these experiments, we use the datacenter network fabric illustrated in Figure. 4.1 and the WAN workloads described in Section. 4.0.2. Section 4.0.3 describes the experimental parameters and their values are listed in Table 6.1. Results plotted in Figures 6.2 and 6.3 show 99% FCT as a function of the load across the bottleneck links. Examining 99% FCT helps establish how long the last 1% of the workload takes to complete.

Table 6.1: Balancia Evaluation: Experimental Parameters

Parameter	**Value**
Initial Congestion Window Size	10 packets
Initial RTO	$1s$
Minimum RTO	$100ms$
Vertigo and DRILL ReOrdering Timer	$0.002038s$
FlowBender's congestion threshold	5%

Table 6.2: Heterogeneous Switch Buffer Parameters

Parameter	**Value**
Edge Switch Per-Port Capacity	$250KB$
ECN Marking Threshold for Edge Switches	56 packets
Aggregate Switch Per-Port Capacity	$250KB$
ECN Marking Threshold for Aggregate Switches	56 packets
Core Switch Per-Port Capacity	$7MB$
ECN Marking Threshold for Core Switches	1555 packets

Results and Discussion: From Figure. 6.2 we see that ECMP, being congestion

Table 6.3: Uniform Switch Buffer Parameters

Parameter	**Value**
All switches Per-Port Capacity	$1MB$
ECN Marking Threshold at all switches	223 packets

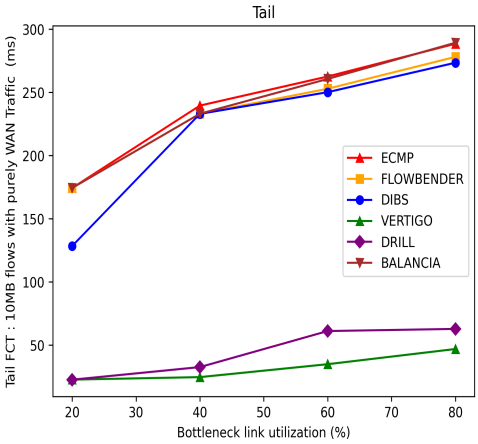

Figure 6.2: WAN Workloads - 99% FCT for 10MB flows with homogeneous switch capacities

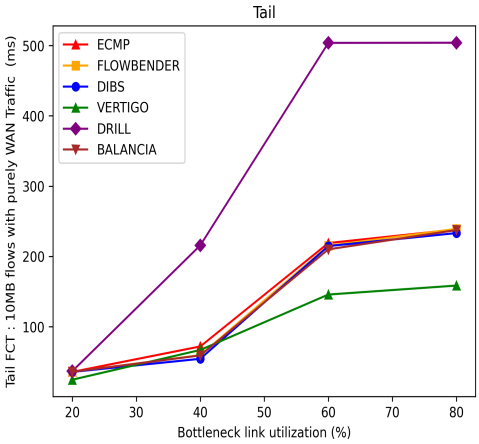

Figure 6.3: WAN Workloads - 99% FCT for 10MB flows with heterogeneous switch capacities

agnostic, causes collisions due to multiple flows being assigned the same path under high load. DIBS' approach of deflecting packets to any switch with available buffer, when the next hop buffer is full; causes packet reordering and significant performance degradation under high loads. However, Vertigo and DRILL still outperform both Balancia and FlowBender. However, it is important to note that this is with buffer capacities of 1MB at all switches, and so this does not scale well with bigger topologies. From Figure. 6.3, when using switch design as outlined in Table. 6.2, we notice that DRILL's 99% FCT is significantly worse compared to rest of the load balancers. This is due to its purely local, per-hop decisions. Which can attribute entire path's congestion

inference to just the local switch occupancy, which in the case of heterogeneous switch buffer designs, or any asymmetric network doesn't always hold true.

6.2.2 DC-WAN Mixed Workloads

In these DC-WAN mixed workload experiments, we study the performance of Balancia alongside state-of-the-art load balancers, using the network topology illustrated in Figure 4.1. In particular, we examine performance with the web search workloads described in Section. 4.0.2. The parameters used in these experiments are the heterogeneous switch capacity settings described Table. 6.2. Rest of the parameters are set as described in Table. 6.1. In order to better understand the results, we classify the results into different flowsize groups. The different flow groups can be defined as :

- < 10KB DC flows
- [10KB, 300KB) DC flows
- [300KB, 10MB) DC flows
- 10MB WAN flows

The x-axis represents the load on the bottleneck links, i.e., the supplied DC load. The y-axis represents mean FCT, and 99% FCT. We vary the utilization over the WAN links, to examine the effects on the DC and WAN traffic due to their co-existence. We have included the results with 80% WAN link utilization graphs for discussion. This is also illustrative of WAN traffic's high utilization, as described in [47],[18],[19] and [20]. Each experiment was run three times by varying the random seed for generating

different inter-arrival times and destination servers for the workloads. In particular, we have added the standard deviation bars across the multiple runs for these experiments.

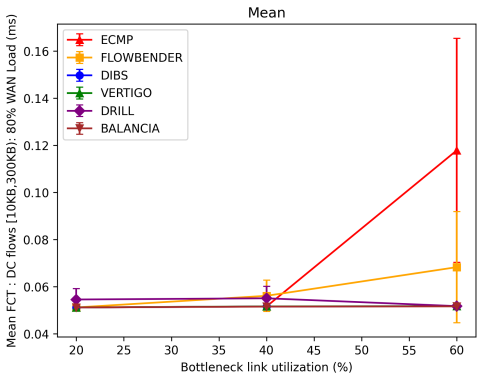

Figure 6.4: Mean FCT for [10KB,300KB) flows with 80% WAN Load

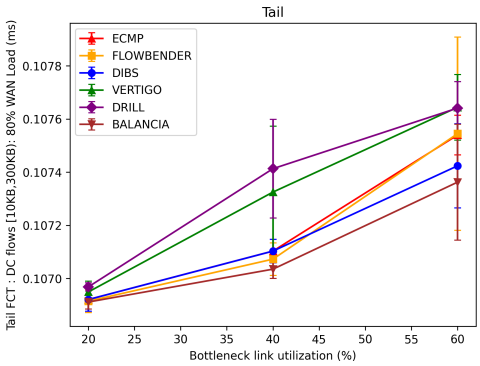

Figure 6.5: 99% FCT for [10KB,300KB) flows with 80% WAN Load

Results and Discussion: From Figure. 6.4 and Figure. 6.5, we observe that Balancia exhibits consistently low FCTs for DC flows under 80% high WAN utilization. DRILL being a local, per-hop based load balancing mechanism, it uses merely the local switch sizes to make the next hop decisions, thus being oblivious to the rest of the path conditions. This kind of decision making is quick to cause high tail latencies when we have non-uniform switch buffer capacities. From Figure. 6.6 and Figure. 6.7 for the

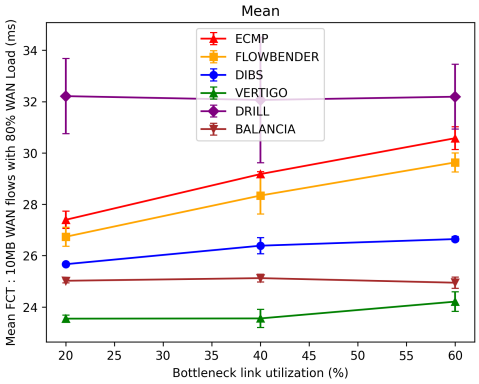

Figure 6.6: Mean FCT for 10MB flows with 80% WAN Load

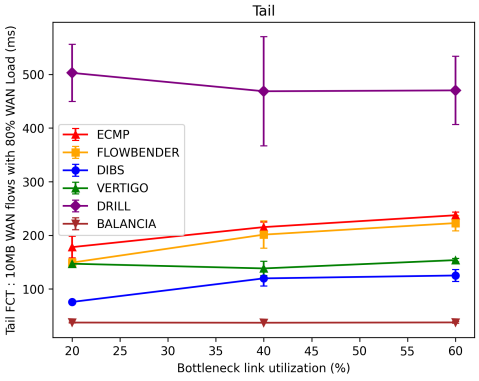

Figure 6.7: 99% FCT for 10MB flows with 80% WAN Load

Table 6.4: Relative 99% FCT reduction (%) over ECMP at 80% DC load and 80% WAN utilization

Load Balancer	**<10 KB**	**[300 KB, 10 MB)**	**10 MB**
Balancia	**0.78**	**99.10**	**85.05**
Vertigo	**0.78**	99.09	38.00
FlowBender	0.21	98.31	4.33
DRILL	0.65	99.04	-105.90
DIBS	0.01	99.00	36.55

WAN flows, while the mean FCT of Vertigo and Balancia is comparable, the 99% FCT shows how Balancia has a significantly lower 99% FCT than Vertigo, with about 78.6% reduction in 99% FCT. With Balancia's dual approach to load balancing, achieved by switching at both flow and packet level, as well as through end-host and in-network decisiong making, we are able to better manage the congestion buildup. By combining congestion signals at different granularities, Balancia is able to lower FCTs even under high load, for both DC and WAN traffic. Vertigo by itself does not perform optimally in particular due to its packet deflections mechanism, which does not add value in networks with heterogeneous switch buffer capacities.

We present in Table. 6.4, the relative reduction of 99% FCT (%) over ECMP, with 80% supplied DC load and 80% utilization of WAN links. We use ECMP as the baseline to show the relative performances, as it is one of the more earlier, and simpler approaches. From the table, we see Balancia's consistent low mean, and 99% FCT for all flowsizes. DRILL's 99% FCT depicts a negative value (Given that it's FCT is higher than ECMP), due it its instability at high loads. This as well is attributed to its local decision, which specifically with the co-existence of DC-WAN flows on heterogeneous switch capacities is quick to fall into instability.

Chapter 7

Balancia2.0

Balancia helped reduce the flow completion times of inter-datacenter flows significantly at higher loads. Works like [20] highlight how packet losses in these inter-datacenter links can further increase completion times, due to the recovery process over these long haul links, with the increased retransmission times. This raises the important question of whether we can proactively avoid packet losses for the intra-, and inter-datacenter interactions. How can effective load balancing be achieved with the help of proactive congestion signaling mechanisms.

With these discussions, we outline the following design goals for Balancia2.0 :

- Exploring sub-RTT in-network rerouting mechanisms to avoid packet loss for inter-datacenter flows
- Decoupling congestion control rate reduction signaling from load balancing reroute signaling
- Impact of in-network responses for inter-datacenter flows

As part of addressing these design goals, we propose Balancia2.0 - which makes use of

phantom queues [20, 49] and an "In-Fabric" FlowBender for WAN flows.

§ 7.1 Design Description

7.1.1 Phantom Queues

In order to help with being proactive and warn of queueing even before queueing occurs, we incorporate phantom queues [20, 49]. Phantom queues was originally introduced in the "Less is More" [49] work, where the idea stems from trading a little bandwidth for ultra-low latency. Uno [20], a very recent work examines intra-, and inter-datacenter communications uses phantom queues as part of its congestion control component. Phantom queues are a signaling mechanism, more particularly a shadow counter that triggers action before onset of actual congestion. While using phantom queues, it is important to understand that no actual packets are buffered by the phantom queue. Packets continue through the real egress port at full line rate.

In Balancia2.0, the motivation behind using phantom queues stems from the idea of HULL's, where we try to see if trading some bandwidth can help with attaining ultra-low latencies. In particular, we explore how phantom queues can be used to achieve "in-network" flowbender for WAN flows. While DCTCP's congestion window adjustment still comes from ECN markings and the ECN threshold, we modify FlowBender to do rerouting based on phantom queues and phantom queue threshold. Additionally, while prior works [20, 49] use phantom queues for better and efficient congestion control functionality, we explore the impacts and usage of phantom queues for rerouting decisions.

7.1.2 Balancia2.0 Design

In this section we describe the design of Balancia2.0. Similar to Balancia, Balancia2.0 also switches between Vertigo for DC flows and FlowBender for WAN flows. However, in order to further optimize for WAN flows, we propose the "In-Network" Flowbender for WAN flows. Essentially, instead of rerouting flows based on ECN markings, we use the phantom queue threshold to determine rerouting decisions. In order to decide the rerouting decisions we deploy "comparative rerouting". Comparative rerouting works from the point phantom buffer occupancy exceeds the threshold. When that happens, all packets of that flow from the point of congestion are rerouted to the next hop candidate port that has the least phantom queue occupancy.

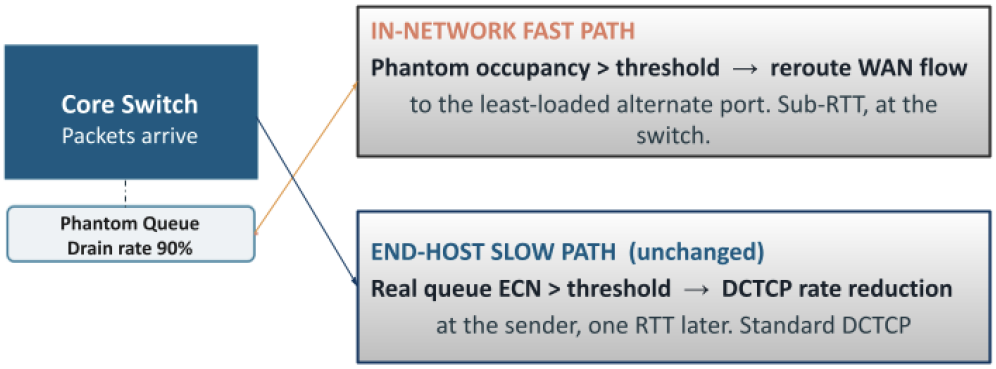

Figure 7.1: Balancia2.0: In-Network FlowBender for WAN Flows

7.1.3 Balancia2.0 Parameters

The main parameters that are critical to Balancia2.0's design are specifically related to the phantom queue's design. We have the "drain rate" and "phantom queue threshold". The drain rate is the determining factor for how much bandwidth we are comfortable

making a tradeoff with. It also describes How aggressively the virtual queue anticipates congestion relative to the real one. Works that discuss phantom queues [20, 49] recommend setting this rate to 90%. Next, we have the phantom queue threshold, which determines how much sustained oversubscription must accumulate before you act. Once the phantom queue is building, the threshold is the depth it must reach to trigger the rerouting mechanism. In the evaluations, we discuss how we set these parameters and our recommendations moving forward.

§ 7.2 Balancia2.0 Evaluation

In order to evaluate the usefulness and effectiveness of Balancia2.0, we compare Balancia and Balancia2.0 under DC-WAN mixed workload experiments, using the network topology illustrated in Figure 4.1. In particular, we examine performance with the web search workloads described in Section. 4.0.2. The parameters used in these experiments are the heterogeneous switch capacity settings described Table. 6.2. Rest of the parameters are set as described in Table. 6.1. In order to better understand the results, we classify the results into different flowsize groups. The different flow groups can be defined as :

- $<$ 10KB DC flows
- [10KB, 300KB) DC flows
- [300KB, 10MB) DC flows
- 10MB WAN flows

The x-axis represents the load on the bottleneck links, i.e., the supplied DC load. The y-axis represents median FCT, mean FCT and 99% FCT in different scenarios. We vary the utilization over the WAN links, to examine the effects on the DC and WAN traffic due to their co-existence. We have included the results with 80% WAN link utilization graphs for discussion. This is also illustrative of WAN traffic's high utilization, as described in [47],[18],[19] and [20]. Each experiment was run three times by varying the random seed for generating different inter-arrival times and destination servers for the workloads. In particular, we have added the standard deviation bars across the multiple runs for these experiments. We set the drain rate to be 90%, reflecting recommendations from [20, 49]. For the phantom queue threshold, based on our experimental tests based on heuristics, set the value to be 3MB. The key point to setting this value is being able to detect transient congestion and enabling sub-RTT reaction. Under the results discussion and looking forward sections, we discuss how this can further be designed adaptively.

7.2.1 Results Discussion

The main objective with these experiments were to ensure that we observe further reduction in flow completion times, in comparison to Balancia - specifically for WAN flows. With DC flows, as seen in figures 7.2- 7.4, the median FCT across different loads remains pretty similar for both Balancia and Balancia2.0; which is still a good takeaway that overall the FCT of these DC flows doesnt get too skewed due to the effect of phantom queues at the core switches. From fig 7.5 and fig 7.6 we see that Balancia2.0 helps reduce the 99% FCT in comparison to Balancia across different supplied DC loads.

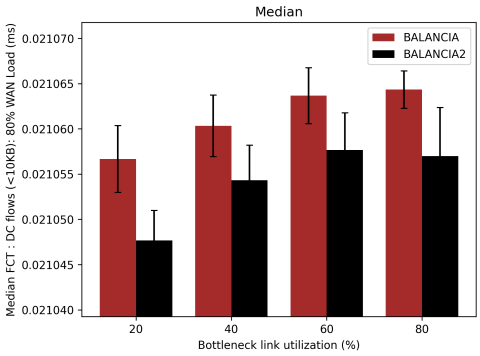

Figure 7.2: Median FCT for <10KB DC flows with 80% WAN Load

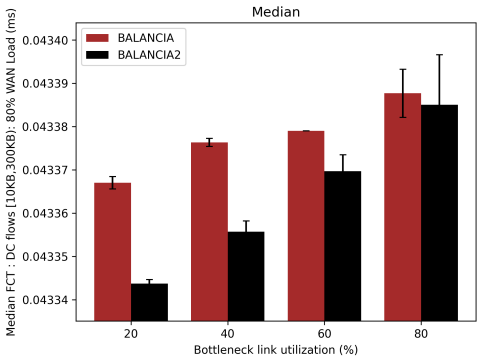

Figure 7.3: Median FCT for [10KB,300KB) DC flows with 80% WAN Load

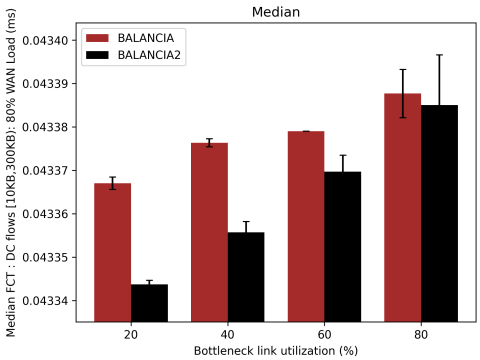

Figure 7.4: Median FCT for [300KB,10MB) DC flows with 80% WAN Load

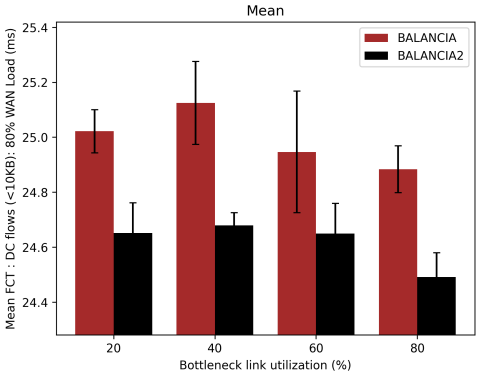

Figure 7.5: Mean FCT for WAN flows with 80% WAN Load

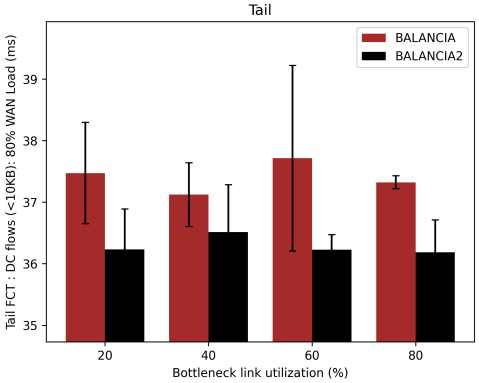

Figure 7.6: 99% FCT for WAN flows with 80% WAN Load

7.2.2 Balancia2.0 - Looking Forward

What we have so far with Balancia2.0, is more preliminary results examining the usefulness of phantom queues for the purposes of rerouting decisions, rather than congestion control. While the initial results are quite promising, this opens up direction for future interesting research exploring the effects of different drain rates, phantom queue thresholds and their interplay. For example, it will be interesting to explore setting the phantom queue threshold to be adaptive to the level of congestion in the network. Can we predict whether the congestion event was a microburst event or sustained congestion? Since, phantom queue based sub-RTT reaction is only justified and deems workable if the congestion is transient. For sustained congestion, End-to-end FlowBender works just fine. With these takeaways and future possibilities, the next chapter concludes the thesis summarizing all the work so far.

Chapter 8

Conclusion

Current datacenter load balancers have been designed assuming intra-datacenter workloads, i.e., workloads that are handled within a centrally located datacenter. As datacenter applications grow more complex and diverse and datacenters scale and become more geographically distributed, load balancers must evolve to ensure applications receive their required QoS, while utilizing datacenter resources adequately. We propose a taxonomy (Table 3.1) highlighting the interplay between different transport-layer congestion control mechanisms and network-level load balancing decisions. Given the limited research on load balancer behavior under intra- and inter-datacenter mixed workloads, we develop experimental models and methodology to represent these workloads and conduct systematic datacenter load balancer performance evaluation, including recommendations for load balancer parameter settings. Guided by the takeaways from our qualitative and quantitative studies, we present Balancia, a lightweight, transport-agnostic network load balancer that dynamically switches between per-flow and per-packet control decisions to provide optimal performance for mixed intra- and inter-datacenter traffic. Further, Balancia has both end-host, and in-network decision making capabilities to

support the different timescales of intra- and inter-datacenter workloads. Through detailed quantitative evaluations, we demonstrate Balancia's superior performance under different workload scenarios when compared to state-of-the-art load balancers. More specifically, we show that Balancia achieves close to 80% reduction in 99% tail flow completion times.

§ 8.1 Future Directions

As part of ongoing work and future directions is the continued development and testing of Balancia2.0 . In particular, the drain rate and phantom queue thresholds are very important parameters for Balancia2.0, and their interplay and sensitivity analysis is important to be explored. As part of our future work, we plan to model AI/ML training workloads and evaluate Balancia's performance as well as Balancia2.0's for AI/ML training traffic across datacenters. We are also in the process of open-sourcing our distributed datacenter experimental setup, including our workload modeling and make it available to the community.

Bibliography

[1] Christian Hopps et al., *Analysis of an equal-cost multi-path algorithm*, RFC 2992, November, 2000.

[2] Abdul Kabbani, Balajee Vamanan, Jahangir Hasan, and Fabien Duchene, *FlowBender: Flow-level Adaptive Routing for Improved Latency and Throughput in Datacenter Networks*, Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies, December 2014, pp. 149–160 (en).

[3] Mohammad Alizadeh, Tom Edsall, Sarang Dharmapurikar, Ramanan Vaidyanathan, Kevin Chu, Andy Fingerhut, Vinh The Lam, Francis Matus, Rong Pan, Navindra Yadav, et al., *Conga: Distributed congestion-aware load balancing for datacenters*, Proceedings of the 2014 acm conference on sigcomm, 2014, pp. 503–514.

[4] Hong Zhang, Junxue Zhang, Wei Bai, Kai Chen, and Mosharaf Chowdhury, *Resilient datacenter load balancing in the wild*, Proceedings of the conference of the acm special interest group on data communication, 2017, pp. 253–266.

[5] Soudeh Ghorbani, Zibin Yang, P. Brighten Godfrey, Yashar Ganjali, and Amin Firoozshahian, *DRILL: Micro Load Balancing for Low-latency Data Center Networks*, Proceedings of the Conference of the ACM Special Interest Group on Data Communication, August 2017, pp. 225–238.

[6] Kyriakos Zarifis, Rui Miao, Matt Calder, Ethan Katz-Bassett, Minlan Yu, and Jitendra Padhye, *DIBS: just-in-time congestion mitigation for data centers*, Proceedings of the Ninth European Conference on Computer Systems - EuroSys '14, 2014, pp. 1–14 (en).

[7] Sepehr Abdous, Erfan Sharafzadeh, and Soudeh Ghorbani, *Burst-tolerant datacenter networks with Vertigo*, Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies, December 2021, pp. 1–15.

[8] _____, *Practical Packet Deflection in Datacenters*, Proceedings of the ACM on Networking **1** (November 2023), no. CoNEXT3, 1–25.

[9] Tommaso Bonato, Abdul Kabbani, Ahmad Ghalayini, Michael Papamichael, Mohammad Dohadwala, Lukas Gianinazzi, Mikhail Khalilov, Elias Achermann, Daniele De Sensi, and Torsten Hoefler, *REPS: Recycled Entropy Packet Spraying for Adaptive Load Balancing and Failure Mitigation*, 2025.

[10] Vamsi Addanki, Prateesh Goyal, Ilias Marinos, and Stefan Schmid, *Ethereal: Divide and Conquer Network Load Balancing in Large-Scale Distributed Training*, 2025.

[11] D. Alves, K. Obraczka, and A. Kabbani, *Gdsim: Benchmarking geo-distributed data center schedulers*, 2021 ieee 10th international conference on cloud networking (cloudnet), November 2021, pp. 148–156.

[12] Anchengcheng Zhou, Carter Costic, Hongyu Hè, Ahmad Ghalayini, Abdul Kabbani, and Maria Apostolaki, *Mitigating Inter-datacenter Incast with a Proxy: The shortest path is not necessarily the fastest*, Proceedings of the 24th ACM Workshop on Hot Topics in Networks, November 2025, pp. 344–353.

[13] *Distributed data center architecture: Ensuring scalability*. Accessed: Mar. 10, 2026.

[14] Google Cloud, *Regional, dual-region, and multi-region configurations*. Accessed: Mar. 10, 2026.

[15] NetApp, *StorageGRID architecture and network topology*. Accessed: Mar. 10, 2026.

[16] Adi Gangidi, *Scaling RoCE networks for AI training*, 2023. [Online video]. Available: https://atscaleconference.com/videos/scaling-roce-networks-for-ai-training/. [Accessed: Mar. 10, 2026].

[17] Bloomberg, *Inside the first stargate AI data center*, 2025. [Online]. Available: https://www.bloomberg.com/news/features/2025-05-20/inside-stargate-ai-data-center-from-openai-and-softbank. [Accessed: Mar. 10, 2026].

[18] Ahmed Saeed, Varun Gupta, Prateesh Goyal, Milad Sharif, Rong Pan, Mostafa Ammar, Ellen Zegura, Keon Jang, Mohammad Alizadeh, Abdul Kabbani, and Amin Vahdat, *Annulus: A dual congestion control loop for datacenter and wan traffic aggregates*, Proceedings of the annual conference of the acm special interest group on data communication on the applications, technologies, architectures, and protocols for computer communication, 2020, pp. 735–749.

[19] G.Zeng, W.Bai, G.Chen, K.Chen, D.Han, Yibo Y.Zhu, and L.Cui, *Congestion control for cross-datacenter networks*, IEEE/ACM Trans. Networking **30** (2022Oct.), no. 5, 2074–2089.

[20] Tommaso Bonato, Sepehr Abdous, Abdul Kabbani, Ahmad Ghalayini, Nadeen Gebara, Terry Lam, Anup Agarwal, Tiancheng Chen, Zhuolong Yu, Konstantin Taranov, Mahmoud Elhaddad, Daniele De Sensi, Soudeh Ghorbani, and Torsten Hoefler, *Uno: A One-Stop Solution for Inter- and Intra-Data Center Congestion Control and Reliable Connectivity*, Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, November 2025, pp. 1195–1210.

[21] Harikrishna S. Kuttivelil, Shesha Sreenivasamurthy, Lakshmi Krishnaswamy, Nayan Bhatia, and Katia Obraczka, *Network simulation bridge: Bridging applications to network simulators*, Proceedings of the 19th acm international symposium on qos and security for wireless and mobile networks, 2023, pp. 39–46.

[22] Mubashir Adnan Qureshi, Yuchung Cheng, Qianwen Yin, Qiaobin Fu, Gautam Kumar, Masoud Moshref, Junhua Yan, Van Jacobson, David Wetherall, and Abdul Kabbani, *PLB: congestion signals are simple and effective for network load balancing*, Proceedings of the ACM SIGCOMM 2022 Conference, August 2022, pp. 207–218.

[23] Keqiang He, Eric Rozner, Kanak Agarwal, Wes Felter, John Carter, and Aditya Akella, *Presto: Edge-based load balancing for fast datacenter networks*, Proceedings of the 2015 acm conference on special interest group on data communication, 2015, pp. 465–478.

[24] Kun Qian, Yongqing Xi, Jiamin Cao, Jiaqi Gao, Yichi Xu, Yu Guan, Binzhang Fu, Xuemei Shi, Fangbo Zhu, Rui Miao, Chao Wang, Peng Wang, Pengcheng Zhang, Xianlong Zeng, Eddie Ruan, Zhiping Yao, Ennan Zhai, and Dennis Cai, *Alibaba HPN: A Data Center Network for Large Language Model Training*, Proceedings of the ACM SIGCOMM 2024 Conference, August 2024, pp. 691–706.

[25] Naga Katta, Mukesh Hira, Changhoon Kim, Anirudh Sivaraman, and Jennifer Rexford, *Hula: Scalable load balancing using programmable data planes*, 2016.

[26] Naga Katta, Mukesh Hira, Aditi Ghag, Changhoon Kim, Isaac Keslassy, and Jennifer Rexford, *Clove: How i learned to stop worrying about the core and love the edge*, Proceedings of the 15th acm workshop on hot topics in networks, 2016, pp. 155–161.

[27] Erico Vanini, Rong Pan, Mohammad Alizadeh, Parvin Taheri, and Tom Edsall, *Let it Flow: Resilient Asymmetric Load Balancing with Flowlet Switching*.

[28] Srikanth Kandula, Dina Katabi, Shantanu Sinha, and Arthur Berger, *Dynamic load balancing without packet reordering*, SIGCOMM Comput. Commun. Rev. **37** (March 2007), no. 2, 51–62.

[29] Sally Floyd, K. K. Ramakrishnan, and David L. Black, *The Addition of Explicit Congestion Notification (ECN) to IP*, Technical Report RFC 3168, Internet Engineering Task Force, 2001.

[30] Jiao Zhang, F. Richard Yu, Shuo Wang, Tao Huang, Zengyi Liu, and Yunjie Liu, *Load Balancing in Data Center Networks: A Survey*, IEEE Communications Surveys & Tutorials **20** (2018), no. 3, 2324–2352.

[31] David Wetherall, Abdul Kabbani, Van Jacobson, Jim Winget, Yuchung Cheng, Charles B. Morrey Iii, Uma Moravapalle, Phillipa Gill, Steven Knight, and Amin Vahdat, *Improving Network Availability with Protective ReRoute*, Proceedings of the ACM SIGCOMM 2023 Conference, September 2023, pp. 684–695.

[32] Hong Xu and Baochun Li, *TinyFlow: Breaking elephants down into mice in data center networks*, 2014 IEEE 20th International Workshop on Local & Metropolitan Area Networks (LANMAN), May 2014, pp. 1–6.

[33] Yilong Geng, Vimalkumar Jeyakumar, Abdul Kabbani, and Mohammad Alizadeh, *Juggler: a practical reordering resilient network stack for datacenters*, Proceedings of the eleventh european conference on computer systems, 2016.

[34] Mallik Mahalingam, Dinesh Dutt, Kenneth Duda, Puneet Agarwal, Larry Kreeger, T. Sridhar, Mike Bursell, and Chris Wright, *Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks*, Technical Report RFC 7348, Internet Engineering Task Force, 2014.

[35] Tommaso Bonato, Abdul Kabbani, Daniele De Sensi, Rong Pan, Yanfang Le, Costin Raiciu, Mark Handley, Timo Schneider, Nils Blach, Ahmad Ghalayini, Daniel Alves, Michael Papamichael, Adrian Caulfield, and Torsten Hoefler, *SMaRTT-REPS: Sender-based Marked Rapidly-adapting Trimmed & Timed Transport with Recycled Entropies*, 2024.

[36] Yuxuan Li, Zhenghang Ren, Wenxue Li, Xiangzhou Liu, and Kai Chen, *Congestion Control for AI Workloads with Message-Level Signaling*, Proceedings of the 9th Asia-Pacific Workshop on Networking, August 2025, pp. 59–65.

[37] Ultra Ethernet Consortium, *Ultra Ethernet Specification, Version 1.0*, Ultra Ethernet Consortium, 2025. Joint Development Foundation Project, Linux Foundation.

[38] *Omnet++ discrete event simulator*.

[39] *Inet framework*.

[40] Dmitry Duplyakin, Robert Ricci, Aleksander Maricq, Gary Wong, Jonathon Duerig, Eric Eide, Leigh Stoller, Mike Hibler, David Johnson, Kirk Webb, Aditya Akella, Kuangching Wang, Glenn Ricart, Larry Landweber, Chip Elliott, Michael Zink, Emmanuel Cecchet, Snigdhaswin Kar, and Prabodh Mishra, *The design and operation of cloudlab*, Proceedings of the 2019 USENIX Conference on Usenix Annual Technical Conference, July 2019, pp. 1–14.

[41] Mohammad Al-Fares, Sivasankar Radhakrishnan, Barath Raghavan, Nelson Huang, Amin Vahdat, et al., *Hedera: dynamic flow scheduling for data center networks.*, Nsdi, 2010, pp. 89–92.

[42] Albert Greenberg, James R Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A Maltz, Parveen Patel, and Sudipta Sengupta, *Vl2: A scalable and flexible data center network*, Proceedings of the acm sigcomm 2009 conference on data communication, 2009, pp. 51–62.

[43] Mohammad Alizadeh, Albert Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan, *Data center TCP (dctcp)*, Proceedings of the acm sigcomm 2010 conference, August 2010, pp. 63–74.

[44] Sangtae Ha, Injong Rhee, and Lisong Xu, *Cubic: a new tcp-friendly high-speed tcp variant* **42** (July 2008), no. 5, 64–74.

[45] Nutanix, Inc., *Physical networking best practices (bp-2050)*, 2023. Version 2.6, March 2023.

[46] Michael Mitzenmacher, Andréa W. Richa, and Ramesh Sitaraman, *The Power of Two Random Choices: A Survey of Techniques and Results*, Handbook of Randomized Computing, 2001, pp. 255–312.

[47] Lakshmi Krishnaswamy, Katia Obraczka, and Abdul Kabbani, *What's Next for Datacenter Load Balancers - A Comparative Performance Study of the State-of-the-Art*, 2024 IEEE 13th International Conference on Cloud Networking (CloudNet), November 2024, pp. 1–9.

[48] The Llama 3 Herd of Models (November 2024)

[49] Mohammad Alizadeh, Abdul Kabbani, Tom Edsall, Balaji Prabhakar, Amin Vahdat, and Masato Yasuda, *Less is more: trading a little bandwidth for ultra-low latency in the data center*, Proceedings of the 9th usenix conference on networked systems design and implementation, 2012, pp. 19.