Optimizing Traffic in Public-Facing Data Centers Amid Internet Protocols

Sen Lin*, Jianfeng Wang†, Aleksandar Kuzmanovic*
*Northwestern University, USA †Oracle America, Inc.
sen.lin@u.northwestern.edu, pkueewjf@gmail.com, akuzma@northwestern.edu

***Abstract*—Rapid development has been witnessed in optimizing the performance of data centers over the past decade. However, such advances thriving in private data centers are rarely deployed in public-facing data centers. A major challenge is synchronizing optimization signals—such as flow sizes, server assignments, and load information—with the traffic they are intended to optimize, especially across networks controlled by different entities. In this paper, we propose CLOUDCOOKIE, a versatile signal carrier within Internet protocols that ensures bidirectional signal presence without client-side cooperation. To exemplify CLOUDCOOKIE's benefits on public-facing data center traffic, we design a set of easy-to-deploy data center infrastructures, including load balancers and switches, to leverage application layer awareness and enable efficient flow packet scheduling and load balancing. Our evaluation shows that these advances synergistically optimize the 99th percentile of flow completion time by up to 20x for the majority of flows.**

I. INTRODUCTION

The performance of data-center networks (DCN) continues to thrive due to ever-improving techniques and systems deployed at different DCN layers [1]–[5]. Contrary to the public Internet where a consensus among multiple disjoint entities needs to be reached before any deployment is made, data centers require no such consensus. Indeed, the data center *centralization* is one of the key reasons driving significant progress in this space. However, these advances, which have proven successful within *private* data centers, where both endpoints and the intermediate network devices are controlled by the same data center authority, are rarely deployed in *public-facing* data centers that serve general web users and handle most of today's Internet traffic. [6].

A public-facing data center authority no longer has central control because the network between the client and server can be segmented by multiple entities (Figure 1). For example, a client's request must first traverse the local ISP. Next, from the local ISP to the destination data center, the traffic could (1) be shipped on other portions of the public Internet, *e.g.,* ISPs, IXPs, or third-party DCNs (P_1); (2) directly enter the controlled DCNs (P_5); or (3) bounce back and forth between the public Internet and controlled DCNs (P_2, P_3, and P_4), when inter-DC traffic needs to be transited via third parties. The uncontrolled off-net partitions prevent data center advances from being deployed in public-facing data centers.

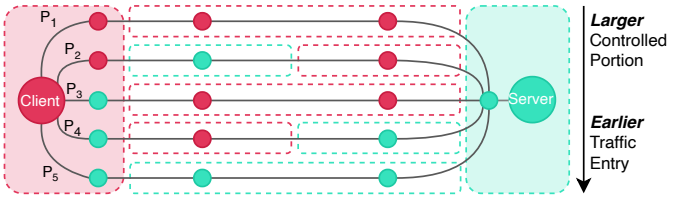


Fig. 1. Different paths of routing public-facing traffic. Network devices (circles) and autonomous systems (ASes, dashed rectangles) are color-coded: green and red indicate whether the device/AS is controlled by the same authority as the destination DCN. Source and destination ASes are highlighted.

As a consequence, synchronizing public-facing traffic with optimization signals for achieving advanced traffic handling becomes challenging. First, the compatibility requirements of Internet protocols prevent the deployment of custom headers, *e.g.,* packet encapsulation. Second, due to the inevitable traversal through uncontrolled devices, common signal fields in Internet protocols, *e.g.,* Traffic Class, are low-res and volatile. Third, without the cooperation from the client-side, it is difficult to ensure the signal presence in the traffic from end users. Last, connections initiated by end users, unlike internal services, introduce increased uncertainty, *e.g.,* varied sizes, bursty arrivals, and dynamic routing. Out-of-bound (OOB) channels hence fail to enable timely and spatially aligned signal communication. As in the example of Figure 1, it is uncertain which path will be chosen. Given these restrictions, we ask: *if it is possible, and how, to utilize private-DCN advances in public-facing data center settings?*

This question is becoming even more relevant as DCN authorities have significantly moved forward their boundaries and enlarged the network control over the past decades. In particular, tech giants construct new data centers closer to client endpoints (P_2 and P_5) and deploy off-net infrastructures in clients' eyeball networks (P_3, P_4, and P_5) as shown in Figure 1 [6]–[9]. The ideal scenario is P_5, despite the client endpoint is still out of control. There are two incentives behind this *flattering* Internet trend. On one hand, directing the traffic to enter the DCN earlier can significantly reduce operational costs, including peering expenses. On the other hand, an increased control over the traffic by the same DCN authority enables more optimization on both data and control planes. With a growing overlap with inter-DC traffic, public-

†Work done while the author was at University of Southern California. Any opinions, findings, conclusions, or recommendations expressed herein are those of the authors and do not necessarily reflect the views of Oracle America, Inc.

979-8-3503-5171-2/24/$31.00 ©2024 IEEE

facing traffic from data centers of these giants now contributes more than half of the global Internet traffic [6].

In this paper, we aim to optimize the overall performance of public-facing traffic, and propose CLOUDCOOKIE as a lever for harnessing the expanded control within the data center. CLOUDCOOKIE serves as an optimization signal carrier embedded in Internet protocols, offering several properties that are key for advanced public-facing DCN traffic handling: (1) **versatility**, as its spacious coding space allows assorted expressive signals for different optimization techniques; (2) **deployability**, as its bidirectional presence requires no client-side modification and it's hot-swappable to existing Internet ecosystem; (3) **timeliness**, with signals piggybacked on regular packets and synchronized in time and space with their corresponding reaction points; (4) **robustness**, since loss occurs only with dropped packets, with transport layer retransmissions ensuring reliable delivery; and (5) **security**, ensuring signal integrity and resistance to tampering through cryptographic tools.

CLOUDCOOKIE is *decoupled* from specific optimization techniques. To exemplify its ability to port private data center advances into public-facing traffic, we let CLOUDCOOKIE carry a combination of signals that enable Shortest Remaining Processing Time (SRPT) flow packet scheduling, for bandwidth allocation, and *predictive* load-aware load balancing, for traffic distribution. To produce the required signals, i.e., RPT and loads, we design a CLOUDCOOKIE layer-7 load balancer ($CCLB_7$) that leverages neglected information in applications and systems. To consume the signals, we design a stateless CLOUDCOOKIE layer-4 load balancer ($CCLB_4$), that makes foresight decisions; and adapt an state-of-the-art in-switch flow scheduler [10]. In §V, we further analyze the synergistic optimization of both techniques.

We summarize the contributions of this paper as follows:

- We propose CLOUDCOOKIE, a versatile, deployable, timely, robust, and secure carrier for optimization signals. CLOUDCOOKIE enables applying state-of-the-art private data center optimizations for public-facing traffic.
- CLOUDCOOKIE's design is hot-swappable: it utilizes Internet protocols and respects the existing Internet ecosystem; it also requires zero client-side modification.
- To serve CLOUDCOOKIE, we propose an easy-to-deploy infrastructure suite consisting of an optimization signal producer–$CCLB_7$, and two signal consumers–$CCLB_4$ and *scheduler-on* switch.
- For the first time, we implement SRPT flow scheduling on real-world web traffic, which is characterized by data flows triggered by bursty end-user requests.
- In emerging QUIC/IPv6 networks, CLOUDCOOKIE utilizes a 64-bit coding space, while, in classic TCP networks, the space is limited to 16 bits. In both contexts, we explore various optimization techniques and their synergistic effects, reducing the 99th percentile flow completion time (FCT) of the majority flows by up to 20x.

In the rest of this paper, we introduce the challenges and technical background of handling public-facing traffic in §II.

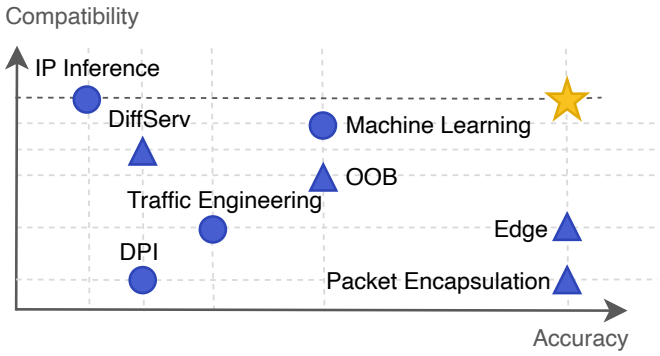


Fig. 2. Comparison of state-of-the-art methods that exchange (▲) and gain (●) . The upper-right corner is preferred, which is the goal of CLOUDCOOKIE (★).

Then, in §III, we propose our design of CLOUDCOOKIE. In §IV, we introduce the design of hot-swappable infrastructure and how they react to CLOUDCOOKIE. After that, we show our evaluation results and insights in §V. Discussion and conclusion are presented in §VI and §VII respectively.

II. BACKGROUND AND RELATED WORK

The split central control of public-facing traffic introduces challenges in exchanging optimization signals among multiple DCN-controlled network devices (§II-A) and obtaining such traffic knowledge (§II-B). These challenges highlight certain ignorance that CLOUDCOOKIE aims to tackle. Additionally, to clarify our design and approach, we present a multi-tier-LB architecture abstraction that enables DCN's adaptation to CLOUDCOOKIE (§II-C) and state-of-the-art flow packet scheduling that CLOUDCOOKIE targets to activate (§II-D).

A. Challenges in Signal Exchanges

Figure 2 compares signal exchange methods in two dimensions: their compatibility with the current Internet ecosystem and their ability to deliver expressive messages accurately and timely. The most straightforward method is to exchange optimization signals by utilizing existing fields in IP headers, which are called differentiated service (DiffServ), *i.e.,* `DSCP` [1] in IPv4 and `Traffic Class` in IPv6 to encode required information. For example, Mahout [13] uses `DSCP` to encode scheduling signals in the packet. This method boasts high compatibility since it doesn't require any specific modifications, yet there are still two issues to consider: First, the compatibility is compromised because of its volatility. DiffServ fields are commonly reset or remarked when crossing controlled network boundaries due to compatibility, policy, and security concerns [14]. Second, DiffServ's low resolution, *i.e.,* 6-bit long excluding 2 ECN bits, forbids deploying fine-grained optimizations, such as scheduling packets according to flow sizes or RPT.

[1] `DSCP` stands for Differentiated Services Code Point, which was originally defined as `ToS`, Type of Service [11], [12].

On the other hand, the OOB communication channel can serve as an effective method for exchanging signals between network devices in private data centers. It offers ample coding space, thereby resolving issues related to resolution. OOB channels can be established through dedicated messages, *e.g.,* pHost [15] and Homa [16], or dedicated connections, *e.g.,* the software-defined networking (SDN) family. Nevertheless, deploying OOB methods in public-facing data centers faces several challenges. Synchronization issues are at the forefront. First, the inevitable delays, between optimization signals and their serving traffic, damage the accuracy of signals. The damage increases as the reaction points become close to the client, but distant from the data center where the signal is generated. We observe recent efforts that optimizes public-facing traffic with OOB methods [8], [17], [18]. Although these proposals offer improvements in directing traffic across multiple paths, we maintain that there remains considerable potential for optimization within the data plane, particularly for bursty and ephemeral public-facing traffic. Second, the spatial unsynchronization of OOB signals with their corresponding reaction points raises additional concerns regarding server compatibility. Transmission paths can differ from one connection to another or even from packet to packet, which complicates the issue. Consequently, it is challenging to ascertain which network devices are the recipients of a specific signal. Therefore, it becomes infeasible to determine which network devices ought to be the subscribers for a certain signal. Third, the implementation of OOB methods entails increased overheads on resources and bandwidth. The deployment of these mechanisms typically requires centralized controllers, and the delivery of OOB signals generates extra traffic that competes with regular data flows. While such overheads may be acceptable in private data centers with over-provisioned resources and bandwidth, this assumption does not hold in public-facing data centers.

Packet encapsulation resolves synchronization issues by aligning the optimization signals in bound with the regular traffic that they serve. Generic tunneling, *e.g.,* GRE [19] and GENEVE [20], and custom encapsulation, *e.g.,* VL2 [3] and PDQ [21] are popular methods in this category. However, a grave concern is that packet encapsulation is often an all-or-nothing choice. Deploying this technique for a service requires changing all servers, clients, and optional intermediate network devices depending on the task. In the case of load balancing, an unsupported load balancer would reject all encapsulated packets, while a supported load balancer would discard any packet that is not encapsulated. Additionally, due to extra headers, the inflated packet size may exceed the MTU size allowed on the public Internet. Based on these two reasons, we rate the compatibility of this method as the lowest level. Moreover, computational overheads and delays are multiplied by the need for encapsulation and decapsulation to be performed on each intermediate device.

So far, *all* above methods fall short when they come to preserving signals in packets from unmodified normal clients. A solution is to deploy edge servers creating a non-intrusive illusion for client endpoints. Taking encapsulation as an example, the edge server is responsible for (1) decapsulating packets going to clients, (2) maintaining an enormous table to cache decapsulated headers, and (3) looking up the table and recovering corresponding headers. It fixes the compatibility problems for clients, whereas it does not change the situation of servers and intermediate devices. Moreover, it raises two new concerns. First, querying a huge table is computationally intensive and, consequently, not scalable. Second, the effectiveness diminishes if the edge server is located far from the client endpoint. Thus, one of our primary motivations is to identify a technique that can offer high accuracy without compromising compatibility.

B. Challenges in Gaining Traffic Knowledge

Gaining traffic knowledge is essential to derive optimization signals, and hence another challenge in handling public-facing traffic. Figure 2 compares state-of-the-art methods. We categorize these methods based on their underlying causes, and expand the elaboration accordingly.

a) Low Visibility in Middleboxes: Is it possible to utilize the increase in DCN-controlled middleboxes for better traffic insight? Unfortunately, the visibility of Internet traffic is shrinking due to evolving technologies, standards, and practices. While these changes have positive intentions, they restrict traffic inference in middleboxes, and hence limit various optimization methods. First, web services and content can no longer be inferred by IP addresses. On the one hand, web services are less coupled with IP addresses. A service may be associated with multiple IP addresses, while an IP address can refer to multiple services. On the other hand, the response content of web service is not idempotent and is determined upon each request individually. Even when sending two identical requests, the second one might be times larger than the first in size due to the change in content, *e.g.,* the Twitter feeds. Second, deep packet inspection (DPI) has stronger insights into complex traffic than IP inference. However, it is at the compatible margins of today's Internet traffic as over 90% of traffic is encrypted [22], [23]. Additionally, the implementation of HSTS preload list [24] and the rolling out of HTTP/3 [25] are making traffic encryption become a mandatory standard. Therefore, it is anticipated that an effective measure should yield knowledge about public-facing flows directly, rather than relying on insights from the middlebox.

b) Client-Initiated Connection: Public-facing flows always initiated by end users increase the uncertainty of traffic patterns and distribution which further differentiate themselves from intra- and inter-DC flows. First, indirect observation, *e.g.,* flow aging [26], [27], is simple yet effective in scheduling flows with sparsely distributed sizes. In contrast, given that the majority of public-facing flows are short [4], [28], indirect observation could prove ineffective or detrimental. Second, to effectively infer flow sizes, efficient machine learning algorithms are proposed [29]. Still, the effectiveness is questionable in public data centers. Recent literature reported that traffic distributions and volumes varying from service to service and

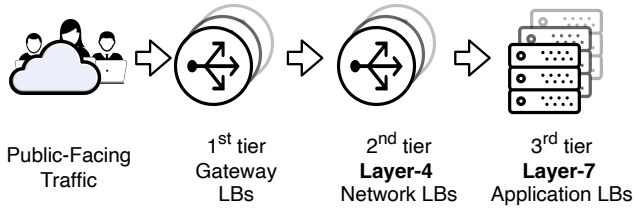


Fig. 3. Multi-tier load balancers that splits public-facing traffic in data centers.

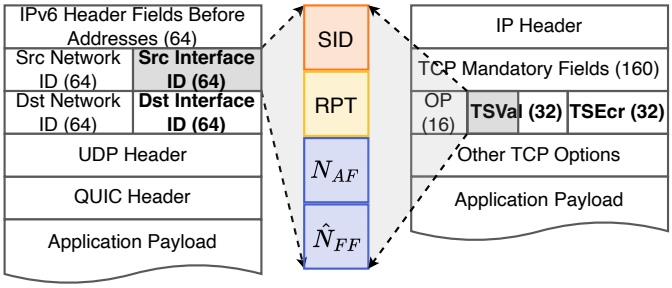


Fig. 4. Accommodations of CLOUDCOOKIE in Internet protocol headers.

time to time were observed in public-facing data centers [30]. Last, to avoid congestion and optimize bandwidth usage, traffic engineering (TE) is often adopted with SDN techniques to split and direct traffic among multiple paths. Besides the signal propagation delay indicated in §II-A, the TE controller makes a centralized decision based on collective observations, which tends to be outdated for ephemeral public-facing flows. For the same reason, TE leaves data plane optimization opportunities, *e.g.,* flow scheduling, which can further optimize network performance. In fact, the other motivation is to leverage the ignored application consciousness to gain accurate traffic knowledge rather than blindly infer it.

C. Multi-Tier Load Balancers

Independent from diverse network topology, let us view the DCN from the perspective of multi-tier load balancers (LBs). In order to serve massive requests, data center providers employ a multi-tier structure of load balancers, as shown in Figure 3, to split the huge traffic around the world [31], [32].

Under this structure, the 3rd tier Layer 7 (L7) LBs are the end hosts providing web services and terminating connections. These L7-LBs are identified through Direct IPs (DIPs). However, these DIPs are internal IP addresses and hence are unrecognizable to the outside networks. In contrast, virtual IPs (VIPs) are public IP addresses announced to the DNS. A web service is identified through one or a set of VIPs, and it can also be hosted on one or many L7-LBs, *i.e.,* multiple DIPs. This decoupled design of VIP and DIP benefits cloud tenants with the flexibility to configure their services.

The 1st tier load balancers are data center *gateways* performing Equal-Cost Multi-Path (ECMP) load balancing to forward incoming packets from the Internet to the next hops. While the 1st tier load balancer is responsible for assigning the packet to one of the 2nd tier Layer 4 (L4) LBs that recognizes this VIP. It is the L4-LB's responsibility to assign the packet to a specific L7-LB by translating this VIP to a DIP of that L7-LB. Either the serving, 1st tier or 2nd tier, load balancers may vary for different incoming packets of the same connection, but these packets have to terminate on the same 3rd tier L7-LB, *i.e.,* per-connection consistency (PCC). Otherwise, a different L7-LB cannot recognize the packet and respond properly.

L4-LBs can be categorized as load-agnostic or load-aware. Load-agnostic balancers, like round-robin and hash-based systems, perform inconsistently, especially during off-peak periods, as they do not use load information [33]. In contrast, load-aware balancers (both statefule [31], [34] and stateless [35], [36]) measure load distribution to make informed assignments. However, stateful variants struggle with scalability, and neither type meets the ideal L4-LB properties (§IV-B).

D. Flow Packet Scheduling

Data centers handling public-facing flows face a unique challenge due to the heavy-tailed distribution of flow sizes [1], where a minority of long flows account for most of the traffic, while the majority are shorter flows. Implementing the optimal SRPT flow scheduling is, however, NP-hard. Achieving near-optimal average FCT in this context has seen advances through flow scheduling on both switches [4], [10], [37] and end-hosts [15], [16]. The effective deployment of SRPT flow scheduling in public-facing data centers is challenging due to the difficulty in obtaining and exchanging necessary flow knowledge, *e.g.,* RPT, as explained in §II-B and §II-A.

III. CLOUDCOOKIE DESIGN

We view CLOUDCOOKIE as the signal *carrier* that establishes a synchronized in-bound communication channel for optimization signals to piggyback on public-facing traffic.

Given this, our first decision is to position CLOUDCOOKIE within the transport and network layers, as opposed to the application layer [38], stemming from the favorable tradeoff between visibility and expressiveness. Since CLOUDCOOKIE's is applied to public-facing traffic where client-side modifications are not feasible, the key idea is to ensure bidirectional presence by leveraging the inherent *automatic echo* in widely-deployed Internet protocols. Ultimately, we discover two implementations for CLOUDCOOKIE: (1) in QUIC/IPv6 networks, and (2) in TCP networks. Figure 4 illustrates the accommodation of CLOUDCOOKIE in both scenarios.

CLOUDCOOKIE, in its QUIC/IPv6 implementation, leverages the *composite* features of both protocols. Initially, we utilize a key aspect of QUIC, an emerging transport layer protocol initially developed by Google and standardized by the Internet Engineering Task Force (IETF) [39], [40], and foundational to HTTP/3 [25]. QUIC's unique `Connection IDs`, as a set of identifiers, decouples a connection from the traditional 5-tuple. Originally designed for connection migration when client-side IP addresses change, we repurpose this feature at the server-side [41] by encoding CLOUDCOOKIE within L7-LB's IPv6 addresses. Meanwhile, we harness a feature

of IPv6: An Internet device commonly receives a /64 subnet [42], bifurcating the IPv6 address into `Network ID` and `Interface ID` [42]–[44]. While `Network ID` is used for routing, the whole range of `Interface IDs` for identifying interfaces are available on host [45]. Thus, with QUIC as the transport layer protocol, CLOUDCOOKIE can be encoded into `Interface ID`, utilizing the 64 least-significant bits (LSB) of the IPv6 address space. The automatic echo feature is ensured via the IP address swap. CLOUDCOOKIE is embedded in the `Src Interface ID` for server-originated packets and in the `Dst Interface ID` for client-originated packets. CLOUDCOOKIE in QUIC/IPv6 networks provides a substantial 64 bit coding space for various optimization signals.

Inspired by Cheetah, an L4-LB that utilizes TCP Timestamps for flow assignment memory [35], CLOUDCOOKIE, in its TCP implementation, expands the use of TCP Timestamps for versatile optimization signals. TCP Timestamps is a TCP option including two equal-size parts: `TSVal` and `TSEcr`, where either one is 32 bit long. TCP Timestamps is originally designed for round-trip measurement of retransmitted packets and the Protect Against Wrapped Sequences (PAWS) mechanism [46], [47]. TCP Timestamps also possess the automatic echo feature. Either connection peer receiving a packet with TCP Timestamps will copy the `TSVal` value to the `TSEcr` field of its upcoming departure packets, and set the `TSVal` to be an independent value of its own interest. Specifically, once CLOUDCOOKIE is injected into `TSVal` of packets for a downstream public-facing flow, the subsequent upstream packets within the same connection will observe CLOUDCOOKIE in their `TSEcr` fields. Thus, the bidirectional visibility of CLOUDCOOKIE is guaranteed. Following Cheetah's practice to preserve the original functionality of TCP Timestamps [35], CLOUDCOOKIE conservatively occupies the 16 most significant bits (MSB) of `TSVal` or `TSEcr` so that the manipulation of CLOUDCOOKIE can be reversed at the server-side. In comparison to CLOUDCOOKIE in QUIC/IPv6 networks, CLOUDCOOKIE in TCP networks works with both IPv4 and IPv6, but has a more constrained space forcing a lower resolution of piggybacked signals.

While CLOUDCOOKIE is a versatile signal carrier decoupled from specific DCN optimization techniques, in this paper, we choose load-aware load balancing and SRPT flow scheduling. Therefore, CLOUDCOOKIE encodes SID for flow assignment, RPT for flow scheduling, and two load metrics– the active flow number (N_{AF}) and the future flow estimation ($\hat{N_{FF}}$)–for updating CCLB_7 (server) loads on CCLB_4, as the scheme shown in Figure 4. Furthermore, we advise the DCN authority shield CLOUDCOOKIE with cryptographic tools, *e.g.,* AES. It is noteworthy that only CLOUDCOOKIE, rather than the whole packet, should be encrypted. Encrypting or decrypting a maximum 64-bit string of AES-128 can be in line rate on programmable switches [48]. Encrypting can not only protect CLOUDCOOKIE against malicious modification and forgery, but also help early filter out invalid traffic.

IV. CLOUDCOOKIE IN ACTION

The lifecycle of CLOUDCOOKIE starts with CCLB_7, which acts as the signal *producer* integrating application layer knowledge into CLOUDCOOKIE. DCN middleboxes, including CCLB_4 and the scheduler-on switch, serves as the signal *consumers*, performing foresight load balancing and SRPT flow packet scheduling based on this knowledge.

A. CCLB_7: Signal Producer

The key philosophy of CCLB_7 is to extend any state-of-the-art L7-LB to integrate application awareness into CLOUDCOOKIE and thereby optimize public-facing flows. Instead of building from scratch, we patch a regular L7-LB with two additional roles to make it an optimization signal *producer*:

- **Signal *collector***: collecte flow knowledge and measure the L7-LB's own load.
- **Signal *feeder***: encode optimization signals into CLOUDCOOKIE and inject it in public-facing flows' packets.

As outlined in §III, CCLB_7 plays a crucial role in generating RPT and two types of loads (N_{AF} and $\hat{N_{FF}}$) for SRPT flow scheduling and load-aware load balancing. To illustrate its functionality, we expand upon the design of CCLB_7 within TCP networks, which operates similarly to its counterpart in QUIC/IPv6 networks. The formula for RPT is given by $RPT = \frac{flow\_size - bytes\_sent}{throughput}$. The initial step involves acquiring `bytes_sent`, conveniently available in the kernel via `struct tcp_sock`. `tcp_sock`, the internal structure managing TCP connection states, is extended with a new field `acc_flow_size`, allowing the application layer to report the flow size. Given the common practice of connection reuse, where a single connection may handle multiple data flows, this field is designed for accumulative updates. The next component involves the application layer, specifically an L7-LB, processing a new request to determine the flow size. Fortunately, the flow size for most web traffic is already known. Preferably, the flow size can be extracted directly from the request. For static object requests, the flow size can be obtained from associated metadata, a method particularly efficient for high-volume transmissions, such as on-demand video streaming, where a video is sent in chunks. For dynamic content requests, where the response size is not predetermined, the flow size can be determined after the response composition and before initiating the `send()` system call to transfer the response to the socket buffer. To accommodate flow size reporting in both scenarios, the default `send()` is replaced with `send_cc()`, which wraps the original `send()` call and updates `acc_flow_size`. There are scenarios, where the flow size is unknown, such as live streaming. Still, `acc_flow_size` keeps growing resulting in a non-small flow size, and hence maintains the flow's low priority all the time as expected. The final component, real-time connection throughput, presents significant challenges. In our study, we adopt a methodology similar to that of previous research [4], [10], [37], which involves assuming a constant bandwidth. This approach has proven to be effective in our evaluations.

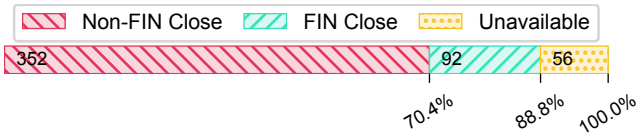


Fig. 5. A measurement over Alexa top 500 websites shows the diversity and uncertainty of TCP connection terminations.

The other optimization signal in CCLB_7 is its own loads. In addition to the current load number (N_{AF}), we propose estimating future flows ($\hat{N_{FF}}$) to maximize the use of flow knowledge. CCLB_7 utilizes two global counters for this purpose. A new flow increments counter (N_{AF}) upon invoking `send_cc()`. If the flow size surpasses a predefined "ahead-looking" threshold, indicating its likelihood of persisting in the near future, counter ($\hat{N_{FF}}$) is also incremented. Conversely, when the remaining bytes ($flow\_size - bytes\_sent$) fall below this threshold, the counter ($\hat{N_{FF}}$) is decremented. Similarly, counter(N_{AF}) is decremented once the remaining bytes reach zero. It is important to note that these increment/decrement operations are at the flow level. In cases of multiple requests reusing the same connection (in HTTP/1.1+) or server-pushes (in HTTP/2+), both counters react accordingly.

Offloading challenging tasks of flow and load measurements from DCN-controlled middleboxes to CCLB_7, paradoxically, adds minimal overhead, as all extended operations have O(1) complexity. These signals are then fed into CLOUDCOOKIE for use by stateless DCN-controlled middleboxes to execute relevant optimizations.

B. Middleboxes: Signal Consumers

To fully harness the potential of CLOUDCOOKIE's carried signals, we have engineered DCN-controlled middleboxes tailored for load-aware load balancing and SRPT flow scheduling. This includes the creation of CCLB_4, which utilizes the loads reported from CLOUDCOOKIE, and the porting of a state-of-the-art flow scheduler, AIFO [10], to public-facing DCNs by supporting CLOUDCOOKIE. This subsection elaborates on the complete end-to-end lifecycle of CLOUDCOOKIE and illustrates how DCN-controlled middleboxes utilize it to optimize public-facing traffic.

a) Cheetah's Teachings: Shaping the Ideal Load Balancing: CCLB_4 draws inspiration from Cheetah [35] by using packet headers to store flow assignments, thereby implementing a stateless L4-LB that ensures PCC. Cheetah has two variants: Cheetah_{RR} for round-robin and Cheetah_{LL} for least-loaded load balancing. However, even Cheetah_{LL}, which considers L7-LB loads in flow assignments, exhibits inherent design flaws. First, Cheetah_{LL} independently measures loads, leading to biased observations. This is due to its method of counting active flows, incrementing or decrementing based on `SYN` or `FIN` packets, which does not reliably indicate connection terminations in real-world scenarios. Our analysis of Alexa top-ranking websites [49] revealed that less than 20% use `FIN` for graceful connection closure, as Figure 5 shows.

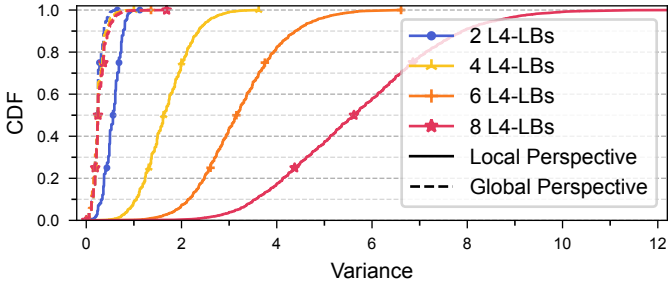


Fig. 6. CDF of cross-L7-LB variance in flow assignments over time, from L4-LBs' local and global perspectives.

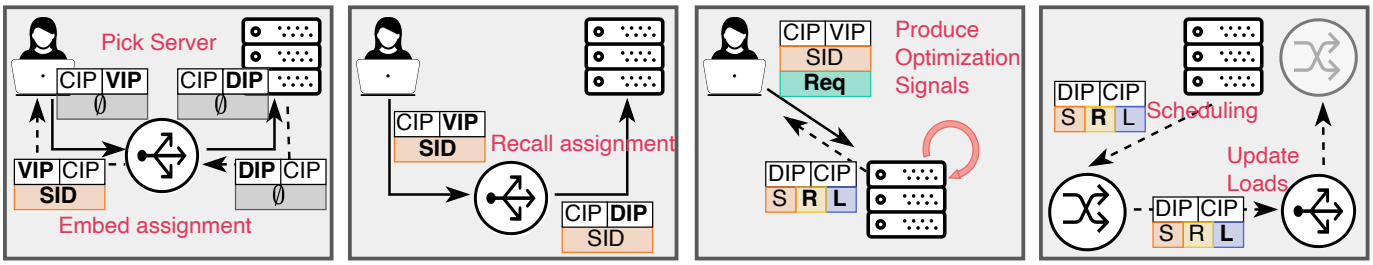
Second, even with accurate active flow counting, the locally measured loads are skewed compared to the global distribution. To illustrate this, we simulate the distribution of 10,000 flows, each with sizes sampled uniformly, from $N \in [2, 4, 6, 8]$ L4-LBs to 16 L7-LBs from both local and global perspectives. The results, shown in Figure 6, indicate higher variance in assignments from the local perspective, worsening as the number of L4-LBs, *i.e.,* N, increases.

Last, even with accurate measurements of global load distributions, optimizing public-facing flows remains inadequate if flow awareness is ignored. As discussed in §II, public-facing flows exhibit high uncertainty. A representative uncertainty is the heavy-tailed distribution in flow sizes [1], [28], necessitating future load estimation to rectify flow assignment decisions. For instance, consider assigning an incoming flow between two L7-LBs, A with 15 active flows and B with 20. Cheetah_{LL} would assign the flow to A. However, if 5 and 15 flows terminate on A and B respectively within the next second, the load distribution will reverse. Cheetah_{LL} does not account for flow awareness and leads to suboptimal flow assignments.

From the lessons of Cheetah's design, we derive that the ideal L4-LB should encompass three core properties: *precise measurement*, a *global perspective*, and *flow awareness*. CCLB_4 is designed to embody these properties by utilizing the load information reported by each CCLB_7 through CLOUDCOOKIE. First, the loads reported by CCLB_7 are the ground truths. Second, the continuous update of load information in every packet sent to CCLB_4 endows it with an approximate global perspective. Third, the loads reported in CLOUDCOOKIE, consisting of N_{AF} and $\hat{N_{FF}}$, capture both the current and future load levels. CCLB_4 maintains a set of maps to link VIPs, SIDs, DIPs, and loads of CCLB_7s.

Figure 7 outlines the workflow of DCN-controlled middleboxes, including CCLB_4 and scheduler-on switches, in response to CLOUDCOOKIE. Depending on CLOUDCOOKIE content and flow states, the workflow is divided into four phases: **Assigning Phase** (Figure 7 (a)): This phase begins with a client initiating a connection to a VIP obtained from DNS. Upon receiving the first packet (TCP's `SYN` or QUIC's `Initial`) at CCLB_4, candidate CCLB_7s associated with the VIP are considered. CCLB_4 picks the candidate with the lowest load score. Score the i-th CCLB_7 as:

$$\text{Score}(i) = \alpha \times N_{AF}^{i} + (1 - \alpha) \times \hat{N}_{FF}^{i} \qquad (1)$$



(a) Assigning (b) Retrieving (c) Insighting (d) Reacting

Fig. 7. Workflow depicting the production and consumption of CLOUDCOOKIE in DCN-controlled infrastructure. Packets are shown in the scheme of SrcIP | DstIP / CLOUDCOOKIE. "CIP" refers to the client IP. Solid and dashed arrows indicate packets in upstream and downstream directions.

where α is a weight coefficient to balance the load balancing sensitivity between the active (N^i_{AF}) and estimated future ($\hat{N}^i_{FF}$) flow numbers. The destination address in the packet is then changed from VIP to DIP and forwarded to the selected $CCLB_7$. In the downstream direction, any packet traversing $CCLB_4$ has its source address changed back from DIP to VIP, with the corresponding SID embedded in CLOUDCOOKIE. **Retrieving Phase** (Figure 7 (b)): This phase happens for every subsequent upstream packet passing $CCLB_4$. SIDs in CLOUDCOOKIE are used to retrieve the previous assignment, ensuring consistent delivery to the same $CCLB_7$. **Insighting Phase** (Figure 7 (c)): Upon each complete request delivery, $CCLB_7$ collects flow knowledge, such as RPT, during request parsing or response composition. This flow knowledge, combined with load information, is packed into CLOUDCOOKIE. **Reacting Phase** (Figure 7 (d)): In the downstream path, $CCLB_4$ updates the load information, while scheduler-on switches, even in client-side eyeball networks, utilize SRPT flow scheduling for proactive bandwidth allocation optimization.

CLOUDCOOKIE imparts *stateful* capabilities to DCN-controlled middleboxes while allowing them to remain inherently *stateless*. This aspect is critical for enabling data center advances without compromising scalability. Furthermore, CLOUDCOOKIE facilitates the synergistic optimization on public-facing traffic by fusing multiple data center advances.

V. EVALUATION

In the development of CLOUDCOOKIE and its associated infrastructure, we have executed two Proof-of-Concepts (PoCs) to validate various facets of our design. The first, a *Functional PoC* extending the Linux kernel, demonstrates CLOUDCOOKIE's basic viability in standard operating systems and public-facing traffic. In this PoC, we also evaluate the overhead associated with integrating CLOUDCOOKIE into $CCLB_7$. The second, a *Performance-Oriented PoC*, utilizes a DPDK-based HTTP suite and L4-LBs implemented on BESS framework [50]–[52], and a Tofino (P4) switch [53], [54] for running the flow scheduler, to conduct performance assessments, particularly in high-speed data center environments. With this implementation, we simulate real-world HTTP traffic and evaluate the effectiveness of CLOUDCOOKIE's approach in 100-Gbps high-speed network settings, thereby demonstrating its potential to substantially improve network performance.

In this evaluation, we formulate five key questions:

- Does load balancing and flow scheduling have a synergistic impact on network performance, given that they focus on different aspects of optimization? (§V-A)
- Is stateless $CCLB_4$ able to improve load balancing effectiveness by integrating load reports from $CCLB_7$ via CLOUDCOOKIE, compared to its alternatives? (§V-B)
- Is CLOUDCOOKIE able to enable SRPT flow scheduling for public-facing flows? (§V-C)
- What is the performance of TCP CLOUDCOOKIE and $CCLB_4$ under different scoring settings? (§V-D)
- What is the cost of introducing CLOUDCOOKIE? (§V-E)

a) Performance-Oriented PoC Testbed: We have the AIFO [10] flow scheduler deployed on a Tofino switch. Four physical machines are connected to the switch via 100-GbE links. Each of these machines has one AMD EPYC™ 7302P 16-core processor at 3.0 GHz with hyper-threading disabled. Four HTTP clients along with four independent L4-LBs are deployed on one dedicated machine. Traffic from each HTTP client will arrive first at its dedicated L4-LB. Then, the routed packets will be forwarded by the switch to twelve L7-LBs running at the three remaining machines. The switch-clients link is the bottleneck link because these clients share the same link. For performance isolation, we run each HTTP-client-L4-LB pipeline with dedicated CPU cores and a virtual NIC by using SR-IOV [55]. [2] Likewise, the other three machines each run four $CCLB_7$s, where each one is allocated dedicated cores and its own virtual NIC.

While it may differ from canonical DCN deployments where client traffic traverses multiple layers of switches, this PoC testbed abstracts these layers and focuses on the aggregation point where traffic converges before reaching the L7-LBs. This setup simulates and optimizes traffic through a single Top-of-Rack (ToR) switch, modeling a rack-scale network.

b) Experimental Settings: We simulate real-world public-facing flows from a well-known data center websearch workload [1], which matches the practice in precedent research [1], [4], [10], [15], [37], [56]. In this heavy-tailed

[2] In this way, pipelines can run with minimal interference because packets are switched in hardware on host.

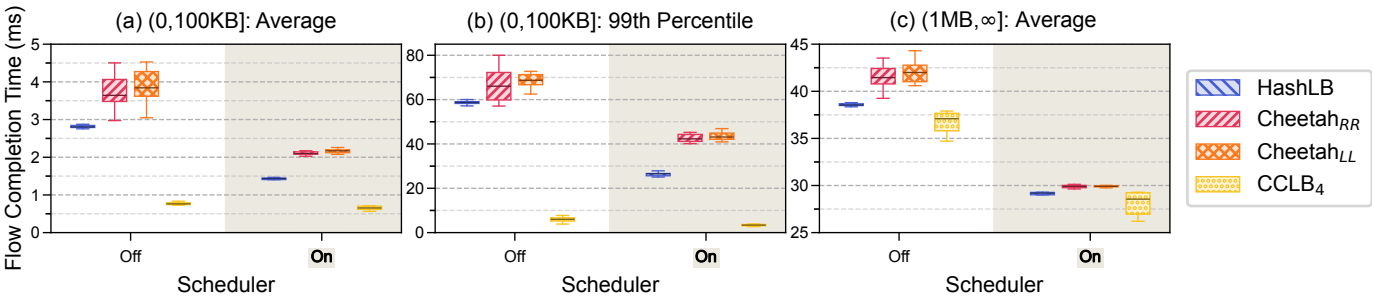


Fig. 8. Performance of different L4-LBs with or without flow scheduler under high load (0.8) demonstrates the individual and synergistic performance improvements brought by CLOUDCOOKIE.

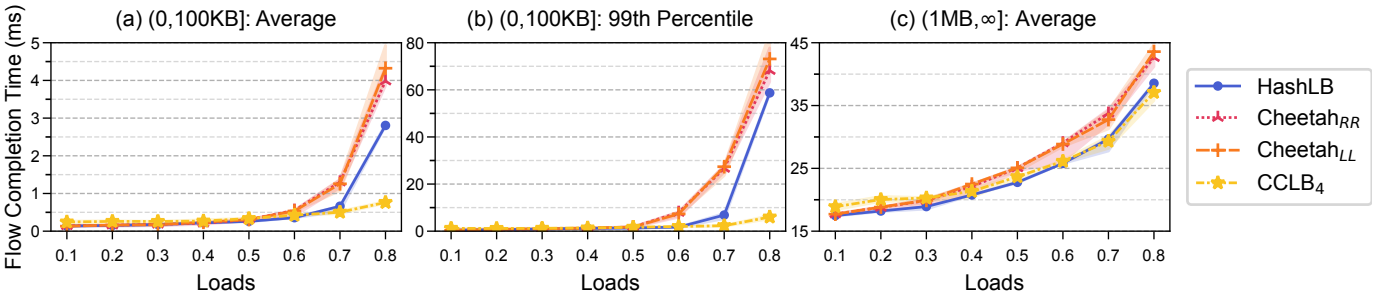


Fig. 9. Performance of different L4-LBs under varying loads (scheduler disabled) further confirms the CCLB$_4$'s effectiveness. The shaded area indicates the interquartile range (IQR) of each L4-LB.

workload, the flow size varies from 8 KB to 28 MB and has a 95th percentile of 4 MB. On each HTTP client, we generate flow arrival events, *i.e.,* sending GET requests, through the Poisson process. Client requests are distributed to L7-LBs by associated L4-LBs. Four L4-LBs are considered in this study: (1) HashLB, which assigns flows based on the 5-tuple hash value; (2) Cheetah$_{RR}$, which memorizes flow assignments in TCP Timestamps and uses round-robin for new assignments; (3) Cheetah$_{LL}$, a variant of Cheetah that selects the least loaded CCLB$_7$ based on local measurements for new assignments [35]; and (4) CCLB$_4$, which utilizes CLOUDCOOKIE-reported loads for flow assignment decisions. The first two are load-agnostic, while the latter two are load-aware. The target network loads vary from 0.1 to 0.8. Each experimental setting involves 10 repetitions, with requests emitted for 10 seconds in each run, unless otherwise specified. We primarily evaluate the performance of CLOUDCOOKIE in QUIC/IPv6 networks. The default weight coefficient α in the CCLB$_4$ score function (Eq 1) is 0.5 to balance loads N_{AF} and $\hat{N}_{AF}$ equally.

A. Studying The Synergy of Multiple Optimizations

Flow scheduling optimizes bandwidth allocation for network flows in local links, while load balancing distributes traffic to enhance overall network efficiency. To assess the synergistic benefits of CLOUDCOOKIE, we concentrate on high-load scenarios (load=0.8), comparing CCLB$_4$ against HashLB, Cheetah$_{RR}$, and Cheetah$_{LL}$, with the flow scheduler either activated or deactivated.

Figure 8 presents the average FCT for short flows (0, 100KB], the 99th percentile FCT for these flows, and the average FCT for long flows (1MB, ∞]. Without the flow scheduler, CCLB$_4$ improves performance for both short and long flows: it achieves a 4-5x reduction in average FCT and a 10-11x in the 99th percentile FCT for short, the majority, flows compared to alternatives. For long flows, CCLB$_4$ reduces the average FCT by up to 12%. After enabling the flow scheduler, the FCT reductions by CCLB$_4$, for short flows, are 2-3x for the average FCT and 8-12x for the 99th percentile. The absolute FCTs decrease for all L4-LBs. And, with fully utilizing CCLB$_4$ and the flow scheduler, the overall improvement in the 99th percentile FCT can reach up to 20x for the majority flows. This is because the two optimizations target different optimization aspects. And the improvement from flow scheduling is consistent with AIFO's large-scale simulation [10] despite that our RPT is extracted from each request rather than manually tagged. However, the performance gap between different L4-LBs narrows. The underlying rationale is that applying flow packet scheduling can decrease packet retransmissions and indirectly reduce the overall network load. In summary, flow packet scheduling and load balancing can collaboratively enhance the efficiency of public-facing flows.

B. Demonstrating The Effectiveness of CCLB$_4$ Design

From the trailer in §V-A, CCLB$_4$ consistently outperforms its competitors with or without the flow scheduler under high loads. Its performance is notably superior in managing both short and long flow types. We now conduct an ablation study to further understand CCLB$_4$'s effectiveness.

First, we disable the flow scheduler and compare the four L4-LBs under varying loads. Results are shown in Figure 9. Under light traffic load (< 0.6), all L4-LBs demonstrate comparable service levels. However, for higher load (≥ 0.6), performance gaps between L4-LBs become increasingly evident. Across various load levels, the performance ranking

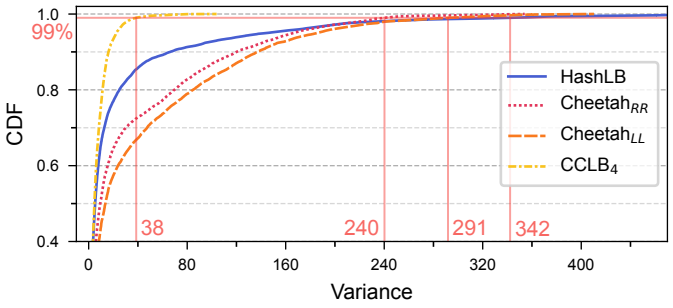


Fig. 10. The ground-truth cross-L7-LBs confirms the superiority of CCLB$_4$.

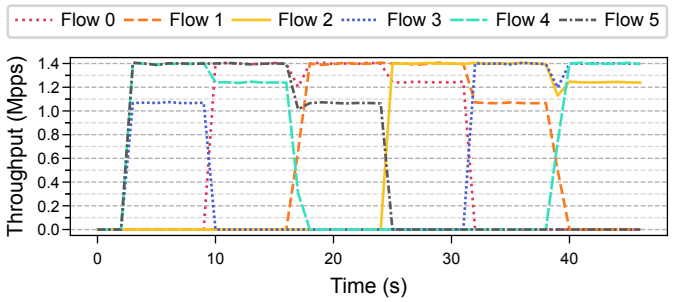


Fig. 11. Shifts in bandwidth allocation in reaction to rotating RPTs in six flows, as carried by CLOUDCOOKIE.

remains constant: CCLB$_4$ consistently outperforms all others, followed by HashLB, and then the two Cheetah variants.

We are taken aback by the underperformance of Cheetah L4-LBs, especially when compared to HashLB. But this indeed validates our earlier observation: the Cheetah's ignorance discussed in §IV-B. According to Figure 9, we notice that Cheetah$_{LL}$ performs even worse than Cheetah$_{RR}$. This is understandable: Even though Cheetah$_{RR}$ is load-agnostic, it operates under the assumption that server loads are uniformly distributed. In contrast, Cheetah$_{LL}$ tends to result in more erroneous flow assignments, because of its biased perspective based on locally measured server load distributions. CCLB$_4$ adopts a fundamentally different strategy by fully offloading load measurement to the server, *i.e.,* CCLB$_7$, with updates facilitated by CLOUDCOOKIE. Notably, CCLB$_7$ is inherently the most credible source for its own load data.

To further verify the load balancing performance, we set up Chrony [57] to elevate the time accuracy among machines, reducing it from several seconds to approximately ten microseconds. Then we periodically dump real-time active flow numbers on the twelve CCLB$_7$s and align them to calculate the variance. Figure 10 is the CDF of the variance collected for different L4-LBs, further confirming the superiority of CCLB$_4$ in load balancing. In summary, serving CLOUDCOOKIE, stateless CCLB$_4$ can perform load-aware load balancing from an approximate global perspective.

C. Validating SRPT Flow Scheduling

In addition to results in §V-A, we design an experiment of 6 flows to verify that CLOUDCOOKIE is effective in enabling SRPT scheduling for public-facing flows. We conduct an experiment by bypassing the L4-LB and directly transmitting 6

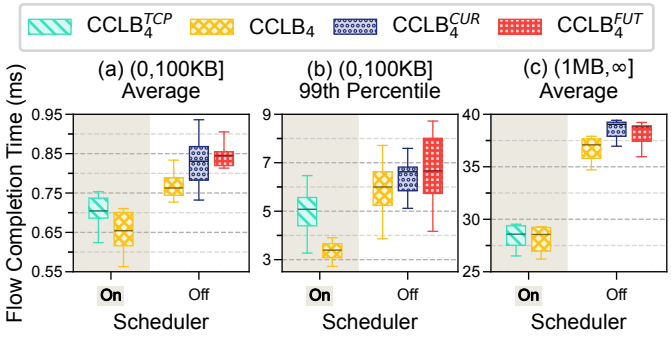


Fig. 12. Performance of CCLB$_4$ variants under high load (0.8).

continuous data flows from a server to a client, operating on a best-effort basis. Each flow is uniquely identified by an RPT assigned via CLOUDCOOKIE, which determines its priority. Initially, the flows, from Flow 0 to Flow 5, are arranged with RPTs in descending order, corresponding to ascending priorities. Every 5 seconds, the RPT assignments are rotated. For instance, after the first rotation, the RPTs are reordered in descending order as Flow Flow 1 to Flow 5, and Flow 0.

Figure 11 shows the throughput of every flow over time. Initially, only Flow 3, 4, and 5 grab shares of the bandwidth, because the CLOUDCOOKIE of them carries the shortest RPTs in that period. Later, after the first rotation, Flow 3 quickly handles its bandwidth share to Flow 4, 5 and the newcomer, Flow 0, due to the RPT reranking. Similar changes apply to subsequent rotations. Besides, during each stable period, we can observe the flow throughputs are divided into 3 tiers: the two flows with the shortest RPT take the maximum throughput, followed by the third shortest RPT flow. The remaining flows do not get any bandwidth share. CLOUDCOOKIE is hereby proved to be an effective and flexible signal carrier to enable SRPT flow packet scheduling.

D. More Experiments on CCLB$_4$ Variants

CLOUDCOOKIE working in TCP networks and carrying lower-resolution signals creates the CCLB$_4^{TCP}$ variant. There are two additional variants: CCLB$_4^{CUR}$ and CCLB$_4^{FUT}$, due to different flow assignment flavors: only from the number of current active flows ($\alpha = 1, Score(i) = N^i_{AF}$), and only from the estimation of future flows ($\alpha = 0, Score(i) = \hat{N}^i_{FF}$).

Figure 12 shows the comparative results of these variants alongside the default CCLB$_4$. The initial comparison focuses on CCLB$_4$ targeting QUIC/IPv6 networks versus CCLB$_4^{TCP}$. Although CCLB$_4^{TCP}$ shows an 8% increase in average FCT and a 50% increase in the 99$^{\text{th}}$ percentile FCT compared to CCLB$_4$, its performance still surpasses that of CCLB$_4$ when the flow scheduler is disabled. With a properly designed coding pattern as in Figure 4 (b), CCLB$_4^{TCP}$ can still achieve competitive performance. The second comparison evaluates various flow assignment flavors, with the flow scheduler disabled. As anticipated, CCLB$_4$, which accounts for both the current number of active flows and future flow estimations, outperforms CCLB$_4^{CUR}$ and CCLB$_4^{FUT}$, each of which considers only one of these aspects. These results highlight the practicality of our

TABLE I
CPU UTILIZATION OF $CCLB_7$ FUNCTIONAL POC

		Idle	Vanilla L7-LB	$CCLB_7$
Mean	User	0.4%	6.1%	6.6%
	Kernel	0.4%	15.7%	16.6%
Median	User	0.0%	6.3%	6.6%
	Kernel	0.0%	15.8%	16.9%
P99	User	4.0%	7.7%	8.4%
	Kernel	2.4%	16.7%	18.4%

heuristic approach in estimating future flows from current flow RPTs, thereby leveraging application layer flow awareness to enhance flow optimization.

E. Quantifying CloudCookie's Overheads

While CLOUDCOOKIE offloading flow insighting and load measurement to $CCLB_7$ from DCN middleboxes introduces additional responsibilities, the overhead remains minimal, as detailed in §IV-A, with all operations being of constant complexity. This subsection quantifies such overhead in both Performance-Oriented and Functional PoCs.

We measure the CPU cycles required for inferring RPT and loads per outgoing packet: load retrieval consistently requires 30 cycles, being a simple memory read, while RPT computation varies between 30 (median) and 630 (99^{th} percentile) cycles. On a 3.0 GHz processor, this translates to a delay of 0.22 μs per packet at the tail, a negligible amount compared to the typical RTT of public-facing flows, *e.g.,* Facebook's reported median RTT of 39 ms [58].

Additionally, we evaluate the overhead of $CCLB_7$ in its Functional PoC, comparing it to a vanilla L7-LB without $CCLB_7$ functionality. The results in Table I show a 1.4% increase in total CPU utilization for both mean and median, and a 2.4% increase for the 99^{th} percentile. This modest rise is negligible compared to the L7-LB's core function of web service provision, accounting for approximately 20% CPU utilization. The alignment of results from both Functional and Performance-Oriented PoCs validates the minimal overheads of handling public-facing traffic with CLOUDCOOKIE.

Further, a line of research has explored performance optimizations in software and hardware for network applications [59]–[63] and can be utilized to enhance the performance of CLOUDCOOKIE's implementation. We left it for future work.

VI. DISCUSSION

a) More Applications: We have explored CLOUDCOOKIE's use for flow scheduling and load balancing by utilizing RPT and load information due to their quantifiable nature. CLOUDCOOKIE's versatility allows it to be applid in a variety of applications. Here are some examples: (1) Network telemetry is crucial for detecting and predicting network anomalies, where CLOUDCOOKIE can be a drop-in replacement for existing telemetry systems. It eliminates the need for additional control traffic, such as required by NetFlow [64], IPFIX [65], or tunneling protocol used in INT [66]. (2) Decryption [67] and DPI are required for traffic auditing but can expose sensitive personal data and compromise privacy. CLOUDCOOKIE provides a solution by tagging sensitive traffic with multidimensional security signals for auditing purposes, with no need to inspect the content. (3) Quality of experience (QoE) or special traffic treatments are often deployed for better user experience [38]. CLOUDCOOKIE enables fine-grained traffic control, requiring no client modifications.

b) QUIC/IPv6 Consideration: QUIC's connection migration is flexible but is not free, as it requires extra traffic and computation for path validation. Thus, we suggest an asymmetric CLOUDCOOKIE pattern, where dynamic signals, *e.g.,* RPT and loads, are masked out when downstream traffic leaves the DCN, and only static signals, *e.g.,* SID, are preserved in upstream traffic. Moreover, QUIC forbids connection migration during handshaking. For this reason, CLOUDCOOKIE will not be set during this period. An ephemeral connection table is required on $CCLB_4$ to ensure PCC.

c) Limitations: CLOUDCOOKIE does not support QUIC/IPv4 networks, as it requires the composite features of both protocols. Neither IPv4 addresses nor QUIC `Connection IDs` can encode CLOUDCOOKIE due to address scarcity, and privacy and security concerns, respectively [40]. Nevertheless, this limitation only affects a small proportion of Internet traffic (4.6%, estimated based on current usage of IPv4 and QUIC [68], [69]), preventing it from benefiting from the optimizations brought by CLOUDCOOKIE.

VII. CONCLUSION

Endorsed by the expanding footprints and increasing controllability of data center authorities, we propose CLOUDCOOKIE, designed to carry flow-optimization signals within Internet protocol headers. This innovation enables private data center advances in public-facing data centers. We develop a system comprising $CCLB_7$, $CCLB_4$, and the scheduler-on switch, as the producer and consumers of flow-optimization signals. $CCLB_7$ gathers application layer insights, adding credible flow scheduling and load balancing signals to CLOUDCOOKIE. Signals are then consumed by stateless $CCLB_4$ and the scheduler-on switch to achieve better performance for public-facing traffic. Additionally, our design incurs minimal overhead and requires zero client-side modification.

Our evaluation reveals that the combining $CCLB_4$ with the SRPT flow scheduler further reduces the 99^{th} percentile FCT of the majority flows from 2-11x to 20x, compared to enabling either one individually. Also, $CCLB_4$, with an accurate, global perspective and future visions empowered by CLOUDCOOKIE, consistently outperforms its counterparts, while maintaining its stateless nature, even in TCP networks with low-resolution signals. While our exploration of CLOUDCOOKIE is in its nascent stages, we remain open to the future potential of this versatile signal carrier, which holds promise for integrating more data center advances into public-facing traffic.

References

[1] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center TCP (DCTCP)," in *Proceedings of the ACM SIGCOMM 2010 Conference*, 2010, pp. 63–74.

[2] Y. Li, R. Miao, H. H. Liu, Y. Zhuang, F. Feng, L. Tang, Z. Cao, M. Zhang, F. Kelly, M. Alizadeh *et al.*, "HPCC: High precision congestion control," in *Proceedings of the ACM Special Interest Group on Data Communication*. Association for Computing Machinery, 2019, pp. 44–58.

[3] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "Vl2: A scalable and flexible data center network," in *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, 2009, pp. 51–62.

[4] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, "pfabric: Minimal near-optimal datacenter transport," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 435–446, 2013.

[5] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu *et al.*, "B4: Experience with a globally-deployed software defined wan," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 3–14, 2013.

[6] P. Gigis, M. Calder, L. Manassakis, G. Nomikos, V. Kotronis, X. Dimitropoulos, E. Katz-Bassett, and G. Smaragdakis, "Seven years in the life of Hypergiants' off-nets," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, ser. SIGCOMM '21. New York, NY, USA: Association for Computing Machinery, Aug. 2021, pp. 516–533.

[7] T. Arnold, J. He, W. Jiang, M. Calder, I. Cunha, V. Giotsas, and E. Katz-Bassett, "Cloud Provider Connectivity in the Flat Internet," in *Proceedings of the ACM Internet Measurement Conference*, ser. IMC '20. New York, NY, USA: Association for Computing Machinery, Oct. 2020, pp. 230–246.

[8] E. Pujol, I. Poese, J. Zerwas, G. Smaragdakis, and A. Feldmann, "Steering hyper-giants' traffic at scale," in *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*, 2019, pp. 82–95.

[9] Y.-C. Chiu, B. Schlinker, A. B. Radhakrishnan, E. Katz-Bassett, and R. Govindan, "Are we one hop away from a better internet?" in *Proceedings of the 2015 Internet Measurement Conference*, 2015, pp. 523–529.

[10] Z. Yu, C. Hu, J. Wu, X. Sun, V. Braverman, M. Chowdhury, Z. Liu, and X. Jin, "Programmable packet scheduling with a single queue," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, 2021, pp. 179–193.

[11] S. Floyd, D. K. K. Ramakrishnan, and D. L. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," RFC 3168, Sep. 2001.

[12] D. B. Grossman, "New Terminology and Clarifications for Diffserv," RFC 3260, Apr. 2002.

[13] A. R. Curtis, W. Kim, and P. Yalagandula, "Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection," in *2011 Proceedings IEEE INFOCOM*. IEEE, 2011, pp. 1629–1637.

[14] A. Custura, G. Fairhurst, and R. Secchi, "Considerations for Assigning a New Recommended Differentiated Services Code Point (DSCP)," RFC 9435, Jul. 2023.

[15] P. X. Gao, A. Narayan, G. Kumar, R. Agarwal, S. Ratnasamy, and S. Shenker, "phost: Distributed near-optimal datacenter transport over commodity network fabric," in *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*, 2015, pp. 1–12.

[16] B. Montazeri, Y. Li, M. Alizadeh, and J. Ousterhout, "Homa: A receiver-driven low-latency transport protocol using network priorities," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, 2018, pp. 221–235.

[17] K.-K. Yap, M. Motiwala, J. Rahe, S. Padgett, M. Holliman, G. Baldus, M. Hines, T. Kim, A. Narayanan, A. Jain *et al.*, "Taking the edge off with espresso: Scale, reliability and programmability for global internet peering," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017, pp. 432–445.

[18] B. Schlinker, H. Kim, T. Cui, E. Katz-Bassett, H. V. Madhyastha, I. Cunha, J. Quinn, S. Hasan, P. Lapukhov, and H. Zeng, "Engineering egress with edge fabric: Steering oceans of content to the world," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017, pp. 418–431.

[19] T. Li, D. Farinacci, S. P. Hanks, D. Meyer, and P. S. Traina, "Generic Routing Encapsulation (GRE)," RFC 2784, Mar. 2000.

[20] J. Gross, I. Ganga, and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation," RFC 8926, Nov. 2020.

[21] C.-Y. Hong, M. Caesar, and P. B. Godfrey, "Finishing flows quickly with preemptive scheduling," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 127–138, 2012.

[22] Google Transparency Report, "HTTPS encryption on the web," Available: https://transparencyreport.google.com/https/overview, (Accessed on 09/01/2024).

[23] D. Naylor, A. Finamore, I. Leontiadis, Y. Grunenberger, M. Mellia, M. Munafò, K. Papagiannaki, and P. Steenkiste, "The cost of the" s" in https," in *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*, 2014, pp. 133–140.

[24] Chromium, "HSTS Preloaded list," Available: https://cs.chromium.org/chromium/src/net/http/transport_security_state_static.json, (Accessed on 09/01/2024).

[25] M. Bishop, "HTTP/3," RFC 9114, Jun. 2022.

[26] I. A. Rai, G. Urvoy-Keller, M. K. Vernon, and E. W. Biersack, "Performance analysis of las-based scheduling disciplines in a packet switched network," *ACM SIGMETRICS Performance Evaluation Review*, vol. 32, no. 1, pp. 106–117, 2004.

[27] W. Bai, L. Chen, K. Chen, D. Han, C. Tian, and H. Wang, "Information-Agnostic flow scheduling for commodity data centers," in *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*. Oakland, CA: USENIX Association, May 2015, pp. 455–468.

[28] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, 2010, pp. 267–280.

[29] V. Đukić, S. A. Jyothi, B. Karlas, M. Owaida, C. Zhang, and A. Singla, "Is advance knowledge of flow sizes a plausible assumption?" in *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*. Boston, MA: USENIX Association, Feb. 2019, pp. 565–580.

[30] C. Zeng, L. Luo, T. Zhang, Z. Wang, L. Li, W. Han, N. Chen, L. Wan, L. Liu, Z. Ding *et al.*, "Tiara: A scalable and efficient hardware acceleration architecture for stateful layer-4 load balancing," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, 2022, pp. 1345–1358.

[31] P. Patel, D. Bansal, L. Yuan, A. Murthy, A. Greenberg, D. A. Maltz, R. Kern, H. Kumar, M. Zikos, H. Wu *et al.*, "Ananta: Cloud scale load balancing," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 207–218, 2013.

[32] M. Fayed, L. Bauer, V. Giotsas, S. Kerola, M. Majkowski, P. Odintsov, J. Sitnicki, T. Chung, D. Levin, A. Mislove *et al.*, "The ties that unbind: Decoupling ip from web services and sockets for robust addressing agility at cdn-scale," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, 2021, pp. 433–446.

[33] D. E. Eisenbud, C. Yi, C. Contavalli, C. Smith, R. Kononov, E. Mann-Hielscher, A. Cilingiroglu, B. Cheyney, W. Shang, and J. D. Hosein, "Maglev: A fast and reliable software network load balancer," in *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*. Santa Clara, CA: USENIX Association, Mar. 2016, pp. 523–535.

[34] R. Miao, H. Zeng, C. Kim, J. Lee, and M. Yu, "Silkroad: Making stateful layer-4 load balancing fast and cheap using switching asics," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017, pp. 15–28.

[35] T. Barbette, C. Tang, H. Yao, D. Kostić, G. Q. M. Jr., P. Papadimitratos, and M. Chiesa, "A High-Speed Load-Balancer design with guaranteed Per-Connection-Consistency," in *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. Santa Clara, CA: USENIX Association, Feb. 2020, pp. 667–683.

[36] V. Olteanu, A. Agache, A. Voinescu, and C. Raiciu, "Stateless datacenter load-balancing with beamer," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. Renton, WA: USENIX Association, Apr. 2018, pp. 125–139.

[37] A. G. Alcoz, A. Dietmüller, and L. Vanbever, "SP-PIFO: Approximating Push-In First-Out behaviors using Strict-Priority queues," in *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. Santa Clara, CA: USENIX Association, Feb. 2020, pp. 59–76.

[38] Y. Yiakoumis, S. Katti, and N. McKeown, "Neutral net neutrality," in *Proceedings of the 2016 ACM SIGCOMM Conference*, 2016, pp. 483–496.

[39] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar *et al.*, "The quic transport protocol: Design and internet-scale deployment," in *Proceedings of the conference of the ACM special interest group on data communication*, 2017, pp. 183–196.

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

[41] C. Puliafito, L. Conforti, A. Virdis, and E. Mingozzi, "Server-side quic connection migration to support microservice deployment at the edge," *Pervasive and mobile computing*, vol. 83, p. 101580, 2022.

[42] IAB, "IAB/IESG Recommendations on IPv6 Address Allocations to Sites," RFC 3177, Sep. 2001.

[43] D. S. E. Deering and B. Hinden, "IP Version 6 Addressing Architecture," RFC 4291, Feb. 2006.

[44] E. Nordmark, D. S. E. Deering, and B. Hinden, "IPv6 Global Unicast Address Format," RFC 3587, Aug. 2003.

[45] M. Żenczykowski and The Linux Kernel Organization, "ipv6: Implement Any-IP support for IPv6 - Linux Kernel," Available: https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git, Sep. 2010, (Commit ID: ab79ad14a2d51e95f0ac3cef7cd116a57089ba82).

[46] D. A. Borman, R. T. Braden, and V. Jacobson, "TCP Extensions for High Performance," RFC 1323, May 1992.

[47] D. Borman, R. T. Braden, V. Jacobson, and R. Scheffenegger, "TCP Extensions for High Performance," RFC 7323, Sep. 2014.

[48] X. Chen, "Implementing aes encryption on programmable switches via scrambled lookup tables," in *Proceedings of the Workshop on Secure Programmable Network Infrastructure*, 2020, pp. 8–14.

[49] Alexa, "The top 500 sites on the web," Available: https://www.alexa.com/topsites, (Accessed on 04/22/2022).

[50] Linux Foundation, "Data Plane Development Kit (DPDK)," Available: http://www.dpdk.org, (Accessed on 09/01/2024).

[51] NetSys, "Berkeley Extensible Software Switch (BESS)," Available: https://github.com/NetSys/bess, (Accessed on 11/01/2023).

[52] S. Han, K. Jang, A. Panda, S. Palkar, D. Han, and S. Ratnasamy, "Softnic: A software nic to augment hardware," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2015-155, May 2015.

[53] Intel, "The Intel Tofino series of P4-programmable Ethernet switch ASICs," Available: https://www.intel.com/content/www/us/en/products/details/network-io/intelligent-fabric-processors/tofino.html, (Accessed on 11/01/2023).

[54] "P4 Programming Language," Available: https://p4.org/, (Accessed on 10/10/2022).

[55] Intel LAN Access Division, "PCI-SIG SR-IOV Primer: An Introduction to SR-IOV Technology (Revision 2.5)," Jan. 2011.

[56] J. Perry, A. Ousterhout, H. Balakrishnan, D. Shah, and H. Fugal, "Fastpass: A centralized" zero-queue" datacenter network," in *Proceedings of the 2014 ACM conference on SIGCOMM*, 2014, pp. 307–318.

[57] M. Lichvar and Red Hat, "The Chrony Project," Available: https://chrony-project.org/, (Accessed on 11/01/2023).

[58] B. Schlinker, I. Cunha, Y.-C. Chiu, S. Sundaresan, and E. Katz-Bassett, "Internet performance from facebook's edge," in *Proceedings of the Internet Measurement Conference*, 2019, pp. 179–194.

[59] J. Wang, S. Gupta, M. A. M. Vieira, B. Raghavan, and R. Govindan, "Scheduling network function chains under sub-millisecond latency slos," *arXiv preprint*, 2023.

[60] J. Wang, T. Lévai, Z. Li, M. A. M. Vieira, R. Govindan, and B. Raghavan, "Quadrant: A cloud-deployable nf virtualization platform," in *Proceedings of the 13th Symposium on Cloud Computing*, ser. SoCC '22. New York, NY, USA: Association for Computing Machinery, 2022, pp. 493–509.

[61] J. Yen, J. Wang, S. Supittayapornpong, M. A. M. Vieira, R. Govindan, and B. Raghavan, "Meeting slos in cross-platform nfv," in *Proceedings of the 16th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 509–523.

[62] J. Wang, "Performant, scalable, and efficient deployment of network function virtualization," Ph.D. dissertation, University of Southern California, 2023.

[63] B. Raghavan, R. Govindan, Z. Li, and J. Wang, "Methods and systems for efficient and secure network function execution," Jun. 2023, uS Patent App. 18/082,873.

[64] B. Claise, "Cisco Systems NetFlow Services Export Version 9," RFC 3954, Oct. 2004.

[65] P. Aitken, B. Claise, and B. Trammell, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information," RFC 7011, Sep. 2013.

[66] The P4.org Applications Working Group, "In-band Network Telemetry (INT) Dataplane Specification (Version 2.1)," The P4.org Applications Working Group, Tech. Rep., Nov. 2020.

[67] Palo Alto Networks, "SSL Forward Proxy," Available: https://docs.paloaltonetworks.com/pan-os/11-1/pan-os-admin/decryption/decryption-concepts/ssl-forward-proxy, Jul. 2024, (Accessed on 09/05/2024).

[68] Google, "IPv6 Statistics," Available: https://www.google.com/intl/en/ipv6/statistics.html, Sep. 2024, (Accessed on 09/01/2024).

[69] W3Techs, "Usage Statistics of QUIC for Websites," Available: https://w3techs.com/technologies/details/ce-quic, Sep. 2024, (Accessed on 09/01/2024).