Implementable selective deflection with Canary

Sepehr Abdous
Johns Hopkins University

Erfan Sharafzadeh
Johns Hopkins University

Soudeh Ghorbani
Meta/Johns Hopkins University

Abstract

Datacenter traffic studies unveil *microbursts*, microsecond-scale periods of high utilization that cause packet drops. Given the low average link utilization, packet deflection, *i.e.,* re-routing packets to neighboring switches when the output queue is full, can be used as a defensive mechanism against microbursts. However, naively deflecting packets can cause head-of-the-line blocking in switch buffers and exacerbate the congestion in the network by increasing the overall network utilization. To avoid these challenges, recent work selectively deflects the packets of large flows. While selective deflection is shown to ensure low latency, it imposes implementability hurdles as it requires packet extraction from arbitrary positions in the buffer. In this project, we tackle these implementability challenges in existing hardware. We propose Canary, an approximation of selective deflection that identifies the potential deflection candidate packets before they enter the switch buffers. This eliminates the need for extracting packets from the queues at runtime. Our simulation results show that Canary achieves comparable performance to selective deflection and reduces the mean query completion time by 58%, 66%, and 69% compared to ECMP, AIFO, and random deflection. Canary also reduces latency by 30% compared to the baseline drop-tail queue on a PISA switch hardware.

1 Introduction

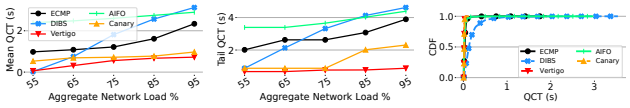
Microbursts, sudden surges in the link utilization occurring in minuscule time scales, are prevalent in datacenters [2, 8]. These short-term spikes are a main cause of packet drops, poor performance, and SLO violation [1]. A large body of work [1, 2, 4, 6, 7] attempts to either proactively avert microburst creation or reactively absorb them when they occur. Among all these proposals, selective deflection is shown to provide good reaction time and resiliency against microbursts [1]. Unfortunately, selective deflection requires complex mechanisms, such as packet extraction from arbitrary positions in the queue, to be implementable while existing buffer management technologies are limited to strictly priority FIFO queues [6].

To realize implementable selective deflection, we propose Canary, a technique that approximates selective deflection by identifying and rerouting the packets that are candidates for deflection before they enter the switch traffic manager. This eliminates the need to extract packets that are already inserted in the queue and thus removes the challenges of implementing selective deflection on commodity programmable switches. Inspired by AIFO [6], an early drop technique that prioritizes packets based on the remaining bytes of their corresponding flows to approximate Shortest Remaining Processing Time (SRPT) scheduling, Canary exploits the queue occupancy information, the priority of the newly arrived packet, and the priorities of the packets previously inserted in the queue to decide if a packet should be forwarded, deflected, or dropped as it enters the ingress pipeline. Canary uses a sliding window to compare the priority of the newly arrived packets with the previous packets that were inserted in the queue.

We compare Canary against ECMP and AIFO. We also compare Canary with DIBS [7] and Vertigo [1], representatives of random and selective deflection, respectively. Our simulations results show that Canary performs comparably to Vertigo in completing incast queries while achieving 50.90%, 38.11%, and 90.56% lower average query completion time than ECMP, AIFO, and DIBS, respectively.

2 Early Deflection

Recent work demonstrates that selectively deflecting packets of large flows can effectively address the drawbacks of random deflection, such as head-of-the-line blocking and congestion aggravation [1]. To realize selective deflection, when a packet arrives and the output queue is full, Vertigo extracts the packet with the lowest priority, *i.e.,* the largest number of remaining bytes in its corresponding flow, from the congested queue and deflects it. While this is shown to be resilient to burstiness, it requires packet extraction from arbitrary locations in the queue which is not implementable in commodity switch hardware. To overcome this challenge, we approximate selective deflection by deflecting packets before they enter the traffic manager. We propose Canary, a technique



(a) Large RTO (b) Large RTO (tail) (c) Small RTO

Figure 1: Canary outperforms ECMP, DIBS, and AIFO under different degrees of load and performs closely to Vertigo.

that uses *Early Deflection* to approximate selective deflection while avoiding its implementability challenges.

For Canary's design, we assume that packets are marked with the remaining bytes of their corresponding flows. The lower the remaining bytes, the higher the packet's priority[1]. When a packet arrives and there is enough space in the destination queue, Canary uses the destination queue occupancy information and the newly-arrived packet's priority to determine whether the packet should be enqueued or deflected. Deflecting low priority packets as they arrive leaves headroom for absorbing bursts of packets with higher priorities. If the destination queue is full, Canary deflects the packet irrespective of its priority. Since queue occupancy information is not available in the ingress pipeline of earlier releases of commodity programmable switches, we use special packets that recirculate inside the switch and transfer the queue occupancy information from the egress to the ingress pipeline [6].

While deflecting a packet, Canary uses the same technique to decide if the deflected packet should be enqueued into the selected queue or get dropped. Under severe congestion, Canary rejects the packets from being enqueued into the destination and deflection queues and drops them to signal the congestion to the endpoints.

Canary's design can adopt early drop paradigms such as Random Early Drop (RED) [3] or AIFO [6], to determine whether to forward, deflect, or drop packets as they arrive. In particular, when a packet arrives, Canary applies the early drop algorithm on the forwarding port and deflects the packet if the algorithm rejects enqueueing it. When deflecting a packet, Canary applies the same algorithm to decide whether to enqueue the packet into the deflection port or drop it.

3 Evaluation

We evaluate Canary using Omnet++ simulations. We simulate a 2-tier leaf-spine topology consisting of 4 spine switches, 8 leaf switches, and 40 machines connected to each leaf. The links connecting the spines to leafs and leafs to servers have 40Gbps and 10Gbps bandwidth, respectively. The switches have a 300KB queue capacity per port [1, 7]. We evaluate Canary under 50% background load and different degrees of incast using Facebook's cache follower and Google's web search workload [2, 5]. We compare the performance of Canary with ECMP (the forwarding protocol widely deployed in datacenters), AIFO [6] (an approximation of SRPT scheduling using early packet drop), DIBS [7] (a random packet deflection technique), and Vertigo [1] (representing selective deflection). For our evaluations, we measure the Query Completion Times (QCT), *i.e.,* the time taken from initiating the incast event to receiving all the responses.

First, we set the initial RTO to one second and the minimum RTO to 10*ms* [1,7]. Figures 1a and 1b present the results. With large RTO values, AIFO will perform poorly due to its early drop mechanism which causes more packet drops and imposes considerable latency when the RTO values are large. Canary, on the other hand, uses early deflection to avoid loss of performance with large RTO values. In particular, under 95% load, Canary achieves 58.41%, 66.41%, and 68.89% lower mean QCT and 40.72%, 47.41%, and 50.04% lower tail (99^{th} percentile) QCT than ECMP, AIFO, and DIBS, respectively. With large RTO values, Vertigo completes 12.45% more queries than Canary by dropping 34.32% fewer packets. We also test Canary with small RTO values (600μs). The results, presented in Figure 1c, illustrate that, while outperforming ECMP, AIFO, and DIBS, Canary performs close to Vertigo as the latency penalties of packet drops are lower due to small RTO values.

We also implement Canary on a Tofino switch using 1600 lines of P4 code. To test our implementation, we run Memcached and Iperf traffic simultaneously in a testbed setup consisting of three server machines connected to two Canary switches. Our initial results show that Canary achieves 30% lower mean response times than a baseline drop-tail queue for Memcached requests.

Future work. We are currently testing Canary's implementation under various traffic patterns. To further evaluate the effectiveness of early deflection, we implement Canary with other early drop paradigms, *e.g.,* RED, and other ranking paradigms, *e.g.,* Least Attained Service (LAS). We hope our approximation moves packet deflection one step forward toward practicality.

[1]There are other alternatives for marking packets, such as Least Attained Service (LAS) in which packets carry the number of bytes sent by their flow. However, marking packets based on their flows' remaining number of bytes is shown to be more effective [1].

References

[1] Sepehr Abdous et al. Burst-tolerant datacenter networks with vertigo. In *CoNEXT*, 2021.

[2] Mohammad Alizadeh et al. Data Center TCP (DCTCP). In *SIGCOMM*, 2010.

[3] Sally Floyd et al. Random early detection gateways for congestion avoidance. *IEEE/ACM ToN*, 1993.

[4] Gautam Kumar et al. Swift: Delay is Simple and Effective for Congestion Control in the Datacenter. In *SIGCOMM*, 2020.

[5] Arjun Roy et al. Inside the Social Network's (Datacenter) Network. In *SIGCOMM*, 2015.

[6] Zhuolong Yu et al. Programmable packet scheduling with a single queue. In *SIGCOMM*, 2021.

[7] Kyriakos Zarifis et al. DIBS: just-in-time congestion mitigation for data centers. In *Eurosys*, 2014.

[8] Qiao Zhang et al. High-resolution measurement of data center microbursts. In *IMC*, 2017.