IX. Supplementary Material

A. Traffic distributions

Fig. 15 includes FB_Hadoop and WebSearch, the two public traffic distributions we use in our experiments. These two distributions are widely used in research [34], [11].

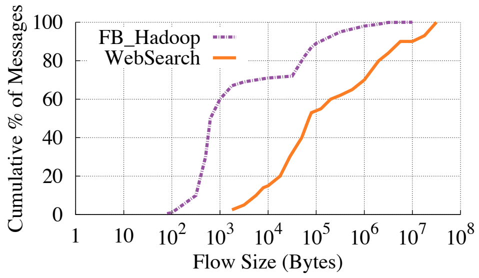


Fig. 15: The traffic workloads used in Escape evaluation.

B. DPDK Implementation

Listing 1 is the API introduced and implemented to support frame extraction based on a given key, in DPDK `struct rte_ring`. The source code of this implementation is publicly available [14].

```
typedef uint16_t (*comp_obj)
(const void *obj1, const void *obj2);
static __rte_always_inline void *
rte_ring_sc_jump(struct rte_ring *ring,
                 comp_obj func,
                 const void *obj2);
```

Listing 1: DPDK API introduced to support frame extraction.

C. Micro-Benchmarks

In § VII-A, we use the topology in Fig. 6 to evaluate Escape in terms of its ability to clear deadlocks. Here we have the link status observations from that experiment. Without Escape, l_1, l_2, and l_3 are in XOFF status after ∼18 ms as a result of deadlock formation, and remain so henceforth (Fig. 16a). Furthermore, q_1^I, q_2^I, and q_3^I remain within XOFF and XON thresholds (Fig. 8a), indicating PFC activation on corresponding upstream egress interfaces, causing deadlock. In contrast, with Escape enabled, l_1, l_2, and l_3 do not simultaneously remain in XOFF (Fig. 16b).

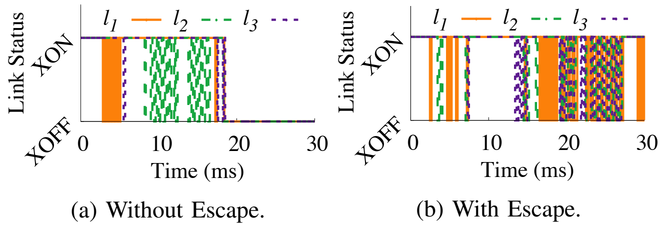
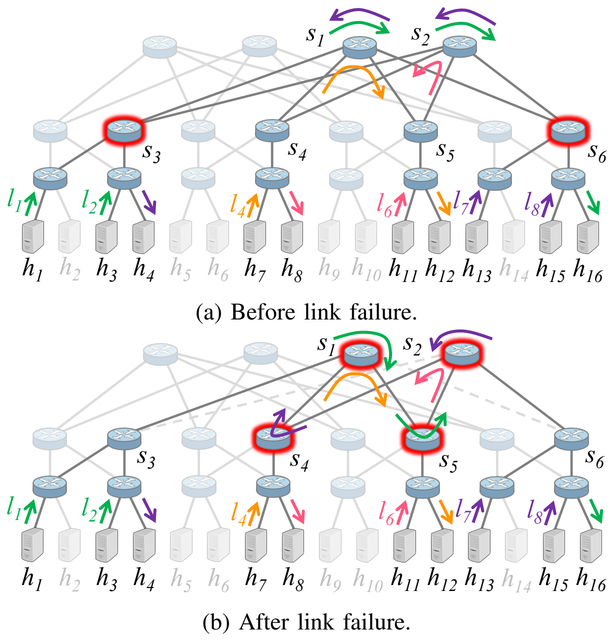


Fig. 16: Deadlock clearance in Fig. 6 with Escape.

D. Datacenter Simulations

We evaluate Escape on a scenario mimicking a modern datacenter topology, load, and traffic. We use HPCC [11] for congestion control. We use the three-level fat-tree topology depicted in Fig. 17a with 8 edge switches, 8 aggregation switches, and 4 core switches. It has 8 source nodes (h_1, h_3, h_5, …, h_{15}) and 8 destination nodes (h_2, h_4, h_6, …, h_{16}) connected to the edge switches. Each interconnection is a 40 Gbps link. Each source node starts 10 concurrent data flows. The flows from h_1 and h_3 are destined to h_{16} and create incast congestion on s_6. Similarly, the flows from h_{13} and h_{15} are destined to h_4, creating incast congestion on s_3. The flows from h_7 and h_{11} are destined to h_{12} and h_8, respectively, and create little or no congestion. Similarly, the flows from h_5 and h_9 are destined to h_{10} and h_6, respectively, creating little or no congestion. As the second step of this experiment, we disable the links $s_1 \to s_6$ and $s_2 \to s_3$, causing the flows destined to h_{16} and h_6 to take "bouncy" paths. As shown in Fig. 17b, the flows from h_1 and h_3 take the new route $s_3 \to s_1 \to s_5 \to s_2 \to s_6 \to h_{16}$ and the flows from h_{13} and h_{15} the route $s_6 \to s_2 \to s_4 \to s_1 \to s_3 \to h_3$. We run the two steps with Escape disabled and then with it enabled. We record link utilization l_i at each source node h_i to observe network stability. When Escape is enabled, we additionally record the size of ingress queue q_j^I of node s_j on the deadlock-prone path $s_1 \to s_5 \to s_2 \to s_4 \to s_1$.



s_1
s_2
s_3
s_4
s_5
s_6
l_1
l_2
l_4
l_6
l_7
l_8
h_1 h_2 h_3 h_4 h_5 h_6 h_7 h_8 h_9 h_{10} h_{11} h_{12} h_{13} h_{14} h_{15} h_{16}
(a) Before link failure.
(b) After link failure.

Fig. 17: Three-level fat-tree topology used in the datacenter simulations.

Fig. 18 shows the link utilization before link failure. The flows from h_1 and h_3 go through s_6, equally sharing the bottleneck link bandwidth, and as a result, the utilization of l_1 (and l_2) stabilizes at ∼20 Gbps. Further, the throughput attained by each of the 10 competing flows on the link, stabilizes at ∼2 Gbps. Similarly, the flows from h_{13} and h_{15} go through the bottleneck link at s_3, and the utilization of l_7 (and l_8) stabilizes at ∼20 Gbps. The flows from h_7 and h_{11} do not create congestion, hence l_4 and l_6 are fully utilized.

Without Escape, switches s_1, s_2, s_4, and s_5 instantly run into a deadlock in step two (see Fig. 17b), causing the flows going through these switches to halt. With Escape, the link failure does not cause a deadlock and each sender attains its fair share link bandwidth of ∼13.3 Gbps, as Fig. 19a shows.

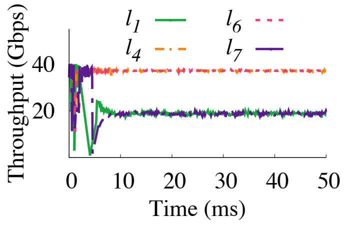
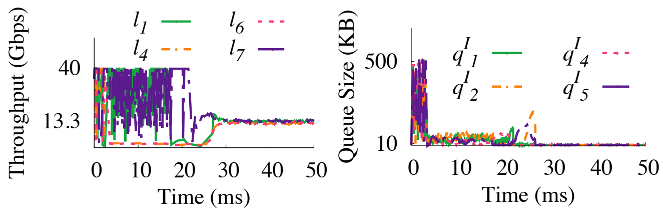


Fig. 18: System stability (link utilization) before link failure (HPCC + Escape).

To understand the impact of Escape on system stability under normal conditions and when there is a potential deadlock, we observed ingress queue depths at congestion points and instantaneous bandwidths attained by the completing senders. Fig. 18 shows that throughput convergence of competing senders (h_1, h_3 and h_{13}, h_{15}) is as expected in a network with HPCC. The system converges in ∼5 ms. As a result of the link failure, the nodes on the deadlock-prone loop become the new congestion hot spots resulting in a different set of competing senders (h_1, h_3, h_7, and h_{11}, h_{13}, h_{15}). The system converges and each competing sender attains its fair share in ∼20 ms (Fig. 19a). At the same time, the queues drain (Fig. 19b) indicating system stability. We can conclude that Escape clears deadlocks caused by HoL blocking in practical fat-tree topologies, while preserving network stability.



(a) Link utilization. (b) Queue size.

Fig. 19: System stability after link failure (HPCC + Escape).