Got Loss? Get zOVN!

Daniel Crisan, Robert Birke, Gilles Cressier, Cyriel Minkenberg and Mitch Gusat*
IBM Research – Zurich Research Laboratory, Säumerstrasse 4, CH-8803 Rüschlikon, Switzerland
{dcr,bir,cre,sil,mig}@zurich.ibm.com

ABSTRACT

Datacenter networking is currently dominated by two major trends. One aims toward lossless, flat layer-2 fabrics based on Converged Enhanced Ethernet or InfiniBand, with benefits in efficiency and performance. The other targets flexibility based on Software Defined Networking, which enables Overlay Virtual Networking. Although clearly complementary, these trends also exhibit some conflicts: In contrast to physical fabrics, which avoid packet drops by means of flow control, practically all current virtual networks are lossy. We quantify these losses for several common combinations of hypervisors and virtual switches, and show their detrimental effect on application performance. Moreover, we propose a zero-loss Overlay Virtual Network (zOVN) designed to reduce the query and flow completion time of latency-sensitive datacenter applications. We describe its architecture and detail the design of its key component, the zVALE lossless virtual switch. As proof of concept, we implemented a zOVN prototype and benchmark it with Partition-Aggregate in two testbeds, achieving an up to 15-fold reduction of the mean completion time with three widespread TCP versions. For larger-scale validation and deeper introspection into zOVN, we developed an OMNeT++ model for accurate cross-layer simulations of a virtualized datacenter, which confirm the validity of our results.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design

Keywords

Datacenter networking; virtualization; overlay networks; lossless; Partition-Aggregate

*Corresponding author: ***gusat@acm.org***

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SIGCOMM'13, August 12–16, 2013, Hong Kong, China.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-2056-6/13/08 ...$15.00.

1. INTRODUCTION

In recent years, profound changes have occurred in datacenter networking that are likely to impact the performance of latency-sensitive workloads, collectively referred to as online and data-intensive [38]. Particularly relevant are the rise of Overlay Virtual Networking (OVN) – remarkable application of Software-Defined Networking (SDN) – and, simultaneously, the shift to lossless layer-2 fabrics based on Converged Enhanced Ethernet (CEE) or InfiniBand. So far, the trends in virtualization and the commoditization of high-performance-computing-like lossless[1] fabrics have been decoupled, each making independent inroads into the datacenter.

While the research community increasingly focuses on the performance of horizontally-distributed data-intensive applications [15, 8, 9, 25, 38, 41, 42], and recently also on virtualization overlays for multitenant datacenters [28, 40, 17], we argue that the combination of virtualization and such workloads merits closer scrutiny [13]. Our main objective is to analyze the impact of the absence versus presence of flow control on workload performance in a virtualized network. As our study specifically focuses on latency-sensitive, data-intensive workloads, the performance metric of interest is flow completion time (FCT) [20]. As a representative workload model, we selected Partition-Aggregate [8, 41].

1.1 Network Virtualization

As server virtualization allows dynamic and automatic creation, deletion, and migration of virtual machines (VMs), the datacenter network must support these functions without imposing restrictions, such as IP subnet and state requirements. In addition to VM mobility and ease of management, complete traffic isolation is desirable for improved security, which can be achieved by layer-2 and -3 virtualization. Rather than treating the virtual network as a dumb extension of the physical network, these requirements can be effectively met by creating SDN-based overlays such as VXLAN [26] and DOVE [17]. An exemplary architectural exposition of modern virtual overlays is NetLord [28], which covers the key motivations and design principles.

SDN as a concept decouples the control and data planes, introducing programmability and presenting applications with

[1]In this paper we use *lossless* and *zero-loss* in the sense of avoiding packet drops due to congestion. Packets might still be discarded because of CRC errors in the physical links. These, however, are extremely rare events under normal conditions (typical bit error rates are 10^{-12} or less) and recovered by TCP.

an abstraction of the underlying physical network. Scalable and flexible "soft" networks can thus be designed to adapt to changing workloads and to datacenter tenants and operators needs. In a nutshell, SDN trades some degree of performance to simplify network control and management, to automate virtualization services, and to provide a platform upon which new network functionalities can be built. In doing so, it leverages both the OpenFlow [27, 29] and the IETF network virtualization overlay [37, 26] standards.

Based on the adoption rate of virtualization in datacenters, the underlying assumption is that virtual networks (VN) will be deployed in practically most, if not all, multitenant datacenters, providing a fully virtualized Cloud platform by default. For the remainder of this paper, we presume that VN overlay is an intrinsic part of the extended datacenter network infrastructure. Thus we envision a fully virtualized datacenter in which "bare-metal" workloads become the exception, even for mission-critical applications.

However, current hypervisors, virtual switches (vSwitches) and virtual network interface cards (vNICs) critically differ from their modern physical counterparts. In fact, they have a propensity to liberally drop packets even under minor congestive transients. These losses can be considerable and non-deterministic, as will be presented in Section 2.3. Consequently, current non-flow-controlled virtual networks will significantly cancel out the investments of upgrading datacenter networks with flow-controlled CEE and InfiniBand fabrics. We argue that this lossy legacy unnecessarily hinders both the application performance and the progress of future datacenters.

1.2 Lossless Fabrics

The recent standardization of 802 Data Center Bridging for 10-100 Gbps CEE triggered the commoditization of high-performance lossless fabrics. First generation 10G products are already on the market, and CEE fabrics at 40G, or even 100G, have been announced by several vendors.

Traditionally, Ethernet did not guarantee losslessness: packets were dropped whenever a buffer reached its maximum capacity. This behavior does not match the modern semantics of datacenter applications, including High-Performance Computing (HPC) environments [19], storage (Fibre Channel over Ethernet [5]), or Remote Direct Memory Access (RDMA) over Ethernet [16].

CEE upgrades Ethernet with two new mechanisms of interest here: A link-level flow control, i.e., Priority Flow Control (PFC) [6], and an end-to-end congestion management known as Quantized Congestion Notification (QCN). PFC divides the controlled traffic into eight priority classes based on the 802.1p Class of Service field. Within each priority PFC acts as the prior 802.3x PAUSE, except that a paused priority will not affect the others. Hence, a 10-100G link is not fully stopped whenever a particularly aggressive flow exceeds its allotted buffer share. Despite the marked improvement over the original PAUSE, a side-effect of PFC still remains the potential global throughput collapse, which differs from the lossy case. The buffer of a flow-controlled blocked receiver may recursively block buffers upstream, spreading the initial congestion into a saturation tree [30]. To address these head-of-line blocking issues, QCN was defined and extensively simulated prior to releasing PFC.

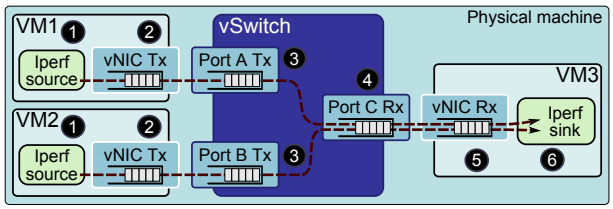


Figure 1: Experimental setup for virtual network loss measurements.

1.3 Contributions

Our main contributions are as follows. (i) We identify and characterize the problem of packet drops in virtual networks. (ii) We implement the first zero-loss Overlay Virtual Network (zOVN) to address the lossless assumption of converged multitenant datacenters. (iii) We quantitatively verify how zOVN improves the *standard* TCP performance for data-intensive applications. Testing Partition-Aggregate on top of zOVN, we achieved up to 15-fold reductions in flow completion times using two distinct testbeds with 1G and 10G Ethernet respectively, and three standard TCPs. Finally, (iv) we investigate the scalability of zOVN by means of accurate full system cross-layer simulations.

The remainder of the paper is structured as follows. In Section 2 we present the main issues of current virtual networks. In Section 3 we explore the design space of virtual overlays. We provide the details of our zOVN prototype in Section 4 and present its evaluation in Section 5. We discuss the results in Section 6 and the related work in Section 7. We conclude in Section 8.

2. VIRTUAL NETWORKS CHALLENGES

The two deficiencies of current virtual networks are latency penalties and excessive packet dropping.

2.1 Latency

A virtual link does not present a well-defined channel capacity. Neither arrivals nor departures can be strictly bounded. The virtual link service time remains a stochastic process depending on the processor design, kernel interrupts, and process scheduling. This negatively affects jitter, burstiness, and quality-of-service. Hence, virtual networks without dedicated real-time CPU support remain a hard networking problem. In addition, virtual networks introduce new protocols spanning layer-2 to 4 and touch every flow or, in extreme cases, even every packet [28, 17]. The result is a heavier stack, with encapsulation-induced delays and overheads possibly leading to fragmentation and inefficient offload processing.

However, the more critical performance aspect is the impact on latency-sensitive datacenter applications. Latency and flow-completion time have been recently established as crucial for horizontally-distributed workloads such as Partition - Aggregate, typically classified as soft real-time. The 200ms end-user deadline [8, 41, 24] translates into constraints of few 10s of milliseconds for the lower-level workers. Although the additional VN-induced delay may be negligible in a basic ping test [28], its impact on more realistic Partition-Aggregate workloads can lead to an increase in the mean flow completion time of up to 82% [13]. This raises concerns about potentially unacceptable VN performance degrada-

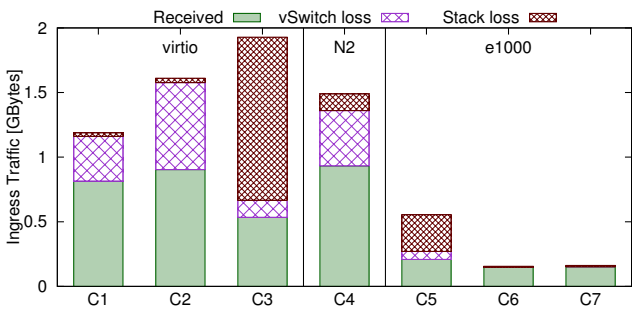


Figure 2: Causes of packet losses. Configurations C1-C7 defined in Table 1.

tions for such critical latency-sensitive applications in a virtualized multitenant environment.

2.2 Losslessness

Ideally a VN should preserve the lossless abstraction assumed by *converged* datacenter applications such as Fibre Channel over Ethernet [5], RDMA over Ethernet [16] or HPC environments [19]. Yet currently *all* the commercial and open-source VNs that we have tested are *lossy*. As losslessness is a critical qualitative feature for the future of converged datacenter networking, CEE spared no effort to ensure zero-loss operation by using two complementary flow and congestion control protocols, namely, PFC and QCN. The same holds for InfiniBand, with its link level credit-based flow control and its FECN/BECN-based end-to-end Congestion Control Annex. In comparison, despite the possibility of relatively simpler and lower-cost flow control implementations, current VNs still resort to packet drop during congestion. This not only degrades datacenter performance, but also fails to correctly terminate modern flow-controlled fabrics, canceling out the investments in a lossless physical network. As an alternative, we demonstrate how a zero-loss Overlay Virtual Network (zOVN) can meet both the desired losslessness *and* the performance requirements.

2.3 Loss measurements

To support the above claims, we assess the extent of packet drops using commonly available virtualization solutions. We perform the experiment shown in Figure 1 in which VM1 and VM2 act as sources and send their traffic towards VM3, which acts as sink, creating a common congestion scenario. We evaluate (i) where and how frequently losses occur, and (ii) the maximum bandwidth that a virtual switch can sustain without dropping packets.

We considered the combinations of hypervisors, vNICs, and virtual switches shown in Table 1. Qemu/KVM is an open-source hypervisor, whereas H2 is a commercial x86 hypervisor. They were used with two types of vNICs: virtio [34] and N2 are virtualization optimized vNICs designed for Qemu and H2, respectively, whereas e1000 fully emulates the common Intel[2] e1000 adapter. In combination with Qemu, we used three virtual switches: Linux Bridge [2], Open vSwitch [3] and VALE [33]. The first two are stable products used in various production deployments whereas VALE is currently a prototype. The combination Qemu-e1000-VALE was omitted as it was affected by an implementation bug that allows internal queues to grow indefinitely, resulting in substantially diverging results between

	Hypervisor	vNIC	vSwitch
C1	Qemu/KVM	virtio	Linux Bridge
C2	Qemu/KVM	virtio	Open vSwitch
C3	Qemu/KVM	virtio	VALE
C4	H2	N2	S4
C5	H2	e1000	S4
C6	Qemu/KVM	e1000	Linux Bridge
C7	Qemu/KVM	e1000	Open vSwitch

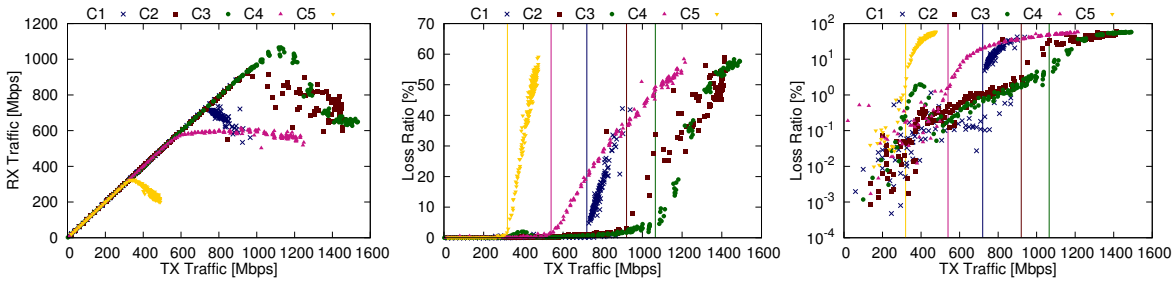
Table 1: Configurations for loss measurements.

runs. With H2 we used its own internal virtual switch S4. All configurations have been tested on a Lenovo T60p Laptop (part of Testbed 1 detailed in Figure 9). Across all experiments, iperf [1] injects 1514B frames of UDP traffic. We determine the losses and bandwidths using the six measurement points shown in Figure 1: (1) and (6) are inside the application itself, (2) and (5) are on the TX and RX side of each vNIC, whereas (3) and (4) are at the virtual switch ports.

Experiment 1: Both generators injected traffic at full speed for 10s, with the last packet being marked. We computed the number of lost packets as the difference between the number of packets transmitted and received at the other end. We investigate (i) *vSwitch losses*, i.e., packets received by the vSwitch input ports (3) and never forwarded to the vSwitch output port (4), and (ii) *receive stack losses*, i.e., packets received by the destination vNIC (5) and never forwarded to the sink (6). The TX path is backpressured up to the vSwitch, hence no losses were observed between other measurement points. A more accurate analysis of the possible loss points is presented in Section 4. With VALE and S4, we could not access the points (3) and (4). Hereby the difference between the sender vNIC and the receiver vNIC counters (points (2) and (5), respectively) was accounted as virtual switch losses. The results are plotted in Figure 2.

Depending on configuration, the total traffic forwarded during the 10s window varied widely. In virtualized networks performance is bounded by the computational resources assigned to each block by the host operating system. Compute-intensive configurations score lower throughputs, inducing less losses in the vSwitch. An example is given by the e1000-based configurations that emulate a fake hardware to "deceive" the guest driver. The virtualization-optimized vNICs – i.e., virtio and N2 – achieved higher rates, thus causing overflows in the virtual switch. The performance optimized VALE switch shifted the bottleneck further along the path, into the destination VM stack. All these results are evidence of the lack of flow control between the virtual network devices, and confirm our initial conjecture.

Experiment 2: To analyze the maximum sustainable bandwidth for the virtual switches, we varied the target injection rate at each generator in increments of 5 Mb/s, starting from 5 Mb/s. The aggregated virtual switch input traffic is the sum, i.e., twice the injection rate. Figure 3a and Figure 3b plot, respectively, the RX rate and loss ratio as a function of the total injection rate. Both were calculated at application level (points (1) and (6)). All configurations exhibit saturation behaviors. The RX rate first increases linearly with the TX rate, up to a saturation peak. Beyond this, with the exception of C4, we observe a drop indicating a lossy congestive collapse, rather than the desired steady saturation plateau. The overloaded system wastes resources to generate more packets, instead of dedicating sufficient re-



(a) vSwitch forwarding performance. (b) Packet loss ratio. (c) Packet loss ratio (log scale).

Figure 3: Experimental loss results. The losses are measured between points 1 and 6 from Figure 1.

sources to the virtual switch and destination VM to actually forward and consume the packets. Although the saturation point varied considerably across configurations, loss rates well in excess of 50% were observed for all configurations (Figure 3b). Even far below the saturation load, marked by vertical lines, we measured losses in the virtual network (Figure 3c) that were significantly above the loss rates expected in its physical counterpart, i.e., up to 10^{-2} instead of 10^{-8} for MTU-sized frames with a typical bit-error rate of 10^{-12}.

The "noise" in Figure 3c confirms our intuitive hypothesis about large non-causal performance variability in virtual networks. In fact, the service rate of each virtual link depends critically on the CPU, load, process scheduling, and the computational intensity of the virtual network code. Suboptimal and load oblivious scheduling causes frequent losses, e.g., by scheduling a sender prior to a backlogged receiver. Lossless virtual switches would be of great interest, not only in terms of efficiency but also for performance predictability. The next sections will present how flow control can be implemented in virtualized datacenter networks.

3. ZOVN DESIGN

In this section we outline the core principles that guided the design of our lossless virtual network.

3.1 Objectives

A converged virtualized network infrastructure must simultaneously satisfy the requirements from the domains being converged. As mentioned above, losslessness is a *functional* requirement of various HPC, storage and IO applications, whereas on-line data-intensive workloads impose *performance* requirements of 200 ms user-level response times.

We base our lossless virtual datacenter stack on CEE-compatible flow control. Transport-wise, we anchor zOVN's design on the established TCP stack combined with lossless overlays as proposed here. Our objectives are :

1) Reconcile the flow completion time application *performance* with datacenter *efficiency* and ease of management. This proves that network virtualization and horizontally-distributed latency-sensitive applications are not mutually exclusive. This may remove an obstacle for virtual network deployment in performance-oriented datacenters.

2) Prove that *commodity* solutions can be adapted for sizable performance gains. As shown in Section 5, a 15-fold flow completion time reduction is also attainable without a clean-slate deconstruction of the existing fabrics and stacks. One can achieve *comparable* performance *gains* with CEE fabrics and standard TCP stacks. Considering the total costs of ownership, this evolutionary reconstruction approach is likely to outperform other, possibly technically superior, alternatives in terms of cost/performance ratios.

3) Expose packet loss as a costly and avertable singularity for modern datacenters, and, conversely, *losslessness* as a key enabler in multitenant datacenters for both (i) the query and flow completion time performance of horizontally-distributed latency-sensitive workloads, and (ii) the convergence of loss-sensitive storage and HPC applications. This basic HPC principle has already been proved by decades of experiences in large-scale deployments. As faster InfiniBand and CEE fabrics are widely available at decreasing prices, datacenters could also now benefit from prior HPC investments in lossless networks.

4) Design and implement a proof-of-concept zero-loss virtual network prototype to experimentally validate the above design principles in a controllable hardware and software environment.

5) Finally, extend and validate at scale the experimental prototype with a detailed cross-layer simulation model.

3.2 End-to-end Argument

The wide availability of lossless fabrics and the thrust of SDN/OpenFlow have prompted us to reconsider the end-to-end and "dumb network" arguments in the context of datacenters. The end-to-end principle [35] can be traced back to the inception of packet networks [12]. Briefly stated, application-specific functions are better implemented in the end nodes than in the intermediate nodes: for example, error detection and correction should reside in NICs and operating system stacks and not in switches and routers. While one of the most enduring design principles, this can also restrict the system level performance in end-to-end delay, flow completion time and throughput [14].

In datacenters, the delay of latency-sensitive flows is impacted not only by network congestion, but also by the end-node protocol stacks [32]. Historically, for low-latency communications, both Arpanet and Internet adopted "raw" transports—unreliable, yet light and fast—instead of TCP-like stacks. Similarly, InfiniBand employs an Unreliable Datagram protocol for faster and more scalable "light" communications. Also HPC protocols have traditionally used low-latency end-node stacks based on the assumption of a lossless network with very low bit-error rates. Given the increasing relevance of latency-sensitive datacenter applica-

tions, current solutions [8, 9, 41, 38] adopted an intriguing option: decouple flow control from the fabric. Here we show that coupling flow control with the fabric positively impacts the workload performance.

3.3 Overlay Virtual Network Design Space

The simplest virtual network would start with a large flat layer-2 network for each tenant. However, this approach does not scale within the practical constraints of current datacenter network technologies. The increasing number of VMs has led to a MAC address explosion, whereby switches need increasingly larger forwarding tables. Also, dynamic VM management stresses the broadcast domains [28]. Moreover, today's limit of 4K Ethernet VLANs is insufficient for multitenant datacenters unless Q-in-Q/MAC-in-MAC encapsulation is used. Finally, the datacenter network must support dynamic and automatic provisioning and migration of VMs and virtual disks without layer-2 or -3 addressing constraints. The emerging solution to full network virtualization are the overlay virtual networks. A number of overlays have recently been proposed [21, 37, 26, 28, 11, 17]. Their key architectural abstraction lies in the separation of virtual networking from the underlying physical infrastructure. Overlays enable an arbitrary deployment of VMs within a datacenter, independent of the underlying layout and configuration of the physical network, without changing or reconfiguring the existing hardware.

Current overlays are predominantly built using layer-2 to -4 encapsulation in UDP, whereby the virtual switches intercept the VM traffic, perform the en-/de-capsulation, and tunnel the traffic over the physical network. Each VM has an associated network state residing in the adjacent switch. Upon VM migration, the virtual switches update their forwarding tables to reflect the new location. Using encapsulation over IP [28, 26, 11, 17], the VM locations are neither limited by the layer-2 broadcast domains, nor by VLAN exhaustion. Instead, full IP functionality is preserved, including QoS and load balancing. Furthermore overlays are independent of location, domains and the physical networking capabilities. Thus these virtual switches are similar to traditional hypervisor switches, but now with additional functionality as overlay nodes. Inherently an overlay network trades some of the bare-metal performance for manageability, flexibility and security.

Performance-wise, such overlays influence datacenter's efficiency and scalability. First, on the data plane: they use encapsulation to build tunnels between virtual switches. Current encapsulation solutions, such as VXLAN [26] and NVGRE [37], solve the original VLAN limitation while reducing the configuration and management overhead. Second, on the management plane: configuration, distribution, and learning protocols are necessary to create tunnels at each virtual switch. To create a tunnel, the overlay switch must map the destination address to its physical location using either the learning or the centralized approach. The *learning* approach, used by VXLAN [26], floods packets with unknown destinations. The *centralized* approach relies on the virtual switches to retrieve the information required for encapsulation. In NetLord [28], this information is learnt by the switches as they communicate with each other and from a central configuration repository. In DOVE [11, 17], this configuration information is retrieved from a centralized database. Both the central configuration repository

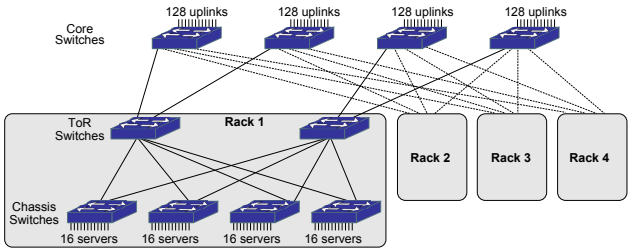


Figure 4: Flat layer-2 fabric with 256 servers.

in NetLord and the centralized database in DOVE must be highly available and persistent. This poses a challenge for multi-million node datacenters, thus indicating a future third option of a distributed repository approach, presuming the entailing coherency issues can be solved efficiently. For now, the learning and centralized approaches are simpler to design and manage. Notably, the centralized method also inherently prevents flooding, the main drawback of the learning approach. For zOVN we have adopted and extended DOVE's centralized approach with a custom encapsulation header.

4. ZOVN IMPLEMENTATION

In this section we describe the details of the implementation of our proposed lossless overlay network (zOVN). We assume a collection of virtualized servers, each running a set of virtual machines. The servers are interconnected through a flat layer-2 fabric (an example is shown in Figure 4). The physical network has per-priority flow control, allowing the network administrator to configure one or more priorities as lossless. The physical per-priority flow control is extended into the virtual domain by our proposed zOVN hypervisor software.

Without loss of generality, to simplify the description, we assume that a single lossless priority is used. In a real setup, different priority classes can be configured to segregate loss tolerant traffic, from mission-critical latency-sensitive traffic that benefits from losslessness, as shown in the next sections.

4.1 Path of a Packet in zOVN

The data packets travel between processes (applications) running inside VMs. Along the way, packets are moved from queue to queue within different software and hardware components. Here we describe the details of this queuing system, with emphasis on the flow control mechanism between each queue pair. The packet path trace is shown in Figure 5.

After processing the packets in the VM's guest kernel, they are transferred to the hypervisor through a vNIC. The hypervisor forwards them to the virtual switch, which provides the communication between VMs and the physical adapter. Packets destined to remote VMs are taken over by a bridge with OVN tunneling functionality that encapsulates and moves them into the physical adapter queues. After traversing the physical network, they are delivered to the destination server, where they are received by the remote bridge, which terminates the OVN tunnel by decapsulating and moving them into the destination's virtual switch input queues. The virtual switch forwards the decapsulated packets to the local hypervisor, which in turn forwards them to the guest OS. After processing in the guest kernel, the

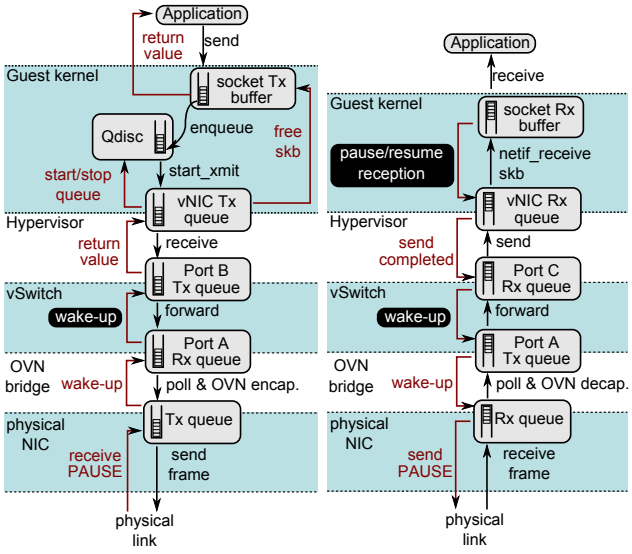


Figure 5: Life of a packet in a virtualized network.

received packets are eventually delivered to the destination application. Based on a careful analysis of the end-to-end path, we identified and fixed the points of potential loss, labeled in white on black in Figure 5, i.e., the vSwitch and the reception path in the guest kernel.

4.1.1 *Transmission Path*

On the transmit side, packets are generated by the user-space processes. As shown in Figure 5, the process issues a `send` system call that copies a packet from user space into the guest kernel space. Next, packets are stored in an `sk_buff` data structure and enqueued in the transmit (TX) buffer of the socket opened by the application. The application knows whether the TX buffer is full from the return value of the system call, making this a lossless operation.

Packets from the socket TX buffer are enqueued in the *Qdisc* associated with the virtual interface. The Qdisc stores a list of pointers to the packets belonging to each socket. These pointers are sorted according to the selected discipline, FIFO by default. To avoid losses at this step, we increase the length of the Qdisc to match the sum of all socket TX queues. This change requires negligible extra memory. The Qdisc tries to send the packets by enqueuing them into the adapter TX queue. If the TX queue reaches a threshold – typically one MTU below maximum – the Qdisc is stopped and the transmission is paused, thus avoiding losses on the TX path of the kernel. When the TX queue drops below the threshold, the Qdisc is restarted and new packets can be enqueued in the TX queue of the virtual adapter. Hence, the transmission path in the guest OS remains lossless as long as the Qdisc length is properly sized.

Our architecture is based on virtio technology [34], hence the virtual adapter queues are shared between the guest kernel and the underlying hypervisor software running in the host user space. The virtio adapter informs the hypervisor when new packets are enqueued in the TX queue. The hypervisor software is based on Qemu [4] and is responsible for dequeuing packets from the TX queue of the virtual adapter and copying them to the TX queue of the zOVN virtual switch.

The Qemu networking code contains two components: virtual network devices and network backends. We use the virtio network device coupled to a Netmap [32] backend. We took the Netmap backend code of the VALE [33] virtual switch and ported it to the latest version of Qemu with the necessary bug fixes, mainly related to concurrent access to the Netmap rings. We use a lossless coupling between the device and the backend, avoiding – via configuration flags – the lossy Qemu VLANs. Packets arrive at the vSwitch TX queue of the port to which the VM is attached. The vSwitch forwards packets from the TX queues of the input ports to the RX queues of the output ports using a forwarding (FIB) table that contains only the MAC addresses of the locally connected VMs. If the destination is found to be local, the respective packets are moved to the corresponding RX queue; else they are enqueued in the RX port corresponding to the physical interface. From here, packets are consumed by a bridge that encapsulates and enqueues them in the TX queue of the physical adapter. Then the lossless CEE physical network delivers the packets to the destination server's physical RX queue.

As shown in Section 2.3, none of the current virtual switches implement flow control, as also confirmed by our discussions with some of the virtualization vendors. Therefore we have redesigned the VALE vSwitch to add internal flow control and to make the TX path fully lossless, as described in Section 4.2.

4.1.2 *Reception Path*

The incoming packets are consumed and decapsulated by the OVN tunneling bridge from the RX queue of the physical NIC. Next, they are enqueued in the TX queue of the virtual switch that forwards them to the RX queue corresponding to the destination VM. This forwarding is again lossless, see Section 4.2. The packets are consumed by the Qemu hypervisor, which copies them into the virtio virtual device. The virtual device RX queue is shared between the hypervisor and the guest kernel. The hypervisor notifies the guest when a packet has been received and the guest OS receives an interrupt. This interrupt is handled according to the Linux[2] NAPI framework. A *softirq* is raised, which triggers packet consumption from the RX queue. The packet is transferred to the `netif_receive_skb` function that performs IP routing and filtering. If the packet is destined to the local stack, it is enqueued in the destination socket RX buffer based on the port number. If the destination socket is full, then the packet is discarded. With TCP sockets this should never happen because TCP has end-to-end flow control that limits the number of injected packets to the advertised window of the receiver. UDP sockets, however, require additional care. We modified the Linux kernel such that when the destination socket RX queue occupancy reaches a threshold – i.e., one MTU below maximum – the softirq is canceled and reception is paused. Once the process consumes data from the socket, reception is resumed. This ensures full lossless operation for both TCP and UDP sockets.

4.2 zVALE: Lossless virtual Switch

As stated before, our lossless vSwitch is derived from VALE [33], which is based on the Netmap architecture [32]. It has one port for each active VM, plus one additional port for the physical interface. Each port has an input (TX) queue for the packets produced by the VMs or received from

Algorithm 1 Lossless vSwitch Operation.

- **Sender** (I_j)
  while *true* **do**
    Produce packet P
    if *Input queue* I_j *full* **then**
      Sleep
    else
      $I_j.enqueue(P)$
      start $Forwarder(I_j)$
    end if
  end while
- **Receiver** (O_k)
  while *true* **do**
    if *Output queue* O_k *empty* **then**
      for all *Input queue* I_j **do**
        start $Forwarder(I_j)$
      end for
    end if
    if *Output queue* O_k *empty* **then**
      Sleep
    else
      $P \leftarrow O_k.dequeue()$
      consume packet P
    end if
  end while
- **Forwarder** (I_j)
  for all *packet P in input queue* I_j **do**
    output port $k \leftarrow fwd\ table\ lookup(P.dstMAC)$
    if not *Output queue* O_k *full* **then**
      $I_j.remove(P)$
      $O_k.enqueue(P)$
      wake-up *receiver* (O_k) **and** *sender* (I_j)
    end if
  end for

the physical link, and an output (RX) queue for the packets to be consumed by VMs or sent out over the physical link. The lossy state-of-the-art implementation forwards packets from input to output queues as fast as they arrive. If an output queue is full, packets are locally discarded.

To make such a software switch lossless, we designed and implemented the pseudocode shown in Algorithm 1. Each sender (producer) is connected to an input queue I_j, and each receiver (consumer) is connected to an output queue O_k. After a packet has been produced, the sender checks whether the associated input queue is full. If the queue is full, the sender goes to *sleep* until a free buffer becomes available, else the sender enqueues the packet in the input queue and then starts a forwarding process to try to *push* packets from the input to the output queues. The forwarder checks each output queue for available space. If a queue has room, the forwarder transfers the packets to the output queue and *wakes* up the corresponding consumers that might be waiting for new packets. On the receiver side, the associated output queue is checked; if not empty, a packet is consumed from it, else the forwarding process is started to *pull* packets from the input queues to this output queue. If data is actually pulled, it is consumed; else the receiver sleeps until woken up by the sender.

The vSwitch is designed to operate in a dual push/pull mode. When the sender is faster (than the receiver), it will sleep most of the time waiting for free buffers, while the receiver will wake it up only when it consumes data. When the receiver is faster (than the sender), it will sleep most of the time, while the sender will wake it up only when new data becomes available. The overhead of lossless operation is thus reduced to a minimum.

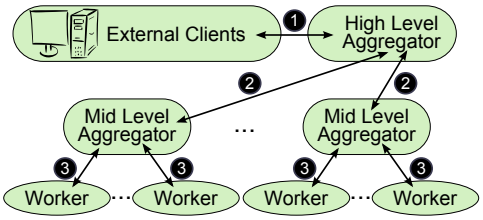


Figure 6: Partition-Aggregate (PA) application.

5. EVALUATION

In this section we evaluate our proposed lossless vSwitch architecture, applying the Partition-Aggregate (PA) workload described in Section 5.1. We run this workload both in two lab-scale experiments with 32 VMs and in a larger-scale simulation using an OMNeT++ model of a 256-server network.

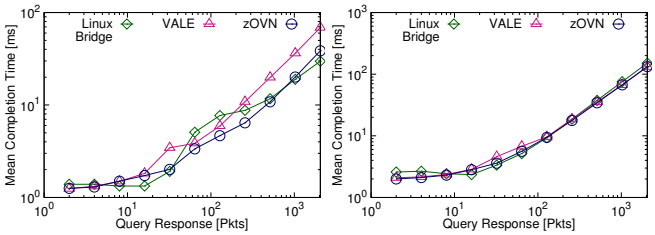
5.1 Partition-Aggregate Workload

A generic 3-tier PA application is presented in [8, 41] and illustrated in Figure 6. At the top tier, a high-level aggregator (HLA) receives HTTP queries from external clients (1). Upon reception of such a request, the HLA contacts randomly selected Mid-Level Aggregators (MLA) and sends them a subquery (2). The MLAs further split the subquery across their workers, one in each server in the same chassis (3). Eventually, each worker replies to the MLA by returning a response. The MLA collects the partial results from workers. When *all* results have been received, the MLA sends its aggregated response to the HLA. The query is completed when the HLA receives the aggregated response from each MLA. The key metric of interest is the flow (or query) completion time, measured from arrival of the external query until query completion at the HLA. In the prototype experiments, similar with the experiments described in [8, 41], we use a reduced two-tier PA workload, in which the MLAs have been omitted, and the HLAs contact the workers directly. In the simulations, on the other hand, we use the full configuration. In both cases, the flows are sent over TCP. The connections between the various components are kept open during the runs to allow TCP to find the optimal congestion window sizes and to avoid slow start.

5.2 Microbenchmarks

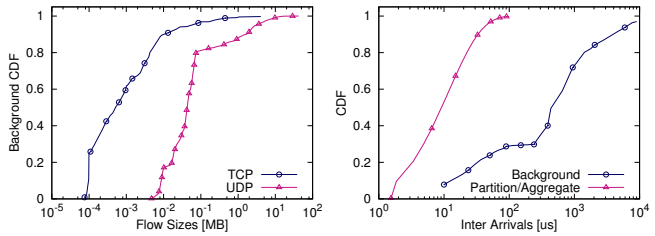
First, we deployed our prototype implementation on two Lenovo M91p-7034 desktops (Intel i5-2400 @ 3.10GHz CPU, 8GB memory, Linux 3.0.3 64-bit kernel both for host and guests). The machines were connected through a 1 Gbps 3com 3GSU05 consumer-level Ethernet switch supporting IEEE 802.3x. The host kernel was patched with the Netmap [32] extensions and our zOVN switch and bridge. The guest kernel was patched with our lossless UDP socket extension.

We ran PA queries with a single aggregator and five workers. In Figure 7 we report the mean query completion time. In Figure 7a the aggregators and the workers resided in VMs on the same server (1-server setup), whereas in Figure 7b the aggregator was on a different server than the workers (2-server setup). We varied the size of the workers response



(a) 1 server (b) 2 servers

Figure 7: Microbenchmarks: 6 VMs PA.



(a) Background flow size. (b) Inter-arrival times.

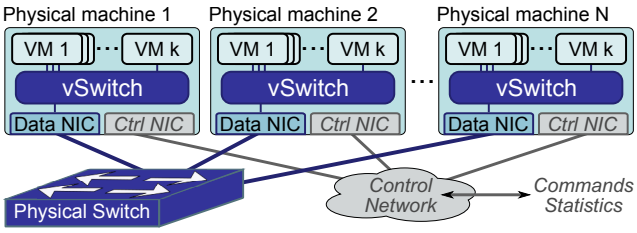
Figure 8: Flow size and inter-arrival distribution.

to the aggregator from 2 to 2048 MTUs. To achieve statistical confidence, each run consisted of 10K repetitions. We compared the Linux Bridge [2] with the lossy VALE implementation [33] and our proposed lossless zOVN. On the 2-server setup, the Netmap-based solutions outperformed the Linux Bridge, but only for small response sizes (up to 30% for 2 MTUs). For medium-sized flows, the Linux Bridge was better (e.g., 8% performance degradation for 64 MTUs when using zOVN). For large response sizes, the three implementations exhibited similar response times. The physical link has a constant service rate, so that TCP was able to find the proper congestion window to avoid most losses. On the desktop machines, the vSwitch could support up to 1.45 Gbps of traffic without losses, compared with the 256 Mbps for the laptop machines. However, the maximum bandwidth through the vSwitch was limited to the 1 Gbps of the physical link, which was the bottleneck in this case. Accordingly, we measured loss ratios of less than 0.02%. Enabling losslessness on such a configuration brings no additional benefits. However, this result validates the efficiency of our implementation.

In the 1-server setup, the zOVN switch was consistently better than the lossy VALE switch across all runs. The Linux Bridge exhibited performance variabilities (up to +19% improvement for the 16 MTU responses over zOVN, but as much as –65% degradation over zOVN for 128 MTU responses). The architecture of the Linux Bridge requires one extra copy for each packet sent or received. This extra overhead slows down the workers reducing the pressure on the vSwitch, thereby reducing packet losses. In the 2-server scenario, the extra overhead was hidden by the physical link bottleneck.

5.3 Lab-Scale Experiments

Next, we deployed zOVN over the two testbeds described in Figure 9. We ran a PA workload using 32 VMs with the



	Testbed 1	Testbed 2
System Type	Lenovo T60p Laptops	IBM System x3550 M4 Rack Servers
CPU	1x Intel Core 2 T7600	2x Intel Xeon E5-2690
Total cores	2	16
Clock speed [GHz]	2.33	2.90
Memory [GB]	3	96
Physical machines	8	4
VMs/machine	4	16
Data network	1G Ethernet	10G CEE
Physical switch	HP 1810-8G	IBM RackSwitch G8264
Control network	wireless	1G wired
Linux kernel	3.0.3 64-bit	3.0.3 64-bit

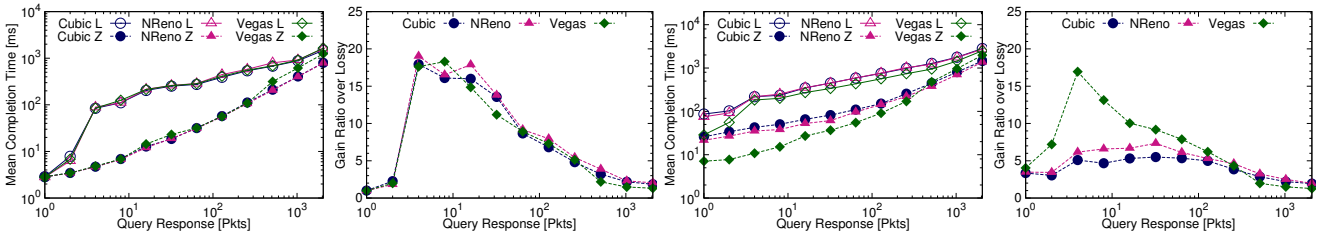
Figure 9: Real implementation testbeds.

same methodology and flow sizes as in the previous paragraph. In addition, we varied the TCP version between NewReno, Vegas and Cubic. As shown in Figure 9, each physical machine has two network interfaces. The PA traffic that is subject to measurements flows through an isolated data network. The workers, aggregators and background traffic generators are started and killed through a separate control network, which is also used to configure the data network before each run and to gather the statistics at the end without interfering with the experiments.

Testbed 1: Laptops. In Figure 10a and 10b, we report the mean completion time and performance gain of zero-loss (Z) over lossy (L). The zero-loss configuration has flow control enabled both in the physical and the virtual network, whereas the lossy configuration has no flow control in any of the two networks. The mean flow completion time was reduced by a factor of up to 19.1×. The highest benefit was achieved for flow sizes between 6 KB and 48 KB (4 and 32 packets). For very small flows, the total size of all worker responses was too small to cause any buffer overflow. For long flows, the losses were recovered through fast-retransmit and selective acknowledgments. All TCP versions performed about equally.

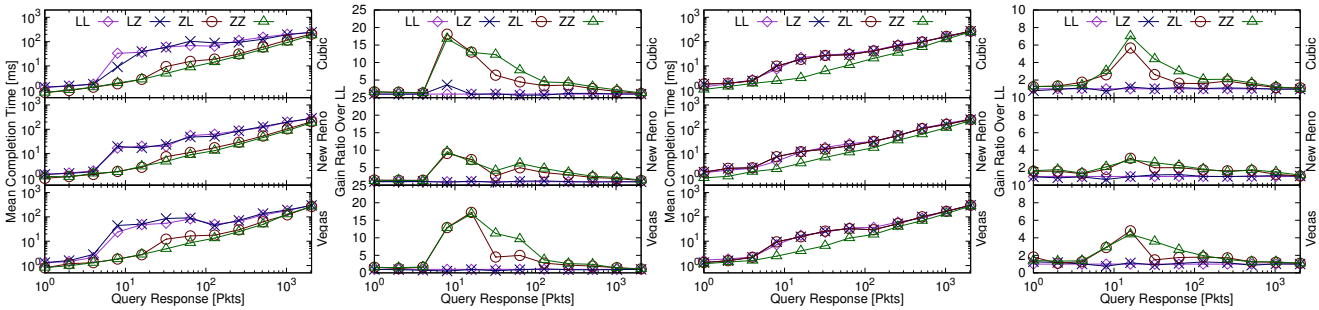
In Figure 10c and 10d, we report the same metrics, but with background traffic. In this scenario, each VM hosts an additional traffic generator producing background flows. The generator chooses a random uniformly distributed destination, then it sends to it a TCP flow with the length drawn from the distribution in Figure 8a. Afterward, the generator sleeps according to the background flow inter-arrival distribution shown in Figure 8b. Both the PA and the background flows use the same TCP version. The gain is smaller than in the previous scenario, because the background flows also benefit from losslessness obtaining a higher throughput. In particular, the congestion window of NewReno and Cubic are kept open due to the absence of losses. On the other hand, the latency sensitive Vegas injects background traffic at a lower rate, thus the completion times are shorter.

Testbed 2: Rack Servers. We repeat the above experiments on 4 rack servers with a 10G CEE network. Each



(a) w/o bkgd. (b) w/o bkgd. (c) with bkgd. (d) with bkgd.

Figure 10: Testbed 1 results: 32 VMs PA running on 8 laptops.



(a) w/o bkgd. (b) w/o bkgd. (c) with bkgd. (d) with bkgd.

Figure 11: Testbed 2 results: 32 VMs PA running on 4 rack servers.

server hosts 16 VMs: 8 for PA traffic and 8 VMs for generating background traffic. We studied four flow control configurations: no flow control (LL), flow control activated in the physical network (LZ), flow control activated in the virtual network (ZL), and flow control activated in both (ZZ). The mean completion times and gains over LL are reported in Figure 11a and 11b. The mean completion times are reduced by a factor up to 15.95×, similar to the laptop experiments. Although the server CPUs have more resources than the laptop CPUs, they have to handle more VMs and more traffic from a 10× faster network. Activating flow control only in the physical network (LZ) showed no major benefit in this scenario, where the primary bottleneck is in the vSwitches. Also, enabling flow control only in the vSwitch (ZL) shifted the drop point from the virtual to the physical domain. Finally, in Figure 11c and 11d, we repeated the experiments with background traffic, confirming the findings from Testbed 1.

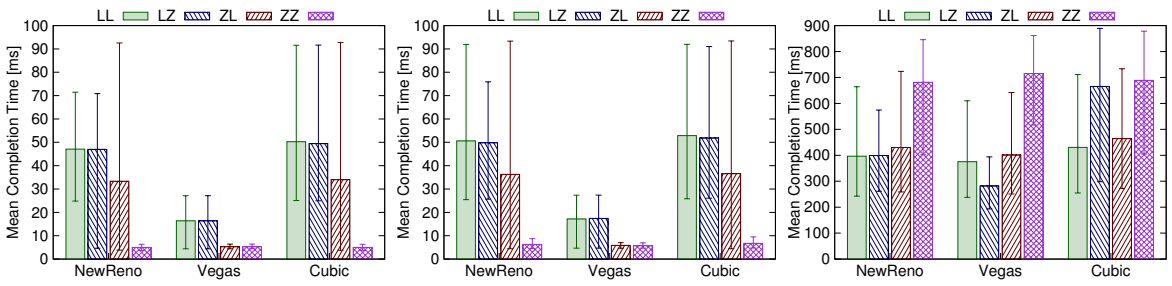
5.4 Simulation Experiments

To finalize our validation, we implemented a model of the zOVN system on top of the OMNeT++ network simulator. The simulator models a 10G CEE fabric at frame level with generic input-buffered output-queued switches. As the TCP models implemented in OMNeT++, as well as those from NS2/3, are highly simplified, we ported the TCP stack from a FreeBSD v9 kernel into this simulator with only minimal changes, most of them related to memory management. As we focus on the network, we did not model the endnode CPUs, assuming that the endnodes can process the segments as fast as they arrive, and that the applications can reply immediately. The stack adds only a *fixed* delay to each segment, calibrated from our prior hardware experiments. Even if idealized, these assumptions are consistent with our network-centric methodology. The simulator also incorporates a thin UDP layer used for background flows performing simple segmentation and encapsulation of the application data.

The zOVN model performs switching and bridging in the same way as in the testbed experiment. However, here we chose a different encapsulation size of 54B, reflecting a VXLAN-type encapsulation: 18B outer Ethernet header + 20B outer IP header + 8B UDP header + 8B VXLAN header. To avoid fragmentation, we decreased the MTU value accordingly from 1500B to 1446B. Modern CEE hardware is able to increase its physical MTUs, thus preserving the default settings.

The simulated network topology is shown in Figure 4. It consists of 256 servers, distributed in 16 chassis, and interconnected through a three-layer fat tree. Clients attached to the up-links inject HTTP queries that are served by the VMs residing on each virtualized server. The queries were generated according to the inter-arrival times shown in Figure 8b. Each server hosts 3 VMs, one HLA, one MLA and one worker. The client query reaches a randomly chosen HLA that in turns chooses 16 MLAs, one in each chassis. Each MLA contacts all worker VMs from the same chassis. The messages exchanged between the HLA, MLAs and workers have a fixed size of 20KB.

Figure 12 compares the mean completion times and the 5- and 95-percentiles for different flow control configurations under no, light, and heavy background traffic. We studied the four flow control configurations introduced above (LL, LZ, ZL, and ZZ) and the same three TCP versions as before. Enabling flow control in only one network (either physical or virtual) is not beneficial, because packet losses



(a) w/o bkgd (b) TCP bkgd (c) UDP bkgd

Figure 12: Simulation results: 768 VMs PA with 256 servers.

are merely shifted from one domain to the other. However, the effects were not altogether identical, because the virtual flow control still benefited inter-VM communications on the same host. Therefore, enabling only the virtual flow control (ZL) still led to a performance improvement, although smaller than in the ZZ case. Enabling both flow controls (ZZ) achieved significant gains, similar to those observed in the testbed: a reduction in FCT of up to 10.1× with Cubic, and no background flows. When adding light background traffic, we observed similar gain decreases. However, a new insight is that in the presence of heavy UDP background traffic, enabling flow control will harm performance. In this case, the uncooperative background UDP packets did no longer get dropped and, consequently, hogged link capacity and harmed the foreground PA workload traffic. These results confirmed the need to segregate the traffic into PFC priorities with true resource separation and scheduling. It may also suggest the need for a layer-2 congestion management loop as in [18].

With background traffic, Vegas outperformed NewReno and Cubic, confirming the results obtained on the testbed setups. In the case without background traffic Vegas was again better. Nonetheless, on the testbeds, all TCP versions produced similar results. The difference here is due to the more complex communication pattern with more hops, as more flows share the same path. This causes longer queues, especially in the core switches. The longer delays are detected by Vegas, which will reduce its congestion window, thus obtaining shorter completion times.

6. DISCUSSION

Here we review the main takeaways from the results presented in this paper. Using zOVN's experimental platform, we demonstrated both absence of packet drops – in support of converged storage and HPC applications – *and* improved flow completion time (FCT) performance. Thus, we have achieved our primary objective of reconciling performance with losslessness for overlay virtual networks.

Is lossless flow control more relevant for physical or for virtual networks? Having tested all four combinations of lossy and lossless physical and virtual flow control both in our testbed and in simulations, we found that contiguous end-to-end flow control, hop-by-hop within each domain, yields the largest reductions in FCT: PA over zOVN with 32 virtual workers distributed across four physical rack servers achieved up to 15-fold peak speedup. Relevant to on-line and data-intensive workloads in general, the highest speedups recorded are for flows between 6 and 50 KB. Unexpectedly, if a suboptimal choice between flow control in either the physical or the virtual network must still be made, the latter is better for FCT performance, as demonstrated by the results for ZL vs. LZ in Figure 12. As noted initially, this situation entails a paradoxical twist: Although CEE and InfiniBand fabrics have already implemented the costlier (buffers, logic, and signaling) hardware flow control, this remains practically non-existent in today's virtual networks - despite much lower implementation efforts.

Are our modest experimental platforms relevant for hundreds of blade-based racks and top-of-rack switches with 40-100 Gbps uplinks? While the definitive answer would entail a multi-million dollar datacenter setup, we are confident in the relevance of our admittedly limited prototype platforms. Thin and embedded low-power CPUs as used in microservers as well as fully virtualized, and hence loaded, "fat" CPUs are likely to exhibit qualitatively similar behaviors as these measured on our two testbeds.

During zOVN experiments we consistently observed how the loss ratio is influenced by the CPU/network speed ratio. On the transmit side, a fast Intel Xeon[2] CPU can easily overload a slower 1G network, producing more losses in the vSwitch than a slower CPU (Intel Core 2) with the same 1G NIC does. On the other hand, on the receive side, a fast 10G network coupled with a loaded Intel Xeon CPU produces more drops than the 1G network with the same CPU does. As TX is network-limited, a fast network is beneficial on the TX side – but hurts performance on the RX side – whereas a fast CPU is beneficial on the RX side – processor-limited – while it hurts the TX side. In conclusion, a different CPU/network speed ratio is not a viable substitute for a correct implementation of flow control in the virtual network.

7. RELATED WORK

In recent years, the TCP incast and flow completion time performance of Partition-Aggregate applications has been extensively analyzed. For example, [15, 39] suggest a 10-1000× retransmission timeout reduction. Other proposals achieve sizable flow completion time reductions for typical datacenter workloads using new single-path [8, 9, 41, 38] or multi-path [42, 24, 36, 7] transports. These are coupled with deadline-aware or agnostic schedulers and per-flow queuing. Related to our work and to [22, 18], DeTail [42] identifies packet loss in physical networks as one of the three main issues. The authors enable flow control, i.e., PFC, and intro-

duce a new multi-path congestion management scheme targeted against flash hotspots typical of Partition-Aggregate workloads. They also employ explicit congestion notification (ECN) against persistent congestion. DeTail uses a modified version of NewReno to reduce flow completion time by 50% at the 99.9-percentile, but does not address virtual overlays.

pFabric [10] re-evaluates the end-to-end argument. It introduces a "deconstructed" light transport stack resident in the end node and re-designed specifically for latency-sensitive datacenter applications. Furthermore, a greedy scheduler implements a deadline-aware global scheduling and a simplified retransmission scheme recovers losses. By replacing both the TCP stack *and* the standard datacenter fabric, this scheme achieves near-ideal performance for short flows. Open issues are the scalability to datacenter-scale port counts, costs of replacing commodity fabrics and TCP version, fairness, and compatibility with the *lossless* converged datacenter applications.

DCTCP [8] uses a modified ECN feedback loop with a multibit feedback estimator filtering the incoming ECN stream. This compensates the stiff active queue management in the congestion point detector with a smooth congestion window reduction function reminiscent of QCN's rate decrease. DCTCP reduces the flow completion time by 29%, however, as a deadline-agnostic TCP it misses about 7% of the deadlines. D3 [41] is a deadline-aware first-come first-reserved non-TCP transport. Its performance comes at the cost of priority inversions for about 33% of the requests [38] and a new protocol stack. PDQ [24] introduces a multi-path preemptive scheduling layer for meeting flow deadlines using a FIFO taildrop similar to D3. By allocating resources to the most critical flows first, PDQ improves on D3, RCP and TCP by circa 30%. As it is not TCP, its fairness remains to be studied. D2TCP [38] improves on D3 and DCTCP, with which it shares common features in the ECN filter, by penalizing the window size with a gamma factor. Thus, it provides iterative feedback to near-deadline flows and prevents congestive collapse. This deadline-aware TCP-friendly proposal yields 75% and 50% fewer deadline misses than DCTCP and D3, respectively. Hedera and MP-TPC [7, 23, 31] propose multi-path TCP versions optimized for load balancing and persistent congestion. However, short flows with fewer than 10 packets or FCT-sensitive applications do not benefit, despite the complexity of introducing new sub-sequence numbers in the multi-path TCP loop.

8. CONCLUDING REMARKS

Fabric-level per-lane flow control to prevent packet loss due to contention and transient congestion has long been the signature feature of high-end networks and HPC interconnects. The recent introduction of CEE priority flow control has now made it a commodity. In spite of the advances at layer-2, we have shown that present virtual overlays lag behind. Congestion, whether inherent in the traffic pattern or as an artifact of transient CPU overloads, is still handled here by dropping packets, thus breaking convergence requirements, degrading performance, and wasting CPU and network resources.

We provided first evidence that, for latency-sensitive virtualized datacenter applications, packet loss is a costly singularity in terms of performance. To remedy this situation, we have identified the origins of packet drops across the entire virtualized communication stack, and then designed and implemented a fully lossless virtual network prototype.

Based on the experimental results using our prototype implementations and also larger-scale simulations, we have demonstrated average FCT improvements of one order of magnitude. Additional takeaways are that (i) packet loss in virtualized datacenters is even costlier than previously studied in physical networking; (ii) FCT performance of Partition-Aggregate workloads is greatly improved by losslessness in the virtualized network; (iii) commodity CEE fabrics and standard TCP stacks still have untapped performance benefits. Furthermore, zOVN can be orthogonally composed with other schemes for functional or performance enhancements on layers 2 to 5.

Acknowledgements

We are deeply indebted to the anonymous reviewers and our shepherd, Amin Vahdat for their helpful feedback; Rami Cohen and Katherine Barabash from IBM[2] HRL for the DOVE-related help; Andreea Anghel for the first OVN loss experiments; Anees Shaikh, Renato Recio, Vijoy Pandey, Vinit Jain, Keshav Kamble, Martin Schmatz and Casimer DeCusatis for their kind support, comments and feedback.

9. REFERENCES

[1] Iperf. URL: `http://iperf.sourceforge.net`.

[2] Linux Bridge. URL: `http://www.linuxfoundation.org/collaborate/workgroups/networking/bridge`.

[3] Open vSwitch. URL: `http://openvswitch.org`.

[4] QEMU-KVM. URL: `http://www.linux-kvm.org`.

[5] Fabric convergence with lossless Ethernet and Fibre Channel over Ethernet (FCoE), 2008. URL: `http://www.bladenetwork.net/userfiles/file/PDFs/WP_Fabric_Convergence.pdf`.

[6] P802.1Qbb/D2.3 - Virtual Bridged Local Area Networks - Amendment: Priority-based Flow Control, 2011. URL: `http://www.ieee802.org/1/pages/802.1bb.html`.

[7] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat. Hedera: Dynamic Flow Scheduling for Data Center Networks. In *Proc. NSDI 2010*, San Jose, CA, April 2010.

[8] M. Alizadeh, A. Greenberg, D. A. Maltz, et al. DCTCP: Efficient Packet Transport for the Commoditized Data Center. In *Proc. ACM SIGCOMM 2010*, New Delhi, India, August 2010.

[9] M. Alizadeh, A. Kabbani, T. Edsall, B. Prabhakar, A. Vahdat, and M. Yasuda. Less is More: Trading a little Bandwidth for Ultra-Low Latency in the Data Center. In *Proc. NSDI 2012*, San Jose, CA, April 2012.

[10] M. Alizadeh, S. Yang, S. Katti, N. McKeown, et al. Deconstructing Datacenter Packet Transport. In *Proc. HotNets 2012*, Redmond, WA.

[2]Intel and Intel Xeon are registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries. IBM is a trademark of International Business Machines Corporation, registered in many jurisdictions worldwide. Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both. Other product or service names may be trademarks or service marks of IBM or other companies.

[11] K. Barabash, R. Cohen, D. Hadas, V. Jain, et al. A Case for Overlays in DCN Virtualization. In *Proc. DCCAVES'11*, San Francisco, CA.

[12] P. Baran. On Distributed Communications Networks. *IEEE Transactions on Communications*, 12(1):1–9, March 1964.

[13] R. Birke, D. Crisan, K. Barabash, A. Levin, C. DeCusatis, C. Minkenberg, and M. Gusat. Partition/Aggregate in Commodity 10G Ethernet Software-Defined Networking. In *Proc. HPSR 2012*, Belgrade, Serbia, June 2012.

[14] M. S. Blumenthal and D. D. Clark. Rethinking the Design of the Internet: The End-to-End Arguments vs. the Brave New World. *ACM Transactions on Internet Technology*, 1(1):70–109, August 2001.

[15] Y. Chen, R. Griffith, J. Liu, R. H. Katz, and A. D. Joseph. Understanding TCP Incast Throughput Collapse in Datacenter Networks. In *Proc. WREN 2009*, Barcelona, Spain, August 2009.

[16] D. Cohen, T. Talpey, A. Kanevsky, et al. Remote Direct Memory Access over the Converged Enhanced Ethernet Fabric: Evaluating the Options. In *Proc. HOTI 2009*, New York, NY, August 2009.

[17] R. Cohen, K. Barabash, B. Rochwerger, L. Schour, D. Crisan, R. Birke, C. Minkenberg, M. Gusat, et al. An Intent-based Approach for Network Virtualization. In *Proc. IFIP/IEEE IM 2013*, Ghent, Belgium.

[18] D. Crisan, A. S. Anghel, R. Birke, C. Minkenberg, and M. Gusat. Short and Fat: TCP Performance in CEE Datacenter Networks. In *Proc. HOTI 2011*, Santa Clara, CA, August 2011.

[19] W. Dally and B. Towles. *Principles and Practices of Interconnection Networks, Chapter 13*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 2003.

[20] N. Dukkipati and N. McKeown. Why Flow-Completion Time is the Right Metric for Congestion Control. *ACM SIGCOMM CCR*, 36(1):59–62, January 2006.

[21] H. Grover, D. Rao, D. Farinacci, and V. Moreno. Overlay Transport Virtualization. Internet draft, IETF, July 2011.

[22] M. Gusat, D. Crisan, C. Minkenberg, and C. DeCusatis. R3C2: Reactive Route and Rate Control for CEE. In *Proc. HOTI 2010*, Mountain View, CA, August 2010.

[23] H. Han, S. Shakkottai, C. V. Hollot, R. Srikant, and D. Towsley. Multi-Path TCP: A Joint Congestion Control and Routing Scheme to Exploit Path Diversity in the Internet. *IEEE/ACM Transactions on Networking*, 14(6):1260–1271, December 2006.

[24] C.-Y. Hong, M. Caesar, and P. B. Godfrey. Finishing Flows Quickly with Preemptive Scheduling. In *Proc. ACM SIGCOMM 2012*, Helsinky, Finland.

[25] S. Kandula, D. Katabi, S. Sinha, and A. Berger. Dynamic Load Balancing Without Packet Reordering. *ACM SIGCOMM Computer Communication Review*, 37(2):53–62, April 2007.

[26] M. Mahalingam, D. Dutt, K. Duda, et al. VXLAN: A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks. Internet draft, IETF, August 2011.

[27] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, et al. OpenFlow: Enabling Innovation in Campus Networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, April 2008.

[28] J. Mudigonda, P. Yalagandula, J. C. Mogul, et al. NetLord: A Scalable Multi-Tenant Network Architecture for Virtualized Datacenters. In *Proc. ACM SIGCOMM 2011*, Toronto, Canada.

[29] B. Pfaff, B. Lantz, B. Heller, C. Barker, et al. OpenFlow Switch Specification Version 1.1.0. Specification, Stanford University, February 2011. URL: `http://www.openflow.org/documents/openflow-spec-v1.1.0.pdf`.

[30] G. Pfister and V. Norton. Hot Spot Contention and Combining in Multistage Interconnection Networks. *IEEE Transactions on Computers*, C-34(10):943–948, October 1985.

[31] C. Raiciu, S. Barre, and C. Pluntke. Improving Datacenter Performance and Robustness with Multipath TCP. In *Proc. ACM SIGCOMM 2011*, Toronto, Canada, August 2011.

[32] L. Rizzo. netmap: A Novel Framework for Fast Packet I/O. In *Proc. USENIX ATC 2012*, Boston, MA.

[33] L. Rizzo and G. Lettieri. VALE, a Switched Ethernet for Virtual Machines. In *Proc. CoNEXT 2012*, Nice, France, December 2012.

[34] R. Russell. virtio: Towards a De-Facto Standard For Virtual I/O Devices. *ACM SIGOPS Operating System Review*, 42(5):95–103, July 2008.

[35] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-End Arguments in System Design. *ACM Transactions on Computer Systems*, 2(4):277–288, November 1984.

[36] M. Scharf and T. Banniza. MCTCP: A Multipath Transport Shim Layer. In *Proc. IEEE GLOBECOM 2011*, Houston, TX, December 2011.

[37] M. Sridharan, K. Duda, I. Ganga, A. Greenberg, et al. NVGRE: Network Virtualization using Generic Routing Encapsulation. Internet draft, IETF, September 2011.

[38] B. Vamanan, J. Hasan, and T. N. Vijaykumar. Deadline-Aware Datacenter TCP (D2TCP). In *Proc. ACM SIGCOMM 2012*, Helsinki, Finland.

[39] V. Vasudevan, A. Phanishayee, H. Shah, E. Krevat, D. G. Andersen, G. R. Ganger, G. A. Gibson, and B. Mueller. Safe and Effective Fine-grained TCP Retransmissions for Datacenter Communication. In *Proc. ACM SIGCOMM 2009*, Barcelona, Spain.

[40] G. Wang and T. S. E. Ng. The Impact of Virtualization on Network Performance of Amazon EC2 Data Center. In *Proc. INFOCOM 2010*, San Diego, CA, March 2010.

[41] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowstron. Better Never than Late: Meeting Deadlines in Datacenter Networks. In *Proc. ACM SIGCOMM 2011*, Toronto, Canada, August 2011.

[42] D. Zats, T. Das, P. Mohan, D. Borthakur, and R. Katz. DeTail: Reducing the Flow Completion Time Tail in Datacenter Networks. In *Proc. ACM SIGCOMM 2012*, Helsinky, Finland, August 2012.