RZ 3840 (# Z1302-037) 03/13/2013
Computer Science 14 pages

Research Report

Got Loss? Get zOVN!

D. Crisan, R. Birke, G. Cressier, C. Minkenberg, and M. Gusat

IBM Research – Zurich
8803 Rüschlikon
Switzerland

LIMITED DISTRIBUTION NOTICE

This report has been submitted for publication outside of IBM and will probably be copyrighted if accepted for publication. It has been issued as a Research Report for early dissemination of its contents. In view of the transfer of copyright to the outside publisher, its distribution outside of IBM prior to publication should be limited to peer communications and specific requests. After outside publication, requests should be filled only by reprints or legally obtained copies (e.g., payment of royalties). Some reports are available at http://domino.watson.ibm.com/library/Cyberdig.nsf/home.

Research
Africa • Almaden • Austin • Australia • Brazil • China • Haifa • India • Ireland • Tokyo • Watson • Zurich

Got Loss? Get zOVN!

ABSTRACT

Datacenter networking is currently dominated by two major trends. One is toward lossless, flat layer-2 fabrics based on Converged Enhanced Ethernet and InfiniBand, with benefits in efficiency and performance. The second trend aims at increasing flexibility by means of Software Defined Networking, which enables Overlay Virtual Networking. Although clearly complementary, these trends also exhibit conflicting traits: In contrast to physical fabrics, which avoid packet drops by means of flow control, almost all current virtual networks are lossy. We quantify these losses for several combinations of hypervisors and virtual switches, and show their detrimental effect on application performance. Moreover, we propose zOVN, a zero-loss Overlay Virtual Network, designed to reduce the 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 a proof of concept, we have implemented a zOVN prototype, which we benchmark with Partition-Aggregate, achieving up to 19-fold reduction of the mean completion time using three widespread TCP versions. For larger scale validation and deeper introspection into zOVN's operation, we developed an OMNeT++ model for accurate cross-layer simulations of a virtualized datacenter. The results obtained through simulation confirm the validity of the testbed findings.

Keywords

Datacenter networking, virtualization, overlay networks, lossless, Partition-Aggregate.

1. INTRODUCTION

Recent years have marked profound changes in datacenter networking that are likely to impact the performance of the latency-sensitive workloads, collectively referred to as On-Line and Data-Intensive (OLDI) [37]. Particularly relevant are the rise of Overlay Virtualized Networking (OVN)—enabled by Software-Defined Networking (SDN)—and, simultaneously, the shift to lossless layer 2 fabrics based on Converged Enhanced Ethernet (CEE) and InfiniBand (IB). So far, the networking trends in virtualization and the commodification 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 OLDI applications [15, 8, 9, 24, 37, 40, 41], and recently also on the new virtualization overlays for multitenant datacenters [27, 39, 16], we argue that the combination of virtualization with such workloads merits closer scrutiny [13, 18]. Our main objective is to analyze the impact on workload performance of the absence vs. presence of flow control in a virtualized datacenter network. As our study specifically focuses on latency-sensitive, data-intensive workloads, the performance metric of interest is flow completion time (FCT) [19]. As a representative OLDI workload model, we selected Partition-Aggregate (PA) [8, 40].

1.1 Network Virtualization

As server and storage virtualization allow for dynamic and automatic creation, deletion, and migration of virtual machines (VMs) and virtual disks, 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, datacenter applications today benefit from complete traffic isolation, which can be achieved by layer-2 and 3 virtualization. Rather than treating the virtual network as a dumbed-down extension of the physical network, these requirements can be effectively met by creating SDN-based overlays such as VXLAN [25] and DOVE [11]. An exemplary architectural exposition of modern OVNs is NetLord [27], which covers the key design principles of SDN-based datacenter overlays.

SDN as a concept decouples the control and data planes, introducing programability and presenting applications with an abstraction of the underlying phys-

[1]In this paper we use *lossless* and *zero-loss* in the sense of being capable to avoid dropping packets due to congestion. Note that packets might still be discarded due to 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.

ical 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 is used to simplify network control and management, automate virtualization services and provide a platform upon which to build new network functionality. In doing so, it leverages both the IETF network virtualization overlays [36, 25] and the OpenFlow [26, 28] standard.

OpenFlow presently embodies the preeminent mechanism to implement an SDN control plane with centralized intelligence. OpenFlow moves the network control plane into software running on a server attached to an OpenFlow-enabled switch or router. The flow of network traffic can then be controlled dynamically without the need to rewire the datacenter for each new tenant.

Based on the adoption rate of virtualization in datacenters, the underlying assumption here is that virtual network (VN) technologies will be deployed in practically most, if not all, multitenant datacenters, providing by default a fully virtualized Cloud platform. For the remainder of this paper we presume the VN overlay as an intrinsic part of the extended datacenter network infrastructure; accordingly, the datacenter traffic, including data, control, and management, is virtualized by default. Therefore, we envision a fully virtualized datacenter where 'bare-metal' OLDI workloads become the exception, even for mission-critical applications.

However, today's VN overlays throw a few unexpected wrenches in the datacenter stack. For one, current hypervisors, virtual switches (vSwitches) and virtual network interface cards (vNICs) critically differ from their modern physical counterparts, because they have a propensity to liberally drop packets even under minor congestive transients. In our experiments these losses may amount to a considerable fraction of the offered traffic. As shown in Table 2, frequently a VN loss episode may appear as non-deterministic to the casual observer. Consequently, current non-flow-controlled virtual networks will to a significant extent cancel the investments in upgrading datacenter networks with carefully flow-controlled CEE and InfiniBand fabrics.

1.2 Lossless Fabrics

The recent standardization of 802 Data Center Bridging (DCB) for 10 Gbps CEE has unleashed the commodification of high-performance lossless fabrics. Whereas the first generation of 'pre-draft' DCB products have already implemented Priority Flow Control (PFC, [6]) in 10GE lossless switches and adapters, currently more than a dozen vendors have announced their second or third generations of 'full' CEE fabrics at 40G, or even 100G. In addition to losslessness, high throughput, and extensive support for consolidated traffic – server, storage, FCoE, RoCEE – and virtualization, new

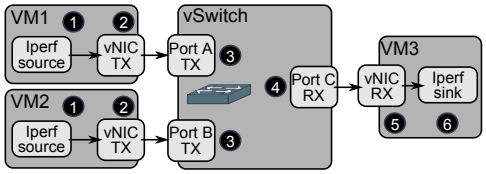


Figure 1: Experimental setup for virtual network loss measurements.

CEE switches use sophisticated scheduling, forwarding, buffering and flow control schemes to reduce the chip traversal latency under a few 100s of ns.

Traditionally, Ethernet did not guarantee lossless communication. Instead, packets were dropped whenever a receive buffer reached its maximum capacity. Reliable upper-layer transports such as TCP interpret such an event as implicit congestion feedback, triggering window or injection rate corrections. This behavior, however, does not match the semantics of today's datacenter applications, including parallel computing (clustering, HPC), storage (Fibre Channel), or RDMA.

CEE substantially upgrades Ethernet's flow control by means of two recently adopted mechanisms: The PFC link-level flow control mechanism, and an end-to-end congestion management scheme referred to as Quantized Congestion Notification (QCN).

PFC divides the controlled traffic into eight priority 'lanes' based on the 802.1p Class of Service field. Within each priority or traffic class, PFC acts as prior 802.3x PAUSE [6], except that a paused priority will not affect the others. Hence, a 100G link is not fully stopped just because a particularly aggressive flow has exceeded its allotted downstream buffer share. Despite the marked improvement with respect to the original PAUSE, a side-effect of PFC remains still the potential global throughput collapse: If a sender is blocked, its buffer may fill up and recursively block switches upstream, spreading the initial hotspot congestion into a saturation tree [29]. To address the head-of-line blocking issues, the DCB task group has first defined its layer-2 congestion control scheme (QCN) before releasing PFC.

1.3 Contributions

Our main contributions are as follows: (i) We identify and make a first attempt to characterize the problem of packet drops in virtual networks. (ii) As a solution, we introduce and implement zOVN, the first lossless virtual network overlay that preserves the zero-loss abstraction required by the converged multitenant datacenter. (iii) Regarding its impact on latency, we quantitatively substantiate how zOVN improves *standard* TCPs' performance for OLDI applications. Thence we perform the first FCT-based *experimental* study of a PA workload running on top of such a virtualization layer, reducing

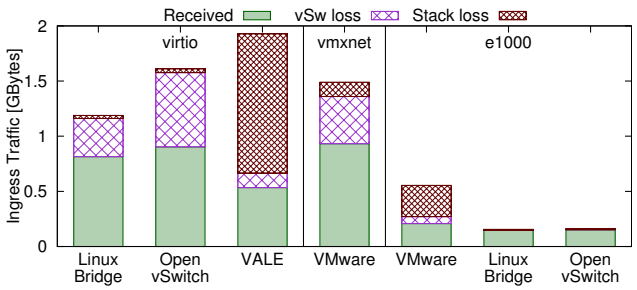


Figure 2: Causes of packet losses.

by up to 19x the FCT of three standard TCPs over commodity Ethernet. Finally, (iv) we investigate the scalability of zOVN by means of accurate full system simulations.

The rest of the paper is structured as follows. In Section 2 we present the main issues of the current virtual networks. In Section 3 we explore the design space of overlay virtual networks. 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

In this section, we summarize and illustrate the two main deficiencies of current virtual network implementations, namely latency penalties and excessive packet dropping.

2.1 Latency

A virtual link does not pose a well-defined channel capacity in bits per second. Neither arrival nor departure processes can be strictly bounded. Instead, the distribution of the virtual link service time remains a stochastic process with temporal dependencies on processor design and kernel interrupt and process scheduling. This negatively affects jitter, burstiness and quality-of-service (QoS). Virtual networks without real time CPU support remain a hard networking problem.

In addition, virtual networks intrinsically introduce new protocols spanning layer-2 to 4, and touching every flow, or in extreme cases, every packet [27, 16]. The result is a heavier stack, with encapsulation-induced delays and OVN tunnel overheads of additional layer-2 to 4 headers. This may lead to fragmentation and inefficient offload processing, or preclude CPU onloading of network functions.

However, the more critical performance aspect relates to VN's impact on the latency-sensitive datacenter applications, particularly on horizontally-distributed workloads such as PA. Latency and its tenant/user-level FCT metric have been recently established as crucial for the performance of this type of applications, typically classified as soft real-time. Their 200ms end-user deadlines [8, 40, 23] translate into constraints of few 10s of ms for the lower level PA workers. While the additional delay introduced by a VN may be negligible in a basic ping test [27], its impact on a more realistic PA workload can lead to an increase in mean FCT by up to 82% [13]. This raises justified concerns about potentially unacceptable OVN performance degradations for critical latency-sensitive applications such as Partition/Aggregate, analytics, databases, web servers, etc.

	Hypervisor	vNIC	vSwitch
LBr	Qemu/KVM	Virtio	Linux bridge
OVS	Qemu/KVM	Virtio	Open vSwitch
VALE	Qemu/KVM	Virtio	VALE
VMw-1	VMware	e1000	Vmnet
VMw-2	VMware	Vmxnet	Vmnet

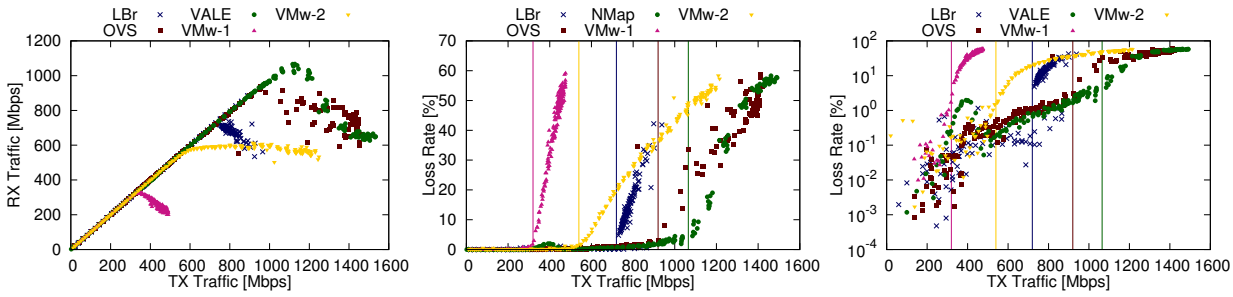
Table 1: Configurations.

2.2 Losslessness

Ideally, a VN should preserve the lossless abstraction provided by CEE/IB as it is assumed by the 'converged' datacenter applications, e.g., FCoE, RoCEE, MPI, etc. Yet we notice a paradox: Whereas CEE, IB, Fibre Channel, and most clustering and HPC networks provide a lossless data link layer (layer-2), currently *all* commercial and open-source virtual networks that we have tested during our work are *lossy*. Critical for the future of 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 IB, which predates CEE by a decade since its introduction of link level credit-based flow control, and in 2006, the FECN / BECN-based end-to-end Congestion Control Annex. In comparison, despite relatively simpler and lower-cost flow control implementations, current hypervisors, virtual switches and virtual network adapters are lagging behind the physical networks, resorting to drop packets at any potential congestion, not only do they degrade the datacenter performance, but they also fail to correctly terminate the CEE fabrics, to a large extent canceling the investments in a lossless physical network. As an alternative, we demonstrate how a zero-loss OVN (zOVN) can meet both the correctness *and* the FCT performance requirements of the virtualized datacenter.

2.3 Loss measurements

To support the above claims, we assess the extent of the packet dropping problem, using commonly available virtualization solutions. We will evaluate (i) where and how frequently losses occur, and (ii) the maximum bandwidth that a virtual switch can sustain without dropping packets. To answer these questions, we per-



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

Figure 3: Experimental loss results.

formed the experiment shown in Figure 1, in which VM1 and VM2 act as sources and send their traffic towards VM3 acting as sink, creating a common congestion scenario.

We measured different combinations of hypervisors, vNICs, and virtual switches. As hypervisors we considered Qemu-KVM [4] and VMware[2] Player [5]. These were used with two types of vNICs: an emulated Intel e1000 NIC and a virtualization optimized virtio [33] / vmxnet NIC, for Qemu and VMware, respectively. With VMware, we used the internal bridged virtual network configuration to interconnect the three VMs. In combination with Qemu we used a Linux Bridge [2], an Open vSwitch [3] and VALE [32]. The entire setup is hosted on a Lenovo T60p laptop equipped with an Intel Core2 T7600 @ 2.33GHz CPU and 3GB of memory. Both the physical host and the VMs run Ubuntu 11.10 with a 3.0 64-bit kernel. Across all experiments iperf [1] injects 1514B MTU frames as 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: The two generators injected traffic at full speed for 10s with the last packet being marked. Between each measurement point we computed the number of lost packets as the difference between the transmitted and the received packets 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); (ii) *receive stack losses*, i.e., packets received by the destination vNIC (5) and never forwarded to the sink (6). Since the TX path is backpressured up to the vSwitch, no losses were observed between other measurement points. A more accurate analysis of the possible loss points is presented in Section 4. With VMware and VALE, we could not access measurement points 3 and 4. Hereby the difference between the measurements done at the sender vNICs and at the receiver vNIC (points 2 and 5, respectively) was accounted for as vSwitch losses. The results are plotted in Figure 2. Since VALE combined with e1000 vNICs was affected by an implementation bug which allows the internal queues to grow indefinitely, resulting in substantially diverging results between runs, we omit these configurations.

Depending on configuration, the total traffic forwarded during the 10s window was highly variable. In the virtualized network, performance is always bounded by the computational resources assigned to each block by the host operating system (OS), same for all runs. On one hand, compute intensive configurations, such as Qemu with e1000 vNICs which need to emulate fake hardware to deceive the driver, reach lower throughputs - which induce less losses in the vSwitch. On the other hand, the virtualization-optimized vNICs, i.e., virtio and vmxnet, achieved higher rates, causing overflows in the vSwitch. The performance-optimized VALE vSwitch shifted the bottleneck further along the path, into destination VM stack. These results bear witness to the lack of flow control between the virtual network devices, and confirm our initial conjecture.

Experiment 2: With the five configurations from Table 1, we analyze the maximum bandwidth a virtual switch can sustain. We varied the target injection rate at each generator starting from 5 Mb/s, in increments of 5 Mb/s; thus the aggregated vSwitch input traffic is 2x larger. Figure 3a plots the RX rate as a function of the total injection rate, whereas Figure 3b plots the packet loss ratio. Both were computed at the application level (measurement points 1 and 6). For all configurations, we observed saturation behaviors as shown in Figure 3a. The RX rate first increased linearly with the TX rate until the saturation peak. Beyond this, with the exception of VMw-2, instead of a desired steady saturation plateau, we observe a drop indicative of congestive collapse. The overloaded system was wasting resources to generate more packets, instead of dedicating suffi-

[2]All other marks and names mentioned herein may be trademarks of their respective companies.

	Min. BW w/ losses	Max. BW w/o losses
LBr	60 Mbps	294 Mbps
OVS	139 Mbps	333 Mbps
VALE	100 Mbps	256 Mbps
VMw-1	136 Mbps	147 Mbps
VMw-2	20 Mbps	270 Mbps

Table 2: vSwitch performance variability.

cient resources to the vSwitch and the destination VM3 to actually forward, respectively consume, the packets already injected and backlogged in the virtual queues. Although the saturation load varied considerably across configurations, loss rates well in excess of 50% were observed for all configurations (Figure 3b). Even far below the saturation offered load, marked by vertical lines, we measured losses in the virtual network significantly above the loss rates expected from its physical counterpart (Figure 3c): up to 10^{-2} instead of 10^{-8} for MTU-sized frames with a typical Ethernet bit error rate value of 10^{-12}.

Table 2 reports the minimum bandwidth with measured losses and the maximum bandwidth without any loss. The wide dynamic ranges confirm our intuitive hypothesis about large non-causal performance variability in virtual networks, as the service rate of each virtual link depends critically on CPU, load, OS scheduling and the computational intensity of the virtual network code. Suboptimal and VN-load oblivious OS scheduling causes frequent losses: E.g. scheduling a sender before a backlogged receiver. Lossless virtual switches would be of great interest, not only for efficiency purposes, but also in terms of introducing a minimal level of predictability - if not deterministic behavior. The next sections will present how a basic 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 network infrastructure must simultaneously satisfy the union of the requirement sets from the domains being converged. For example, losslessness is a *functional* requirement of converged HPC, storage and IO –e.g., FCoE, PCIe over Ethernet, RoCEE– applications, whereas OLDI workloads impose a key *performance* requirement 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 main objectives are:

1) Reconcile the FCT-measured application *performance* with datacenter *efficiency* and ease of management, by proving that network virtualization and horizontally-distributed latency-sensitive applications are not necessarily mutually exclusive. This reconciliation removes the main obstacle hindering virtual network adoption in performance-oriented datacenters.

2) Prove that *commodity* solutions can be adapted for sizable performance gains. As shown in Section 5, an 19-fold FCT reduction is attainable also without a clean-slate deconstruction of the existing fabrics and network stacks. Instead, one can achieve *comparable* performance *gains* by just reconfiguring the CEE fabric using the standard TCP stack. When total cost of ownership and management costs are considered, 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 for both the (a) FCT performance of horizontally-distributed latency-sensitive workloads, and (b) loss-sensitive converged storage and HPC applications in multitenant datacenters. This basic HPC tenet has already been proven by decades of experiences in large-scale deployments. Since IB and CEE fabrics are widely available with line rates between 10 and 56 Gbps, the datacenter can now also benefit from prior HPC investments in lossless interconnection 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, whether CEE or IB, and the thrust of SDN/OpenFlow have prompted us to reconsider the end-to-end and "dumb network" arguments in the context of datacenters. First explicitly stated in [34], the end-to-end principle 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: E.g., error detection and correction (EDC) should reside in NICs and OS protocol stacks, instead of in switches and routers. While one of the most enduring network design principles, it can be harmful to end-to-end delay, FCT and throughput [14].

In datacenters, the delay of latency-sensitive flows is impacted not only by network congestion, but also by sender's and receiver's protocol stacks [31]. Historically, for low-latency communications, both Arpanet and In-

ternet adopted "raw" transports—unreliable, yet light and fast—instead of TCP-like stacks. Similarly, IB employs an Unreliable Datagram protocol for faster and more scalable "light" communications, whereas HPC protocols have traditionally used low-latency endnode stacks based on the assumption of a lossless network with very low bit error rates.

Given the high relevance of latency-sensitive datacenter applications, current solutions [8, 9, 40, 37] took an intriguing option: Decouple flow control (correctness, reliability) from the fabric. In this paper we show that coupling flow control with the fabric positively impacts applications performance.

3.3 OVN Design Space

The simplest virtual networking solution 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 networking. The increasing number of VMs has led to a MAC address explosion, whereby switches need increasingly larger forwarding tables. Moreover, dynamic VM management stresses the broadcast domains [27]. Furthermore, today's 4K VLAN limit is insufficient for multitenant datacenters unless the complex 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 achieve full network virtualization are the OVNs. Although a number of overlays have been recently proposed [20, 36, 25, 27, 11, 16], their key architectural abstraction lies in the separation of the virtual networking from the underlying physical infrastructure. Overlays enable the 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, typically located within the physical hosts, intercept VM traffic, perform en-/de-capsulation and tunnel it 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 [27, 25, 11, 16], 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, independent of location, domains and the physical networking capabilities. Thus, virtual switches are similar to the traditional hypervisor switches, but with additional functions as overlay nodes.

What about performance? There are a few key aspects in which OVNs influence datacenter performance and scalability. First, on the data plane: OVNs use encapsulation to build tunnels between the virtual switches that host a connection's source and destination. Current encapsulation solutions such as VXLAN [25] and NVGRE [36] solve the original VLAN limitation while reducing the configuration and management overhead. Second, on the management plane: Network configuration, distribution and learning protocols are necessary for tunnel creation at each virtual switch. To create a tunnel, the overlay switch needs to map the destination address to its physical location. The overlay configuration management can be performed either by learning or in a centralized fashion. The *learning* approach, adopted by VXLAN [25], floods packets with unknown destinations. In the *centralized* approach, the virtual switches are responsible for retrieving the information required for encapsulation. In NetLord [27], this information is learnt by switches through communication with each other, and from a central configuration repository. In DOVE [11, 16], this configuration information is retrieved from a centralized database. Both the central configuration repository in NetLord and the centralized database in DOVE must be highly available and persistent, which poses a challenge for multi-million node datacenters, thus raising the future third option of a distributed repository approach, presuming the entailing coherency issues can be solved efficiently. For now, the former two approaches, learning and centralized, 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.

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 as shown in Figure 10. 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, and aiming to simplify the description, we assume a single lossless priority is used. In a real setup, different priority classes can be configured to segregate delay / loss tolerant traffic, not requiring losslessness, 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 (applica-

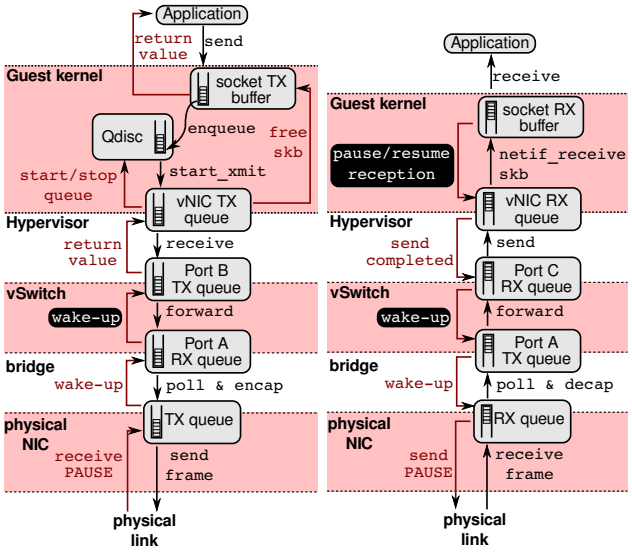


Figure 4: Life of a packet in a virtualized network.

tions) 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 4.

After processing within the VM's guest kernel the packets are transferred to the hypervisor through a vNIC. The hypervisor sends the packets to the virtual switch. The virtual switch assures the communication between VMs and the physical adapter. The packets destined to remote VMs are taken over by a bridge that encapsulates and moves them to the physical adapter queues. Packets travel through the physical network and are delivered to the destination server. Here they are received by the remote bridge, which decapsulates them and moves them into the destination's virtual switch. The virtual switch forwards the packets to the hypervisor, which in turn forwards them to the guest OS. After processing in the guest kernel, packets are finally delivered to the destination application. Based on careful analysis of the entire end-to-end path, we identified and fixed the points of loss marked in white on black in Figure 4, i.e., the vSwitch and the reception path in the guest kernel.

4.1.1 *Transmission Path*

On the transmit side the packets are generated by the user-space processes. As shown in Figure 4 the process issues a `send` system call that copies the packet from user space to the guest kernel space. After the copy, the packets are stored in an `sk_buff` data structure that is enqueued in the transmit (TX) buffer of the socket opened by the application. The application is aware whether the TX queue is full through the return value of the system call, making this operation lossless.

The 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. The pointers are sorted according to the selected discipline, which is FIFO by default. To avoid packet 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 enqueueing 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 entire transmission path in the guest OS is lossless as long as the Qdisc length is properly sized.

Our architecture is based on virtio technology [33], hence the virtual adapter queues are shared between the guest kernel and the underlying hypervisor software running in the user space of the host. The virtio network 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 dequeueing 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 [31] backend. We took the Netmap backend code delivered with the VALE [32] virtual switch and we ported it to the latest version of Qemu with the necessary bug fixes related to concurrent access to the Netmap rings. Attention is required to implement a lossless coupling between the device and the backend, avoiding via configuration flags, the use of the lossy Qemu VLANs. Packets arrive at the vSwitch TX queue of the port where the VM is attached. The vSwitch moves the packets using a forwarding (FIB) table, from the TX queues of the input ports to the RX queues of the output ports. The forwarding table contains only the MAC addresses of the locally connected VMs. If the destination is found to be locally connected, the packets are moved to the corresponding RX queue, else they are enqueued in the RX port corresponding to the physical interface. From the physical interface port the packets are consumed by a bridge that encapsulates and enqueues the packet in the TX queue of the physical adapter. Then the lossless physical network takes over the packet and delivers it to the destination server physical RX queue.

As shown in Section 2.3, none of the current virtual switches implement flow control, fact also confirmed

Algorithm 1 Lossless Switch 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

by our discussions with some of the leading vendors. Therefore we have redesigned the VALE switch to add internal flow control and make the TX path fully lossless, as described in Section 4.2.

4.1.2 Reception Path

The packets are consumed by the bridge from the RX queue of the physical NIC and decapsulated. Next, they are enqueued in the TX queue of the virtual switch that forwards them to the RX queue corresponding to the destination VM. The forwarding is again lossless; see Section 4.2 for the details. The packets are consumed by Qemu hypervisor that 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 is received

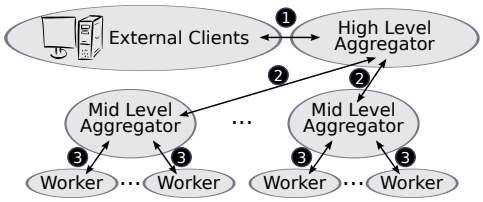


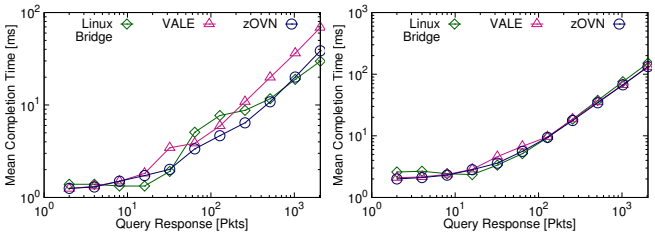
Figure 5: Partition-Aggregate application.

and the guest OS receives an interrupt. This interrupt is handled according to the Linux NAPI framework. A softirq is raised, which triggers consumption of the packets from the RX queue. The packet is transferred to the `netif_receive_skb` function that does the IP routing and filtering. If the packet is found to be 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 amount of injected packets to the advertised window of the receiver. On the other hand, with UDP sockets additional care is required. We modified the Linux kernel such that when the destination socket RX queue occupancy reaches a threshold—one MTU below maximum—the softirq is canceled and the reception is paused. Once the process consumes data from the socket, reception is resumed. This ensures full lossless operation both for TCP and UDP sockets.

4.2 zVALE: Lossless virtual Switch

As stated before, our lossless vSwitch is derived from VALE [32], which is based on the Netmap architecture [31]. It has one port for each VM running on the server 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 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 is forwarding packets from input queues to output queues as fast as they arrive. If any output queue is full packets are discarded.

To make such a software switch lossless we 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 is produced, the sender checks whether the associated TX 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 TX queue and then starts a forwarding process to try to *push* some 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



(a) 1 server (b) 2 servers

Figure 6: Microbenchmarks: 6 VMs PA scenarios.

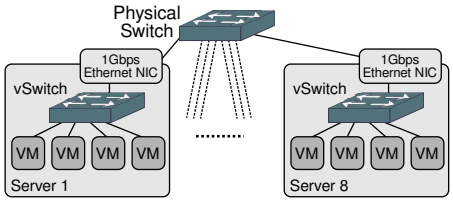


Figure 7: Testbed setup: 8 servers.

up the corresponding consumers that might be waiting for new packets. On the receiver side, the associated output queue is checked. If a queue is not empty, a packet is consumed from it, else the forwarding process is started to *pull* some packets from the input queues to this output queue. If some data is pulled, then 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, the sender will sleep most of the time waiting for free buffers and the receiver will wake it up only when it consumes data. On the other hand, when the receiver is faster than the sender, the receiver will sleep most of the time and the sender will wake it up only when new data is available. The overhead of lossless operation is thus reduced to a minimum.

5. EVALUATION

In this section we evaluate our proposed lossless vSwitch architecture, applying the PA workload described in Section 5.1. We run this workload both in a lab-scale experiment with eight virtualized servers hosting 32 VMs and in a larger-scale simulation using an OMNeT++ model of a 256-server network.

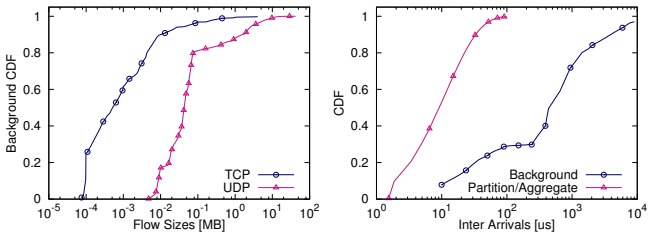
5.1 Partition/Aggregate Workload

A generic 3-tier PA application is presented in [8, 40] and illustrated in Figure 5. 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 a 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 within 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* the results have been received, the MLA sends back 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 we employ a reduced two-tier PA workload, in which the MLAs are 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 different 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 in a two-server configuration. The servers were Lenovo M91p-7034 desktops (Intel i5-2400 @ 3.10GHz CPU , 8GB memory, Ubuntu 11.10 with a 3.0 64-bit kernel both for host and guests). The servers were connected through a 1 Gbps 3com 3GSU05 consumer Ethernet switch supporting IEEE 802.3x. The host kernel was patched with the Netmap [31] 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 6a the aggregators and the workers resided in VMs on the same server whereas in Figure 6b the aggregator was on a different server than the workers. We varied the size of the workers response to the aggregator from 2 to 2048 MTUs. To achieve statistical confidence each run consisted of 10K repetitions. We report the mean query completion time in Figure 6. We compared the Linux Bridge with the lossy VALE implementation [32] 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 correctly find the 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 to 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 bot-



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

Figure 8: Flow size and inter-arrival distribution.

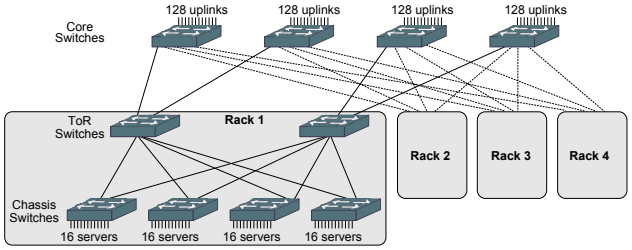


Figure 10: Simulated topology: 256 servers.

tleneck in this case. Therefore, we measured loss ratios of less than 0.02%. Enabling losslessness on such a configuration brings no additional benefits. On the other hand, this result validates the efficiency of our implementation.

In the single server setup the zOVN switch was consistently better than the lossy VALE switch across all runs. The Linux Bridge showed performance variability (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 previous scenario the extra overhead was hidden by the physical link bottleneck.

5.3 Lab-Scale Experiments

Next, we deployed zOVN over the testbed shown in Figure 7, containing eight Lenovo T60p notebooks (Intel Core2 T7600 @ 2.33GHz CPU, 3GB memory) connected through a 1 Gbps HP 1810-8G managed Ethernet switch supporting IEEE 802.3x. The host and guest kernels were identical to the ones used previously.

We ran a PA workload using 32 VMs with the same methodology and flow sizes as in the previous paragraph. In addition we varied the TCP version between NewReno, Vegas and Cubic. In Figure 9 we report the mean completion time, throughput, and performance gain of lossless over lossy. Mean flow completion time was reduced by a factor of up to 19.1×. The highest benefit was achieved for flow sizes between 6KB and 48KB (4 and 32 packets). For very small flows the total size of all the workers responses was too small to cause any buffer overflow. For long flows the losses were recovered through fast-retransmit. All TCP versions performed about equally.

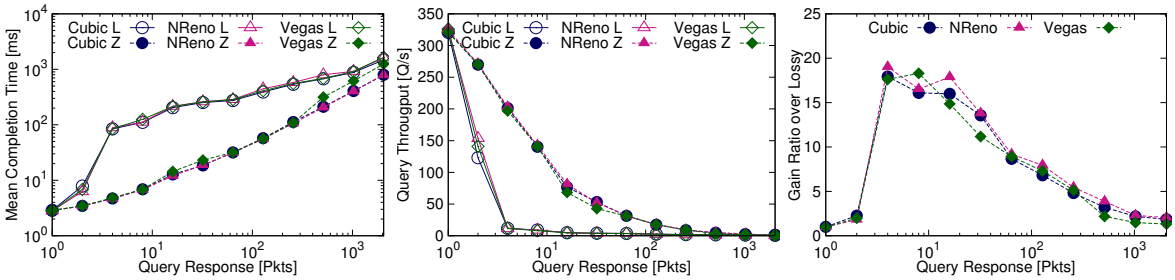
In Figure 11 we report the same metrics, but with added background traffic. In this scenario, each VM hosts an additional traffic generator producing background flows. The generator chooses a random uniformly distributed destination, then it sends to it a TCP flow with the length drawn from the distribution in Figure 8a. Afterward, the generator sleeps according to the background flow inter-arrival distribution shown in Figure 8b. Both PA and background flows use the same TCP version. The gain is smaller than in the previous scenario, because the background flows also benefit from losslessness. 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.

5.4 Simulation Experiments

To finalize our validation, we implemented a model of the zOVN virtualized network on top of the OMNeT++ network simulator. The simulator models at frame level a 10G CEE fabric with generic input-buffered output-queued switches. We model a faster fabric than the one used in the testbed measurement in an attempt to quantify the behavior of our proposal on fabrics currently used in production datacenters. 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, mostly 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 immediately reply. 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, 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 10,



(a) Completion time. (b) Throughput. (c) Performance gain.

Figure 9: Testbed results: 32 VMs PA running on 8 servers. Without background traffic.

consisting of 256 servers, distributed in 16 chassis, 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 the 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-percentile for different flow control configurations under no, light, and heavy 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). We also considered the same three TCP versions as before.

Enabling flow control in only one network (either physical or virtual) almost entirely nullified the benefits, because packet losses were just shifted from one domain to the other. However, the effect was not totally the same, because virtual flow control still benefited the flows between VMs on the same host. Therefore, enabling only virtual flow control (ZL) still achieved a performance improvement, although significantly smaller than the ZZ case. Enabling both flow controls (ZZ) achieved significant gains similar to the ones 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 with heavy UDP background traffic enabling flow control harms performance. In this case, the uncooperative background UDP packets did not get dropped anymore and, consequently, hogged link capacity and harmed the foreground PA workload traffic. This results confirmed the need to segregate the traffic into PFC priorities with true resource separation and scheduling.

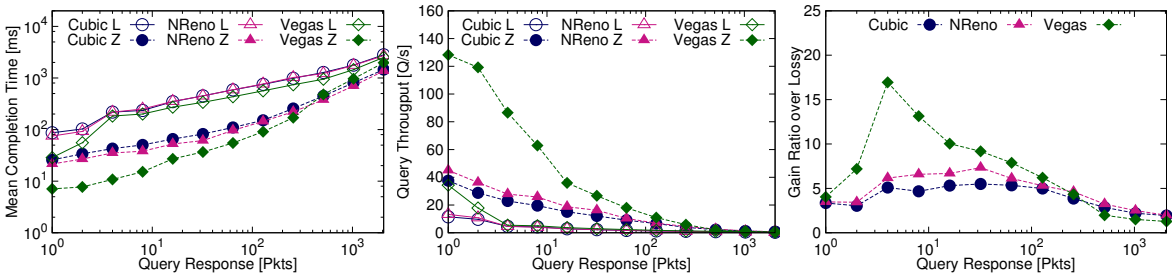
In the scenario with background traffic, Vegas outperformed NewReno and Cubic confirming the results obtained on the testbed setup. In the case without background traffic Vegas was again better. On the other hand, on the testbed all TCP produced similar results. This difference is due to the more complex communication pattern with more hops, where more flows share the same path. This produces longer queues, especially in the core switches. These hotspots produce longer delays that are detected by Vegas, that in turns reduces its congestion window to avoid packet losses, obtaining thus shorter completion times.

6. DISCUSSION

We review the main takeaways from the preceding results. Using the zOVN's experimental platform we demonstrated both correctness, i.e., no packet drops to support converged storage and HPC applications, *and* improved FCT performance. Thus, we have achieved our primary objective of reconciling performance with correctness for overlay virtual networks.

Is lossless flow control more relevant for physical or 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 respective domain, yields the largest reductions in FCT: PA over zOVN with 32 virtual workers distributed across eight physical servers achieved a 19-fold peak speedup. Relevant to OLDI workloads in general, the highest speedups recorded are for flows between 6 and 50 KB.

Unexpectedly, if a suboptimal choice must still be made between flow control in either the physical or the virtual network, 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 IB 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



(a) Completion time. (b) Throughput. (c) Performance gain.

Figure 11: Testbed results: 32 VMs PA running on 8 servers. With background traffic.

lower implementation and configuration effort.

Is our modest experimental platform relevant for blade-based racks and top-of-the-rack switches with 40 Gbps uplinks? While the definitive answer entails a multi-million dollar datacenter setup, we are confident in the relevancy of our admittedly limited prototype platform. 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 measured. In our zOVN experiments, we consistently observed a correlation between CPU performance and network capacity. A fast (or unloaded) CPU coupled to a slow network produces much less packet losses in the virtual switch than a slow (or loaded) CPU coupled to a fast network. A fast CPU has more computational resources to assign to the vSwitch to handle the network load.

7. RELATED WORK

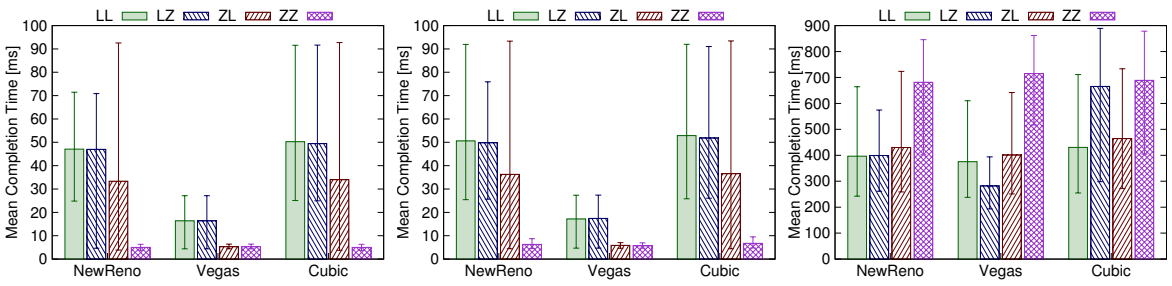
In recent years, the TCP incast and FCT-based performance of PA applications has been extensively analyzed. For example, [15, 38] suggest a 10-1000× retransmission timeout reduction. Other proposals achieve sizable FCT reductions for typical datacenter workloads using new single-path [8, 9, 40, 37] or multi-path [41, 23, 35, 7] transports, in conjunction with deadline-aware or agnostic schedulers and per-flow queuing. Related to our work and to [21, 17], DeTail [41] identifies packet loss in physical networks as one of the three major issues; the authors enable the PFC mechanism and introduce a new multi-path congestion management scheme targeted against flash hotspots typical of PA workloads, also employing explicit congestion notification (ECN) against persistent congestion. While DeTail uses a TCP-compatible version of NewReno to reduce FCT by 50% at the 99.9-percentile, it 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, while relegating the deadline-aware global scheduling complexity to a greedy scheduler and a simplified retransmission scheme to recover from losses. By replacing both the TCP stack *and* the standard datacenter fabric, this scheme achieves near-ideal FCT performance for short flows. Open issues are scalability to datacenter-scale port counts, costs of replacing commodity fabrics and TCP, 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 AQM setup in the congestion point detector with a smooth congestion window reduction function reminiscent of QCN's rate decrease. While DCTCP reduces the FCT by 29%, as a deadline-agnostic TCP it misses ca. 7% of the deadlines. D3 [40] is a deadline-aware first-come first-reserved non-TCP transport. Its performance comes at the cost of priority inversions for ca. 33% of the requests [37], and a new protocol stack. PDQ [23] introduces a multi-path preemptive scheduling layer for meeting flow deadlines, using FIFO taildrop similar to D3. By allocating resources first to the most critical flows, PDQ improves by ca. 30% on D3, RCP and TCP. As it is not TCP, its fairness remains to be studied. D2TCP [37] 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, 22, 30] 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



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

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

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 are lagging behind. Congestion, whether inherent to the traffic pattern, or as an artifact of transient CPU overload, is still handled by dropping packets, thus breaking correctness, 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 correctness and performance.

To remedy this situation, we first identified the origins of packet drops across the entire virtualized communication stack, and subsequently designed and implemented a fully lossless virtual network prototype.

Based on this experimental results using our prototype implementation, as well as larger-scale simulations, we have demonstrated average FCT improvements of one order of magnitude. Additional takeaways from this work are that (i) packet loss in virtualized datacenters is even costlier than previously shown in physical networking, (ii) FCT performance of Partition/Aggregate workloads is greatly improved by losslessness in the virtualized network, and (iii) commodity CEE fabrics and standard TCP stacks still have untapped performance benefits.

9. REFERENCES

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

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

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

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

[5] VMware Player. Available from: `http://www.vmware.com/products/player/`.

[6] P802.1Qbb/D2.3 - Virtual Bridged Local Area Networks - Amendment: Priority-based Flow Control, 2011. Available from: `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, B. Prabhkakar, and S. Shenker. Deconstructing Datacenter Packet Transport. In *Proc. HotNets 2012*, Redmond, WA, October 2012.

[11] K. Barabash, R. Cohen, D. Hadas, V. Jain, R. Recio, and B. Rochwerger. A Case for Overlays in DCN Virtualization. In *Proc. DCCAVES'11*, San Francisco, CA, September 2011.

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

[17] 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.

[18] D. Crisan, R. Birke, N. Chrysos, and M. Gusat. How Elastic is Your Virtualized Datacenter Fabric? In *Proc. INA-OCMC 2013*, Berlin, Germany, January 2013.

[19] 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.

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

[21] 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.

[22] 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.

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

[24] 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.

[25] 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.

[26] 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.

[27] J. Mudigonda, P. Yalagandula, J. C. Mogul, B. Stiekes, and Y. Pouffary. NetLord: A Scalable Multi-Tenant Network Architecture for Virtualized Datacenters. In *Proc. ACM SIGCOMM 2011*, Toronto, Canada, August 2011.

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

[29] 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.

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

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

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

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

[34] 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.

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

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

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

[38] 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, August 2009.

[39] 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.

[40] 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.

[41] 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.