THE LOSSLESS NETWORK

For Data Centers

HUAWEI TECHNOLOGIES CO., LTD.

BAIDU, INC.

Revision 1.0

November 7th, 2017

Abstract

Data centers are tasked with delivering intelligent multi-media responses to real-time human interactions. Massive amounts of data are being churned and sifted by highly parallel applications, such as Online Data Intensive Services (OLDI) and Artificial Intelligence (AI), which historically required specialized High-Performance Computing (HPC) infrastructure. New advancements in high-speed distributed NVMe storage, coupled with new networking technologies to better manage congestion, are allowing these parallel environments to run atop more generalized next generation Cloud infrastructure. The key to advancing Cloud infrastructure to the next level is the elimination of loss in the network; not just packet loss, but no throughput loss and no latency loss. In the network, congestion is the common enemy. This paper discusses the need for new technologies to combat loss in the data center network.

Our Digital Lives are Driving Innovation

For better or worse, our lives are forever changed by digital technology. Digital technology is increasingly accessed and offered as a service from the cloud. Our lives and digital technology are coming together as cloud services become more a part of our natural lives.

Interacting with cloud services is now done in a human and natural way – through voice commands and visual recognition. Someday, in the not too distant future, as predicted by Futurist Ray Kurzweil [1], the way we think will be augmented by the cloud. Already today, services are personalized to our individual tastes by online data intensive cloud services. We've come to expect instantaneous access to massive amounts of digital content by our very own voice commands. But how does all this work – in the backend – in the data center? How is it that massive amounts of data can be rendered into useful information within a timeframe that meets real-time human interaction delays?

The requirement to integrate digital technology into our natural lives is driving innovation in the data center. This innovation is driving the need for new levels of performance, scale and reliability from the infrastructure. Enormous amounts of computing cycles are rendering massive amounts of data into real-time information and action. The delivery of information and action from the cloud data center needs to be fast! As a consequence, the fabric within the data center needs to eliminate loss and deliver low latency and high throughput.

Trends in the Data Center

Application and storage architectures within the data center are frequently evolving to address increasing demands for real-time, interactive digital technology. Currently, three critical data center use cases are stressing today's data center network. These include large scale Online Data Intensive (OLDI) services such as automated recommendation systems for online shopping, social media and web search; High performance Deep Learning networks; and high speed distributed pools of Non-Volatile Memory Express (NVMe) storage.

OnLine Data Intensive (OLDI) Services

The fundamental difference between Online Data Intensive services and their offline counterparts (such as MapReduce computations) is that they require immediate answers to requests that are coming in at a high

rate. Latency control is a key concern. The end-user experience is highly dependent upon the system responsiveness, and even moderate delays of less than a second can have a measurable impact on individual queries and their associated advertising revenue. A large chunk of unavoidable delay, due to the speed of light, is inherently built into a system that uses the remote cloud as the source of decision and information. This puts even more pressure on the deadlines within the data center itself. To address these latency concerns, OLDI services deploy individual requests across 1000s of servers simultaneously. The responses from these servers are coordinated and aggregated to form the best recommendations or answers. Delays in obtaining these answers are compounded by delayed or 'straggler' communication flows between the servers. This creates a long tail latency distribution in the data center for highly parallel applications. To combat tail latency, servers are often arranged in a hierarchy, as shown in Figure 1, with strict deadlines given to each tier to produce an answer. If valuable data arrives late because of latency in the network, the data is simply discarded and a sub-optimal answer may be returned. Studies have shown that the network becomes a significant component of overall data center latency when congestion occurs in the network [2].

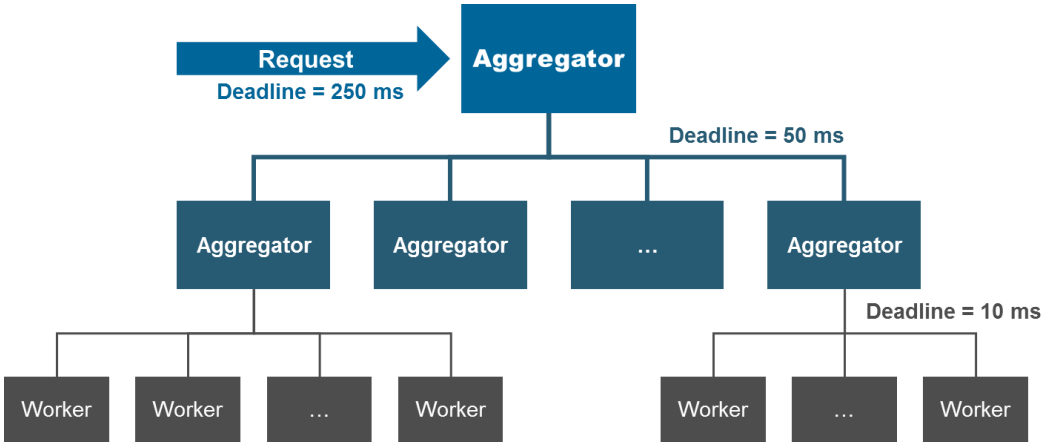

Figure 1 – Parallel Application Hierarchy

The long tail of latency distribution in OLDI data centers can be caused by a couple of factors [3]. One is simply related to the mix of traffic between control messages (mice) and data (elephants). While most of the flows in the data center are mice, most of the bytes transferred across the network are due to elephants. So a small number of elephant flows can delay the set-up of control channels established by mice flows. Since OLDI data centers are processing requests over 1000s of servers simultaneously, the mix and interplay of mice and elephant flows is highly uncoordinated. Another cause of latency is due to incast at the tiers of the node hierarchy. Leaf worker nodes return their answers to a common parent in the tree at nearly the same time. This can cause buffer over-runs and packet loss within an individual switch. It may invoke congestion management schemes such as flow-control or congestion notification, which have little effect on mice flows and tail latency – more on this later.

Deep Learning

Deep Learning is a branch of Machine Learning that is having tremendous success at allowing computers, applications and cloud based services to see and hear. Everyday human tasks such as speech recognition and image recognition are being mastered by large neural networks, trained with millions and sometime billions of parameters, forming models that can be integrated into an online service. Complex tasks such as social network filtering, fraud and anomaly detection are performed effortlessly once these models are

formed. Think of the deep learning network as equivalent to a brain with its millions of neural interconnections. The larger the deep learning network, built from a larger number of model parameters, the better the network can perform at its job. Current deep learning networks can have billions of parameters and millions of interconnections [4].

Building the neural networks and deep learning models, a process called training, is often accomplished by high-performance computing systems. Training is a highly parallel application that requires low latency and high throughput. Throwing more computing resources at the problem can improve the time it takes to create a model; however, the communication overhead involved in the parallel application can offset the gains of more CPUs or GPUs. As seen in Figure 2, the huge training data sets are partitioned into chunks and distributed across a number of working clusters. Each cluster processes separate chunks of data, and returns gradient results to be folded together by a common parameter server or other peers in a coordinated fashion. The process repeats with model parameters being refined, reduced and redistributed until the model can recognize a known input with an acceptable level of accuracy. Once the models are built, they can be distributed and used as part of a new type of OLDI service that takes complex input such as voice, handwriting, high-resolution images and video.

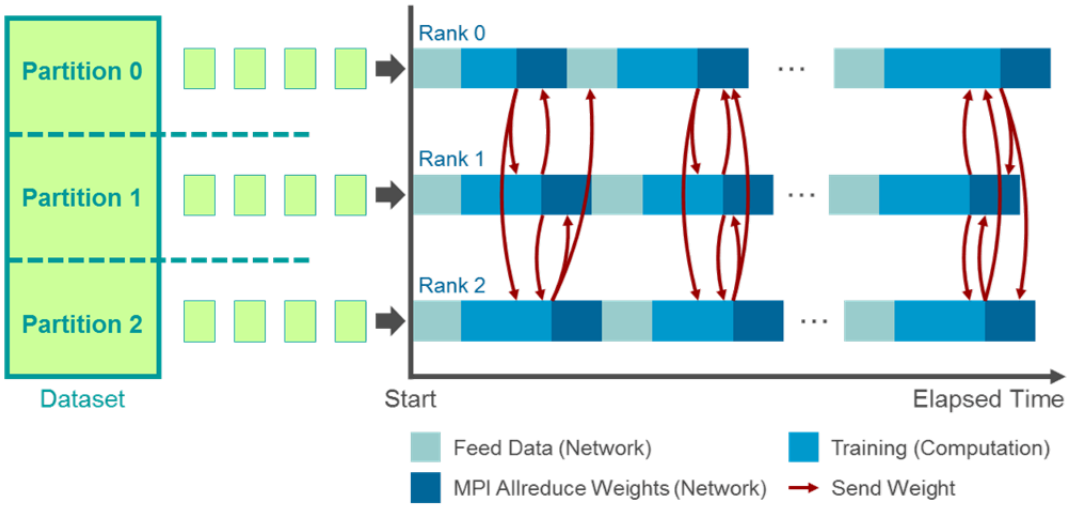

Figure 2 – Deep Learning Training

Deep learning models are constantly being trained and tuned. The challenge with this ongoing process is the high communication cost. Large amounts of data are frequently being shared and computation processes are stalled if synchronization delays occur. The network is often blamed for causing these training delays [5]. When a parameter server is used in the training process an inherent incast problem exists in the network. Clusters of worker nodes return gradient results to the parameter server at nearly the same time. This incast scenario creates congestion at the switch connecting the parameter server and can result in packet loss and synchronization delays. Further parallelizing the problem only compounds the delay as more communication is required between a larger numbers of nodes multiplying the impact of network congestion. Figure 3 shows that there is an optimal tradeoff between the number of parallel nodes and the time it takes to train a model. Reducing packet loss and improving latency and throughput can allow a larger number of parallel nodes to train the model, thus reducing the overall time.

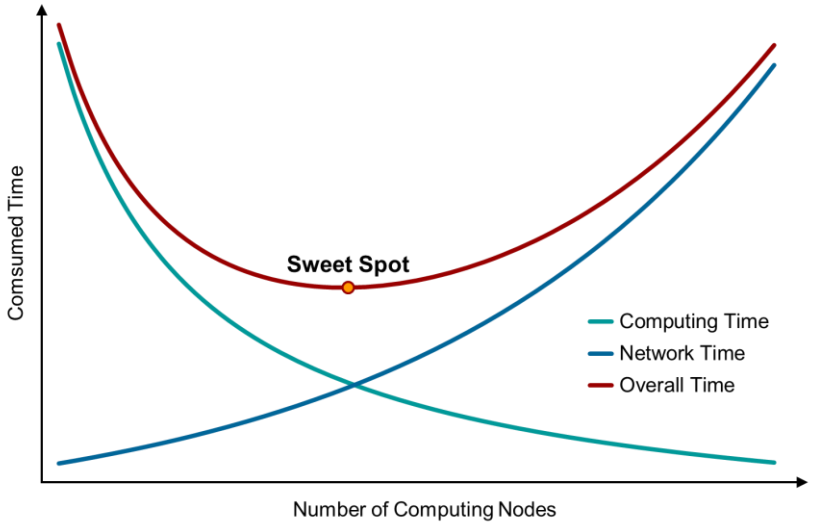

Figure 3 – Parallelism Tradeoff

NVMe over Fabrics

Non-Volatile Memory Express (NVMe) is a storage communications interface and protocol that was designed from conception to capitalize on the low latency and internal parallelism of flash-based storage devices known as solid-state drives (SSDs). NVMe is fast, reliable and a perfect fit for the highly parallel environments of the future cloud data center. All-Flash-Arrays (AFA) need NVMe access over the network. They need extremely low latency in order to compete with their on-board counterparts within servers. This latency needs to be on the order of 10µs [6] [7]. Going forward NVMe interfaces will only get faster and access latencies will continue to drop.

Cloud data centers are built on converged infrastructure where resources are pooled for lower cost, better manageability and higher utilization. This means high-speed NVMe storage needs to be accessed on the same infrastructure as virtualized computing and application nodes. However, the latency and reliability requirements of NVMe storage make this access a challenge. To reduce latency special host adapters utilize remote direct memory access (RDMA) communication semantics. RDMA supports zero-copy networking by allowing the network adapter to transfer data directly to or from remote application memory, bypassing the operating system. While extremely fast, bypassing the operating system means the network protocols responsible for reliable transmission and congestion control need to be implemented in hardware on the adapter. Resources on the adapter can be quite restricted and in order to keep cost and complexity low, some of the support for reliability and congestion control can be passed to the network.

First generation converged infrastructure focused on providing a large scale lossless layer-2 fabric to support Fiber Channel over Ethernet (FCoE) and RDMA over Converged Ethernet (RoCE). These Layer-2 networks needed to provide a lossless transport because the storage protocols themselves were not tolerant of packet loss and did not provide an adequate congestion control approach. The Layer-2 networks implemented priority-based flow control (PFC) and quantized congestion notification (QCN) to support a lossless environment for this first generation of converged infrastructure. Current cloud data centers are based on Layer-3 technology and storage protocols are running over TCP and UDP. The storage protocols over TCP and UDP take advantage of end-to-end congestion control to mitigate congestion, but packet loss is still a problem.

In the converged infrastructure data center, NVMe over Fabrics are specified to run over RoCEv2 (UDP-based) or iWARP (TCP-based). If the network detects congestion, it has the opportunity to mark packets with explicit congestion notification (ECN) indicators. The receiver will signal congestion notification messages back to the sender so that it can reduce the rate of injection in hopes of avoiding packet loss. If the round-trip time for these messages is too long, packet loss may still be unavoidable. Packet loss will require retransmission which will severely slow down NVMe storage access.

Parallelism

One common attribute that all of the above use cases have in common is parallelism. In order for large scale cloud services to meet real-time interactive latency requirements, the applications and storage must divide and conquer. There is simply too much data to process, and the true value of data is how quickly it can be rendered into human information and action. As Figure 4 suggests, parallelism in a distributed system depends upon an enormous amount of messaging for synchronization and parameter distribution. Inherent in this messaging are traffic patterns that create congestion due to incast and disorderly flows. Left unattended, congestion leads to overall loss in the network: packet loss, latency loss and throughput loss. Successful data centers of the future must eliminate this loss.

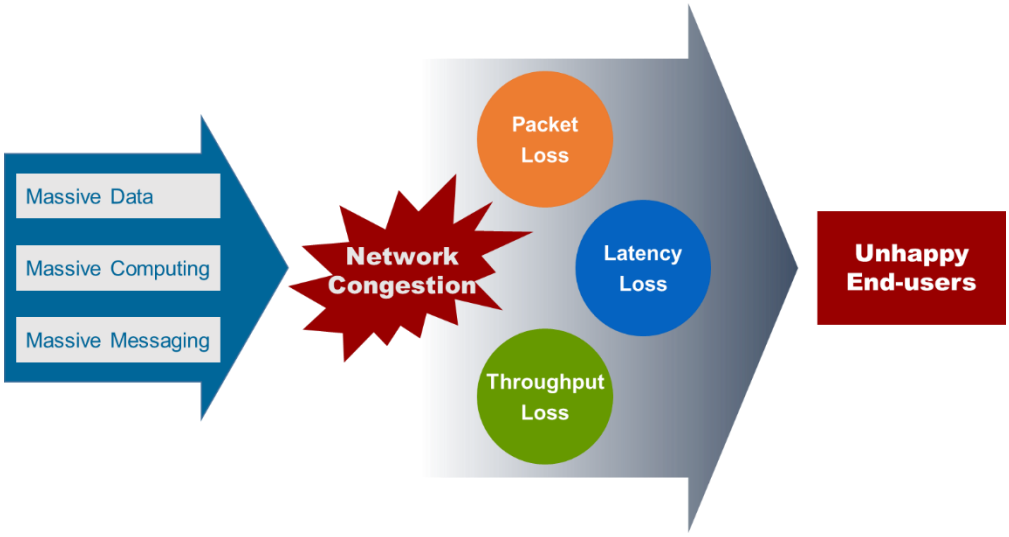

Figure 4 – The Problem with Network Congestion

Why Today's Data Centers Aren't Keeping Up

Whether building a public cloud or a private data center that operates as an internal cloud service for Enterprises, a common set of problems need to be addressed. Network designers need to build a highly flexible fabric for rapidly changing environments that carry a diverse set of traffic; application, storage and control. A common goal is to minimize or eliminate packet loss, provide high throughput while maintaining low-latency. These tenants are especially important to support the applications of OLDI, Deep Learning and NVMe over Fabrics.

The 3-Stage Clos network shown in Figure 5 is a popular network design in today's data centers. The Clos network achieves non-blocking performance and resiliency through equal cost multi-paths. Layer-3 networking is used between the switches because it is scalable, simple, standard and well understood. In the

Clos network, the top of rack (ToR) switches are the leaf switches. They are attached to the core switches which represent the spine. The leaf switches are not connected to each other and the spine switches only connect to the leaf switches.

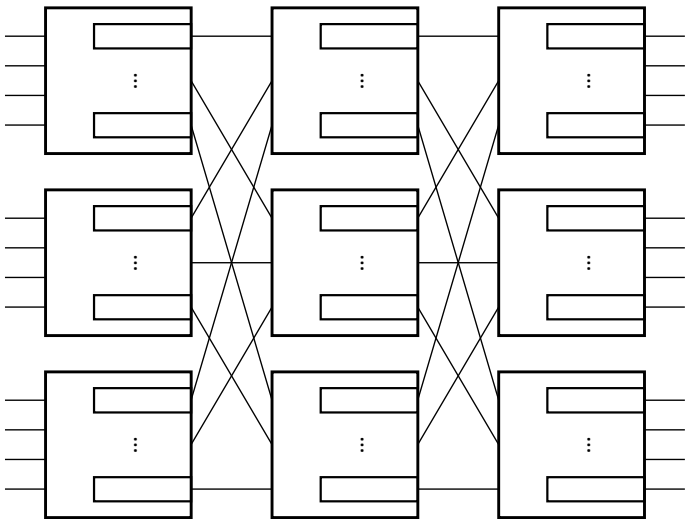

Figure 5 – 3-Stage Clos Network

There are multiple equal cost paths from each ToR switch to any other ToR switch in the network. As a consequence, a ToR switch can spread traffic across the multiple paths in order to balance the load and hopefully avoid congestion. The algorithm used for distributing traffic is called Equal Cost Multi-Path (ECMP) routing. As shown in Figure 6, ECMP typically selects a path by hashing the flow identity fields in the routed packet such that all packets from a particular flow traverse the same path.

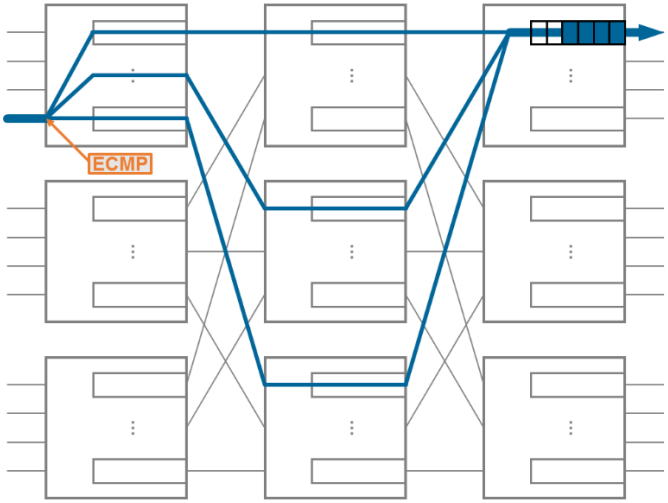

Figure 6 – ECMP Load Balancing

Server-to-server flows in the data center are TCP or UDP connections across the fabric. When congestion occurs in the network, packets are either dropped or the switches mark the IP packets with Explicit Congestion Notification (ECN) indicators. ECN allows end-to-end notification of congestion before dropping packets and is clearly the preferred approach. Figure 7 shows how the congestion feedback is returned to the sender via acknowledgement or specific congestion messages so the sender may reduce its rate of traffic

injection into the network. The way a sender adjusts its sending rate depends upon the protocols in use. Slight modifications to TCP for data center use are being proposed by the IETF's DCTCP specification [8]. Applications running over UDP are responsible for their own congestion control algorithms and most are using approaches that also recognize ECN indicators. RoCEv2, for example, runs over UDP and adjusts sending rate when it receives explicit Congestion Notification Packet (CNP) from the receiver.

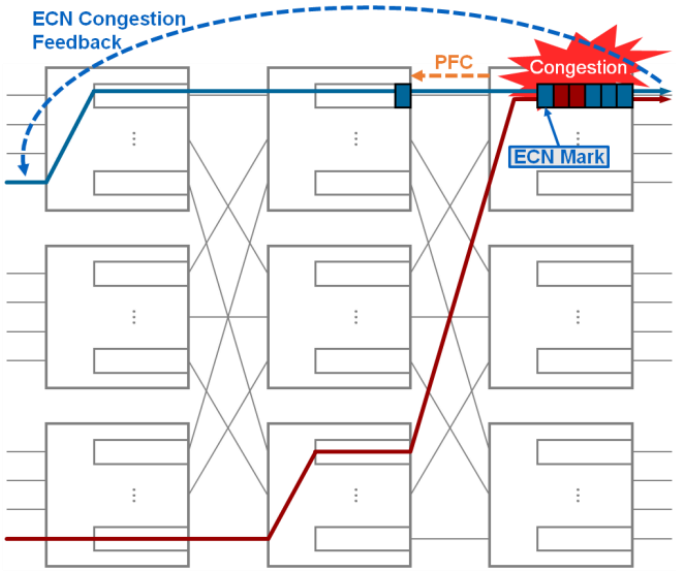

Figure 7 – Current Congestion Management

End-to-end congestion control is effective at getting the sending nodes to reduce their sending rates, but it does not completely eliminate the possibility of packet loss due to congestion. It takes some time for the ECN congestion feedback to make its way back to the source, and for the rate reduction to have an impact. Data that is already in flight and unfortunate traffic patterns, such as incast, will result in buffer overrun in the switches along the path. To avoid packet loss, which can have a dramatic effect on protocols such as RoCEv2, the IEEE 802.1 has defined a backpressure message call Priority-based Flow Control (PFC) [9]. A PFC message sent by the downstream switch signals to the immediate upstream switch to pause the sending of packets on a particular priority / traffic class in order to avoid buffer overrun. To avoid packet loss, the downstream switch needs to assure it has enough buffer headroom remaining to absorb the packets inflight on the link before issuing PFC. While pausing, if the upstream switch buffers fill, it may issue its own PFC message to the next upstream switch, and so on, until eventually the sending node is paused. Typically, these congestion hotspots are temporary and PFC never has to propagate very far back, but PFC itself is a heavy hammer and has other negative implications – which will be discussed later.

The technologies used in today's state-of-the-art data center are all designed for congestion management, and while they have made improvements, they still fall short of providing the lossless data center network required for future use-cases. In particular the following issues remain:

ECMP collisions

Selecting a path by hashing the flow identifiers is simple but does not take into consideration whether the selected path itself is congested. It is quite easy for the identity of multiple flows to hash to the same selection, resulting in overloaded links, as seen in Figure 8. Additionally, flow size is typically bi-modal (mice

or elephants), with the majority of flows being mice, but the majority of bytes transferred being from elephants. ECMP does not consider flow size when selecting a path. It is unfortunate when ECMP collisions occur on elephant flows because the chance of creating in-network congestion is much greater. Improvements to ECMP could involve being more congestion aware when selecting a path and load balancing traffic at a finer granularity.

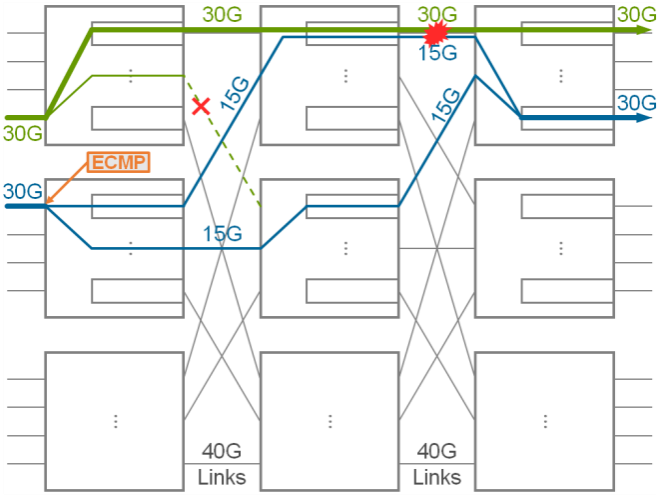

Figure 8 – ECMP Load Balancing Collisions

ECN control loop delays

There is a desire to scale data center networks larger to eliminate bottlenecks, simplify workload provisioning and reduce costs. Large networks have more hops, and as a consequence, have a longer round-trip-time (RTT) for the ECN control loop. Larger networks can also support more data in-flight, making it difficult to absorb bursts of traffic before ECN congestion control can reduce the sending rate. Adding more switch buffers to absorb bursts is not desirable because it increases cost and increases network queuing delays for innocent well behaved flows. End-to-end congestion control is essential to orderly networks, but additional assistance is needed to assure it can be effective and avoid packet loss.

PFC head-of-line blocking

PFC is a technique to avoid packet loss, but it is a heavy hammer and should be used as a last resort. PFC is invoked when switch ingress buffers back-up because of congestion at one of the egress ports. It is common for some of the flows arriving on the ingress port to be destined to other non-congested egress ports within the switch. However, because PFC will stop all traffic in a particular traffic class at the ingress port, the flows destined to other ports will also be blocked. The phenomenon is known as head-of-line blocking, as seen in Figure 9. To avoid head-of-line blocking it is critical to identify the flows that are causing congestion as early as possible and provide congestion mitigation techniques that are specific to the flow's characteristics. The flows that are causing congestion are most frequently elephant flows.

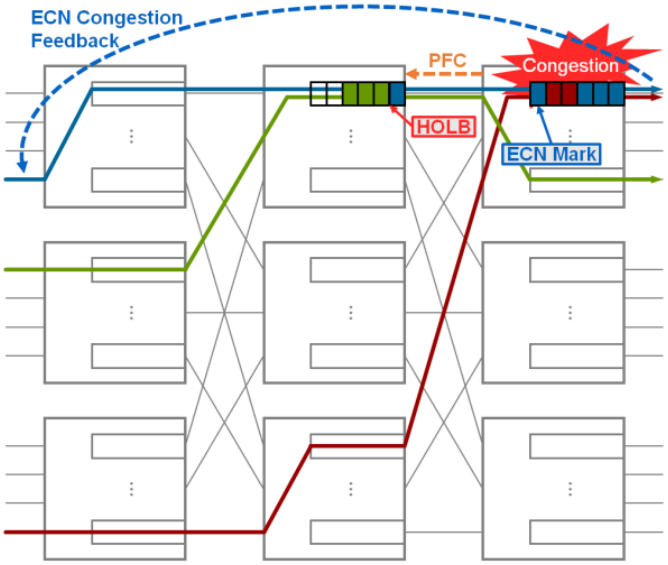

Figure 9 – PFC Head-of-Line Blocking

Head-of-line blocking can cause additional congestion upstream. Since PFC blocks all flows, even those destined to paths that are not currently congested, all flows must queue in the upstream switch. This queuing delay may in-turn create congestion in the next upstream switch. When input buffers in the upstream switch fill, PFC messages are sent further back into the network, creating more head-of-line blocking and more congestion. This is known as congestion spreading.

Incast congestion

Incast is a naturally occurring phenomenon in highly parallelized cloud applications and has been shown to be responsible for the majority of packet loss in the data center [10]. Iterative divide and conquer strategies with periodic synchronization require a significant amount of many-to-one communication. Incast congestion occurs at the ToR switch where the node that multiple parties are synchronizing with is connected. Multiple inputs are simultaneously directed to a single output, creating an oversubscription scenario. This type of congestion is an attribute of the application design more than an issue with the network. However, the network can assist by eliminating packet loss both locally within the switch and across the fabric. Current data center network equipment simply reacts to incast using a combination of ECN, PFC and smart buffer management in an attempt to minimize packet loss.

Technologies for the Future

What if there was no loss in the data center network? None whatsoever! No packet loss, no latency loss and no throughput loss. We would be able to parallelize applications and datasets as needed to meet the real-time interactive latencies required to meld the intelligence in the cloud with our own human lives. We would be able to create new and unique user experiences from unbounded information.

To create such an environment, we have to mitigate congestion in the network. Not simply cope with it, like today's networking technologies, but mitigate the effects and create a lossless network. The following new and proposed technologies are aiming to do just that – progressing towards the lossless data center network for the future.

Virtual Input Queuing

The lossless network must begin within the switch itself. There are many different silicon and system architectures available to build a switch, but without coordination between the ingress and egress ports it is difficult to create a lossless environment. Figure 10 shows how incast can create packet loss within a switch if there is no coordination between the ingress and egress ports. PFC is typically implemented on the ingress queues of a switch. When those queues back-up because the egress port is full, they will eventually trigger PFC to the upstream neighbor. However, in the incast scenario without ingress-egress coordination, it is possible that the egress queue will overflow before all the ingress queues have reached their threshold to generate PFC.

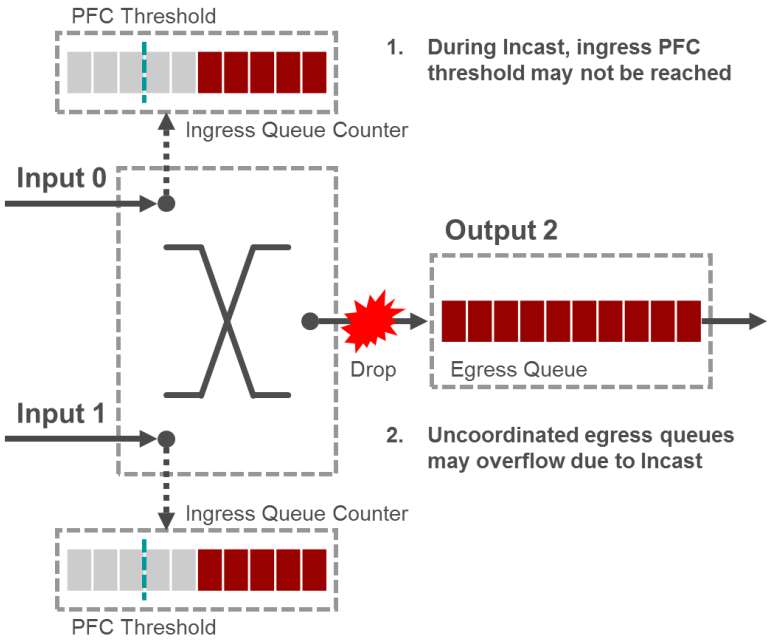

Figure 10 – Switch Packet Loss

Virtual Input Queuing (VIQ) is an approach to coordinate the resources available on an egress port with the demands of an ingress port to deliver data. With VIQ, the egress port informs the ingress ports of its buffer availability to avoid internal switch transfers that will result in packet loss. Packets can naturally back-up in the ingress port and PFC can be applied appropriately if needed. VIQ can be modeled as having a dedicated queue at egress for each ingress port and as a consequence, fair share scheduling can be applied to traffic leaving the switch.

VIQ has the advantage of avoiding congestion induced packet loss within the switch itself. In addition, VIQ modelling can allow traffic to exit the switch in a fair and orderly manner to help maintain the foundation of the lossless data center.

Dynamic Virtual Lanes

Intermittent congestion within the network can be caused by the unfortunate mix of flows across the fabric. A small number of long duration elephant flows can align in such a way to create queuing delays for the larger number of short, but critical mice flows. The delay in the control loop for end-to-end congestion control of the elephant flows cannot prevent PFC flow control from being invoked. When buffers fill and

eventual flow-control kicks in, mice flows can be blocked by the unfortunate burst alignment of elephant flows. If PFC flow control is not being used, packet loss on short mice flows can result in full retransmission timeouts, significantly penalizing the latency of mice flows used for control and synchronization within the parallel application.

Dynamic Virtual Lanes (DVL) is an implementation of Congestion Isolation (CI) that eliminates head-of-line blocking caused by the over-use of PFC. DVL identifies the flows that are causing congestion, isolates them to a separate traffic class and then signals to the upstream neighbor to do the same. DVL effectively moves the congestion out of the way, temporarily, while the end-to-end control loop has time to take effect.

Figure 11 shows the operation of DVL. When flows unfortunately collide at the egress port of a switch, congestion is detected and the offending flows are identified. Subsequent packets from the offending flows are routed through a dedicated congested flow queue (i.e. they are effectively moved out of the way). Once the congested flow queue reaches a threshold, DVL signals to the upstream switch using a Congestion Isolation Packet (CIP) that contains enough information for the upstream switch to identify the same congested flow. The upstream switch also isolates the same flow and begins to monitor the depth of the congested flow queue. The packets in the congested flow queue are drained at a lower priority than other non-congested queues, so when congestion persists, the congested flow queue may fill. A switch implementing DVL may utilize Virtual Input Queuing (VIQ) to coordinate the congested flow queue with the ingress port. When the congested flow queue fills, the ingress port can issue PFC to avoid packet loss. Flow control is only blocking the congested flow queues and other well behaved mice and elephant flows are free to traverse the fabric via non-congested queues.

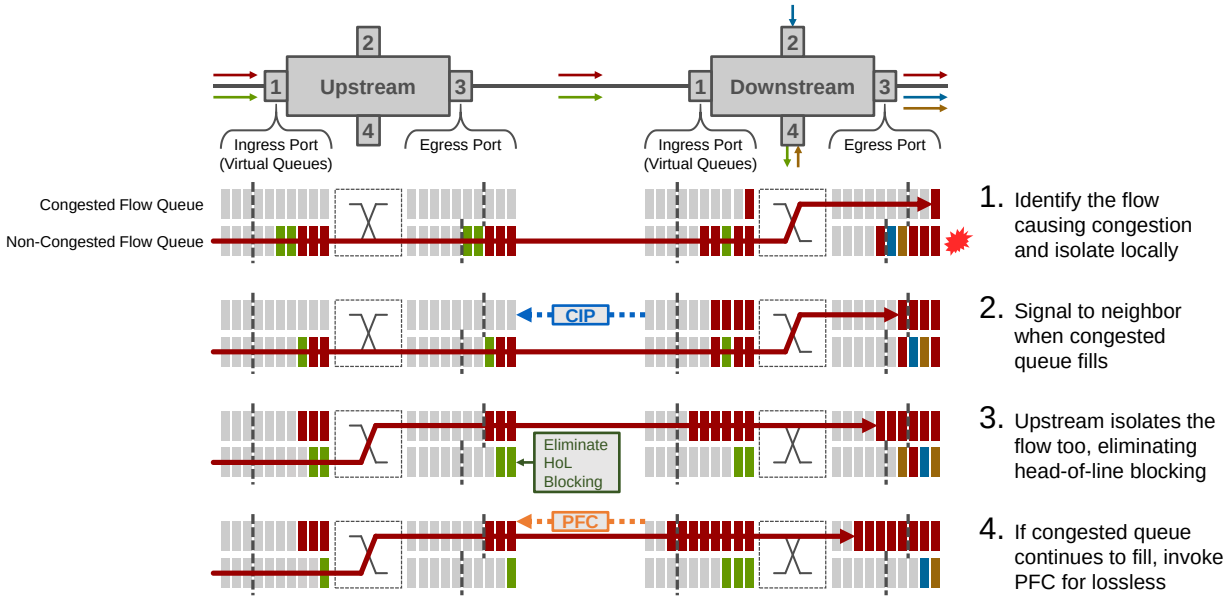

Figure 11 – Dynamic Virtual Lanes

The advantage of DVL is that latency can be reduced for critical control flows and packet loss can be eliminated without head-of-line blocking or congestion spreading. If PFC is needed, it typically is only needed on the congested flow queue. The offending flows will be delayed enough to allow end-to-end congestion control, such as ECN, to take effect. The temporary bursts of the offending flows are absorbed

by the coordinated congested flow queues between peers in the fabric. Simulation results have shown that DVL significantly reduces flow completion times by dramatically reducing the use of PFC in the network.

Load-Aware Packet Spraying

Load balancing network traffic is a technique to avoid in-network congestion; however, ineffective approaches can actually do the opposite. Figure 12 shows the design space for load-balancing technologies. Centralized approaches have difficulty scaling and meeting real-time latency requirements. Network wide congestion awareness provides more information than local in-switch decisions. The granularity of load balancing has trade-offs between the uniformity of the distribution and complexity associated with assuring data is delivered in its original order. From this design space we choose Load-Aware Packet Spraying (LPS) – a distributed, packet level, congestion aware approach that achieves fine grain load balancing without causing packets to be delivered out-of-order.

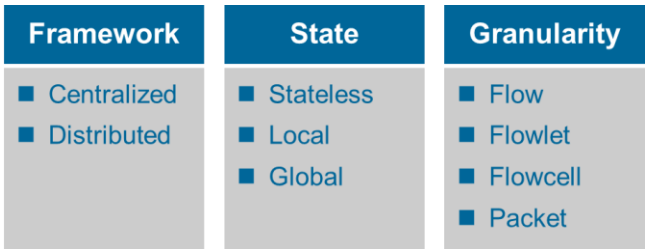

Figure 12 – Load Balancing Design Space

With LPS, packets between two ToR switches are sprayed across the multiple paths according to the degree of congestion measured on those paths. In Layer-3 virtualized environments, flows between two ToR switches can be identified by the virtualization encapsulation. LPS includes a sequence number in this encapsulation to allow the destination ToR to reorder packets back into their original sequence. Since a destination ToR may be receiving flows from many source ToRs at the same time, there needs to be a reordering queue for each ToR in the Clos network. The LPS source ToR maintains an indicator of congestion along the path to other destination ToR switches. This indicator can be determined by any number of congestion measurement techniques. The source ToR uses the congestion indicator to determine how to spray packets across the multiple paths – lighter loaded paths will take more packets than congested paths, which may be skipped entirely.

The advantages of LPS over current ECMP load balancing are threefold. LPS avoids elephant flow collisions because it distributes traffic with fine granularity at the packet level. LPS can rapidly adapt to network status changes because it is congestion-aware. Finally, LPS is more parallel than ECMP and can reduce flow completion times in lightly loaded networks by distributing a single flow across multiple parallel paths at the same time.

Push and Pull Hybrid Scheduling

While incast congestion is often an artifact of the parallel application design, the network can assist in eliminating packet loss at the destination by scheduling traffic delivery when it would otherwise be lost. In the traditional approach, a source ToR forwards packets to a destination ToR without considering the processing capacity of destination ToR. This works well when the network is lightly loaded and no congestion exists, however, once incast congestion appears at the destination ToR, delays increase and buffers overflow, throughput is lost and latency rises. Pulling data from the source is an alternative, but it requires an

extra round-trip delay for a request / grant message exchange before transferring data. In the pull scenario, the source ToR issues a request to send, and the destination ToR schedules a grant response when resources are available to receive the transfer. The pull approach incurs a request-grant RTT delay, but during incast, the transfers can be scheduled in such a way to avoid queuing delays and packet loss entirely.

The Push and Pull Hybrid (PPH) approach achieves the best of both approaches by monitoring the congestion between the source and destination ToR. As seen in Figure 13, if the network load is light, the push approach is used. If the network load is high, the pull approach is used. The source ToR measures the congestion to the destination ToR in order to decide which mode to use.

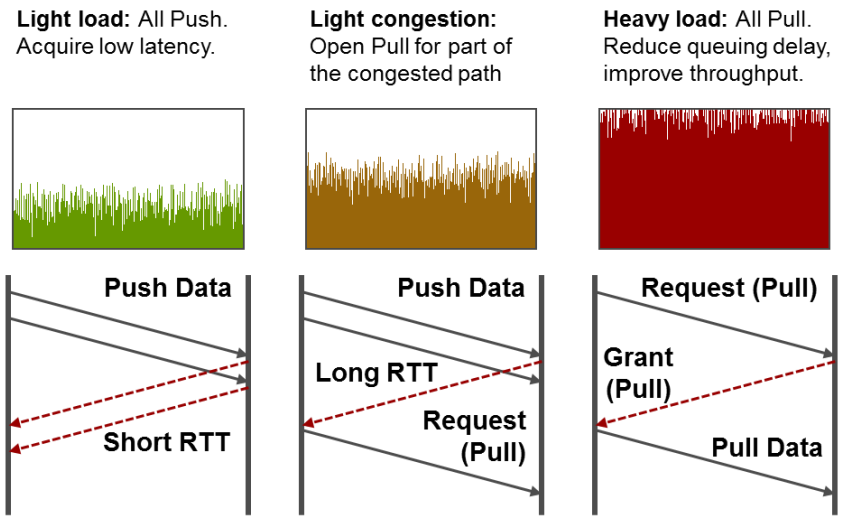

Figure 13 – Push and Pull Hybrid

The advantage of PPH is that it can eliminate congestion and packet lost due to incast oversubscription. Current data center networks are unable to avoid packet loss caused by incast congestion without applying a heavy hammer (PFC) that ripples across the network, spreading congestion. With PPH, the traditional push approach is used when possible, but as soon as incast congestion exists, the traffic is scheduled to match the available resources.

Summary

The demands on the data center network will be great. Highly parallelized applications and online services must deliver instantaneous response with no delay. There is simply no time for loss in the network due to congestion. In this paper we have introduced Load-Aware Packet Spraying, Dynamic Virtual Lanes, Push and Pull Hybrid scheduling and Virtual Input Queues. Each of these technologies is designed to mitigate congestion in the data center. Load-Aware Packet Spraying provides fine grain load balancing that is congestion aware to avoid the problem of large flow collisions due to simple ECMP load balancing. Dynamic Virtual Lanes reduces the use of PFC in the network and eliminates head-of-line-blocking by moving the flows that are creating congestion to a separate traffic class. Hybrid Push and Pull scheduling eliminates loss due to incast without sacrificing latency in a lightly loaded network. Packets are scheduled for delivery across the fabric with end-to-end congestion awareness. Virtual Input Queues avoid packet loss due to congestion within the switch itself by coordinating ingress and egress queue handling. These new innovations work together to eliminate loss in the cloud data center network.

References

[1] R. Kurzweil, The Singularity Is Near: When Humans Transcend Biology, Penguin Publishing Group, 2005.

[2] R. Kapoor, G. Porter, M. Tewari, G. M. Voelker and A. Vahdat, "Chronos: predictable low latency for data center applications," in *Proceedings of the Third ACM Symposium on Cloud Computing*, San Jose, California, 2012.

[3] V. Jalaparti, P. Bodik, S. Kandula, I. Menache, M. Rybalkin and C. Yan, "Speeding up distributed request-response workflows," in *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, Hong Kong, China, 2013.

[4] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang and A. Y. Ng, "Large scale distributed deep networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, Lake Tahoe, Nevada, 2012.

[5] L. Mai, C. Hong and P. Costa, "Optimizing network performance in distributed machine learning," in *Proceedings of the 7th USENIX Conference on Hot Topics in Cloud Computing*, Santa Clara, CA, 2015.

[6] NVM Express, "NVM Express® Moves Into The Future," [Online]. Available: http://www.nvmexpress.org/wp-content/uploads/NVMe_Over_Fabrics.pdf. [Accessed 2 11 2017].

[7] Cisco; EMC; Intel, "The Performance Impact of NVMe and NVMe over Fabrics," 13 November 2014. [Online]. Available: https://www.snia.org/sites/default/files/NVMe_Webcast_Slides_Final.1.pdf. [Accessed 28 10 2017].

[8] IETF, "Data Center TCP (DCTCP): TCP Congestion Control for Data Centers," 17 10 2017. [Online]. Available: https://datatracker.ietf.org/doc/rfc8257/. [Accessed 1 11 2017].

[9] *IEEE Std. 802.1Q-2014, Clause 36, Priority-based Flow Control*, IEEE, 2014.

[10] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano, A. Kanagala, J. Provost, J. Simmons, E. Tanda, J. Wanderer, U. Hölzle, S. Stuart and A. Vahdat, "Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, London, United Kingdom, 2015.