Stardust: Divide and Conquer in the Data Center Network

Noa Zilberman
University of Cambridge

Gabi Bracha
Broadcom

Golan Schzukin
Broadcom

Abstract

Building scalable data centers, and network devices that fit within these data centers, has become increasingly hard. With modern switches pushing at the boundary of manufacturing feasibility, being able to build suitable, and scalable network fabrics becomes of critical importance. We introduce Stardust, a fabric architecture for data center scale networks, inspired by network-switch systems. Stardust combines packet switches at the edge and disaggregated cell switches at the network fabric, using scheduled traffic. Stardust is a distributed solution that attends to the scale limitations of network-switch design, while also offering improved performance and power savings compared with traditional solutions. With ever-increasing networking requirements, Stardust predicts the elimination of packet switches, replaced by cell switches in the network, and smart network hardware at the hosts.

1 Introduction

For the last ten years, cloud computing has relentlessly grown in size [28]. Nowadays, data centers can host tens of thousands [75] of servers or more. The complexity of the data center network (DCN) has grown together with the scaling of data centers. While scale has a direct effect on the bisection bandwidth, it also affects latency, congestion, manageability and reliability. To cope with these demands, network switches have grown in capacity by more than two orders of magnitude in less than two decades [85].

The computing community faced the end of Dennard's scaling [35] and the slowdown of Moore's law [60] over a decade ago, prompting a move to multi-core and many-core CPU design [68]. Similar challenges are faced by the networking community today. In this paper, we discuss limitations on network device scalability, and assert that in order to continue to scale DCN requirements, data center network devices need to be significantly simplified. We introduce Stardust, a DCN architecture based on the implementation of network-switch systems on a data center scale.

In Stardust, we divide the network into two classes of devices: top-of-rack (ToR) devices maintain classic packet-switch functionality, while any other device in the network is a simple and efficient cell switch. We refer to the part of the network created by these simple switches as the *network fabric*. Devices within the network fabric do not require complex header processing and large lookup tables, have minimal queueing and reduced network interface overheads. Their data path requires minimum clock frequency, regardless of packet size. Creating a network fabric that is made completely of a single type of a simple switch is a radical move from common approaches, whereby each network device is a full-fledged, autonomous packet switch. To conquer the DCN scalability and performance requirements, we build upon a number of insights, drawn from silicon through the network level:

- The DCN can be compared to a large-scale network-switch system, where complex routing decisions are taken at the edge, and simple forwarding is applied within the fabric. By using only basic routing mechanisms within the core of the DCN, significant network-switch resources can be saved, while maintaining functionality.
- The increase in port rates, utilizing an aggregation of N serial links per port as a norm, limits the scalability of the DCN. By using discrete, rather than aggregated links, the scale of Fat-tree networks can improve by $O(N^2)$.
- Unnecessary packet transmissions can eventually lead to packet loss. Credit-based egress scheduling can prevent packet loss due to congestion, and increase fairness between competing flows.
- Congestion can happen within underutilized networks due to imperfect balancing of flows across available paths. By evenly distributing traffic across all links, optimum load balancing can be achieved.
- Network-switch silicon is over-designed: it is under-utilized for small packet sizes, and packet sizes not aligned to the internal data path. By batching packets together, and chopping them into cells that perfectly match internal data-path width, maximum data path utilization can be achieved.

Stardust has no sense of flows, and it does not require route installation. The network fabric is completely distributed, with no need for complex software-defined networks (SDN) management, central control or scheduling. Stardust is also protocol agnostic, reducing silicon level requirements by 33%. The resulting network provides full bisection bandwidth. It is lossless, load balanced and provides significant power and cost savings.

Switch-silicon vendors have produced, and shipped in volume, chipsets such as we exploit in Stardust (e.g., [20, 61]); we do not claim these designs as a contribution. Similarly, switch vendors have produced single-system switches that internally use such chipsets to produce the illusion of a single many-port packet switch (e.g., [14, 43, 26]). Our contribution, in this paper, is to architect and implement our approach on a data center scale, using such commodity chipsets.

To summarize, we make the following contributions:
• We explore performance and scalability limitations in current DCN network devices (§2), motivating the need for simpler network devices.
• We introduce Stardust, an architecture offering a better approach to realizing DCN (§3,§4), and discuss its advantages on a data center scale (§5).
• We evaluate Stardust (§6): experimental measurements on *O(1K)s* of port environments, event simulations of *O(10K)s* port systems and an analysis of large scale *O(100K)s* port networks, illustrating the advantages of Stardust.
• We propose a future vision of DCNs (§8), where ToR functionality is reduced to fit within network interface cards, with ToR switches turned into cell switches.

2 Motivating Observations

The design of scalable network devices has hit a wall. If a decade ago the main question was *"how can we implement this new feature?"*, now the question is *"is this feature worth implementing given the design constraints?"*. We discuss three types of limitations to network device scalability: resources, I/O, and performance. Our observations are orthogonal, providing trajectories to improving scalability.

2.1 The Resource Wall

Every network device is limited in its die size and its power consumption. Chip architects need to balance the design to avoid exceeding these resources limitations; die size is not just a function of cost, but also of manufacturing and operating feasibility, and reliability.

Specializations and simplifications enabled DCNs to evolve considerably from being an *Internet-in-miniature* using complex router and switch architectures. The data center's unique administrative environment allows Stardust to present a *dispersed switch* design, eliminating the need to support in each network hop a full suite of protocols, traffic-control, or full interface flexibility.

Previous large switches have been custom-made machine-room-wide HPC interconnects, or multi-chassis systems interconnecting interface line-cards and control processors. Such platforms, (e.g., [27, 43]) implement advanced packet operations on the line-cards, interconnected using a simple fabric consisting of minimal packet queuing or processing. In DCN, the ToR switches take the role of the line-cards, while the "interconnect fabric" is the spine and leaf switches interconnecting all ToR devices. This model of the DCN equally applies to Stardust.

Fig. 1 illustrates the difference between the two approaches: In both cases, the first switch (the ToR) provides packet processing, queueing, and full network interfaces. However, in the typical data center network approach, traversing a Fat-tree will include going through all functions (ingress and egress processing, queueing, scheduling,...) in each and every hop. In contrast, with Stardust, scaled up from the single system approach, these functions will be used only at edge devices. By eliminating stages, logic and memory, consuming considerable silicon area [19], significant resources can be saved. While the I/O remains the same, the network interface is considerably smaller; there is no need to support multiple standards (e.g., 100GE, 400GE), and a single MAC functionality is sufficient. Stardust benefits include reducing both network-wide latency and network-management complexity as it practically presents as a single (large) switch. Shallow buffering is not entirely eliminated (see §6).

On the order of microseconds [16, 69] rather than milliseconds, the reduced fabric latency of a Stardust DCN behaves as a single hop (albeit across a larger *backplane* fabric). This allows applications to continue operating unchanged

Observation: *Significant resources can be saved by simplifying the network fabric and adopting a scaled-up single-system approach.*

2.2 The I/O Wall

The maximum number of network interfaces on network devices has grown by an order of magnitude over the last decade, climbing from 24 ports [31] to 260 ports [64]. Device packages are big, $55mm \times 55mm$ [79] or more, bigger than high-end CPUs [44]. Such big packages raise concerns about manufacturing feasibility, cost and reliability. It is unlikely the I/O number will continue to scale at the same rate.

The second part of the I/O problem is that a network port is not necessarily a single physical link. A *Link Bundle (l)* is the number of serial links connecting two hops. For example, connecting two switches using a port of 100GE CAUI4, four lanes at 25Gbps each, is a *link bundle of four*. Link bundling is a common practice in high-speed interfaces, used to increase the bandwidth of a logical port through the multiplexing of information from multiple physical links. This practice overcomes the signaling and physical limitations of the media (e.g., copper), and also applies to integrated photonics devices for network switches [17].

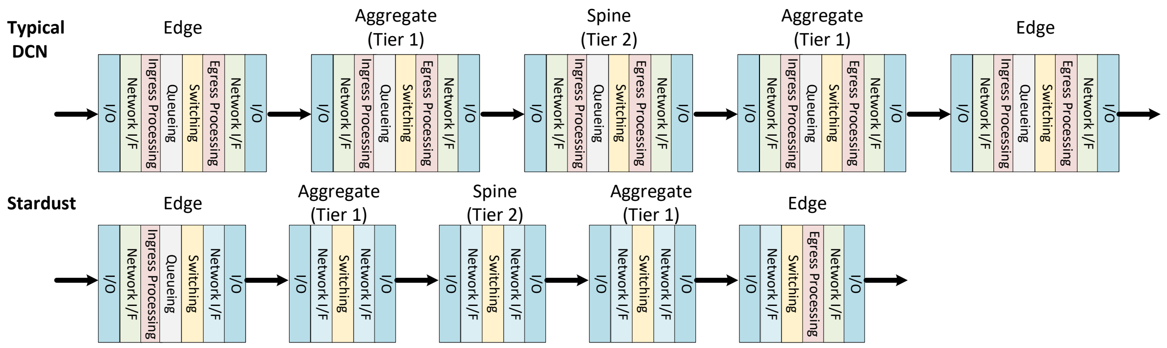
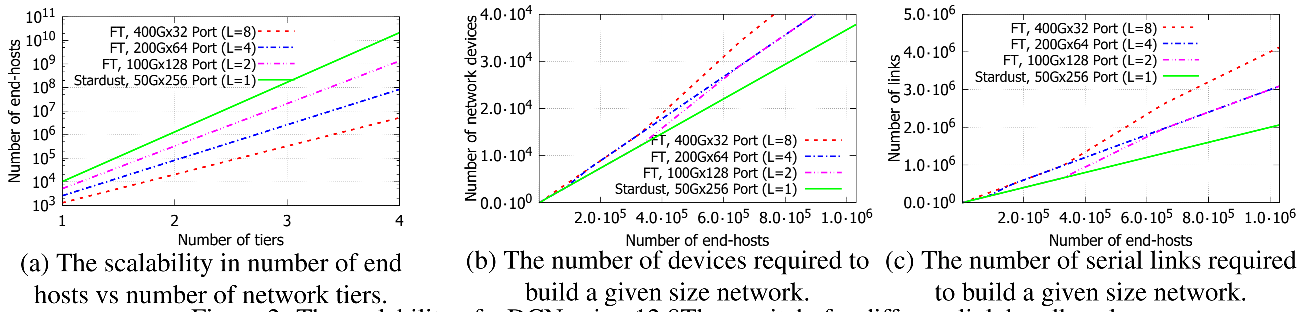


Figure 1: Traversing through a Fat-tree network, in a typical data center network (top) and in Stardust (bottom). In a network with n tiers, a typical DCN requires $n \times (\mathit{Ingress\ processing} + \mathit{Queueing} + \mathit{Scheduling} + \mathit{Egress\ Processing})$. Stardust requires just $1 \times (\mathit{Ingress\ processing} + \mathit{Queueing} + \mathit{Scheduling} + \mathit{Egress\ Processing})$.



(a) The scalability in number of end hosts vs number of network tiers.

(b) The number of devices required to build a given size network.

(c) The number of serial links required to build a given size network.

Figure 2: The scalability of a DCN using 12.8Tbps switch, for different link bundle values.

For the rest of this paper, we refer to each switch port as a link bundle, and a switch radix indicates the number of ports (link-bundles) in a switch. We use a Fat-tree topology to demonstrate the effect of switch radix and link bundling on DCNs scalability, and provide in Appendix A the mathematical justification. As each end host connects to a single ToR, link bundling from the host has no effect on scalability.

Assume a network that consists of edge network devices (e.g., ToR) and a network fabric connecting all edge devices. The number of layers within the network fabric is the number of *Tiers*. A network built from only ToR and aggregation devices is a 1-Tier network, whereas the common Fat-tree Edge-Leaf-Spine structure is a 2-Tier network. A network can be extended to include more edge devices by introducing additional tiers. Fig. 2(a) demonstrates the size of the DCN required to connect a given number of end hosts, as a function of the link-bundle. In this figure, we assume a network device of 12.8Tbps, using different link bundling configurations, ranging from $256 \times 50Gbps$ $(l = 1)$ to $32 \times 400Gbps$ $(l = 8)$, and assuming 40 servers connected to each edge device using $100Gbps$ $(l = 2)$ links. A link bundle of one enables a 1-Tier network of over ten thousand servers, whereas a 1-Tier network with a link bundle of eight is limited to an eighth of this number of hosts. For a 2-Tier network, a link bundle of eight allows connecting only 20K hosts, compared with $\times 64$ the number of hosts using a link bundle of one. The link bundle affects not just the overall scalability of the network, but also the number of network devices and links required to build a given size network (Fig. 2). §5.1 and Appendix A expand on link bundling.

Observation: *For a given switch bandwidth, a link bundle of one will allow the optimum number of switches in a DCN.*

2.3 The Performance Wall

Network-silicon devices support throughput in the order of 12.8Tbps [21]. This is equivalent to $32 \times 400Gbps$ ports or 19.05Gpps (19.05 billion packets per second) for 64B packets, and 5.8Gpps for 256B packets. The clock frequency of switch silicon is in the order of 1GHz, meaning that even if a new packet can be processed every clock cycle, between 6 (for 256B) and 20 (for 64B) packets need to be processed in parallel. Even for 1500B packets, more than a single packet needs to be processed every clock cycle. To address this problem, network silicon today either does not support full line rate for all packet sizes, or implements multiple data-plane cores within each device, connected through an on-chip switching module or a shared memory [39, 62]. See Appendix B for a detailed analysis.

Observation: *To support full line rate for all packet sizes, network devices need to process multiple packets each and every clock cycle.*

In Fig. 3 we explore the scaling requirements for a switch's pipeline. We assume two 12.8Tbps devices: one device implements standard packet switching, and the other device (Stardust) optimally packs data into the pipeline, in units that are equal to its data path's width. The internal data path frequency is 1GHz, and the data path is 256B wide[1]. Fig. 3 shows the number of parallel buses, or processing

[1]Data path width is upper-bounded by timing and resources limitations, and lower-bounded by performance requirements.

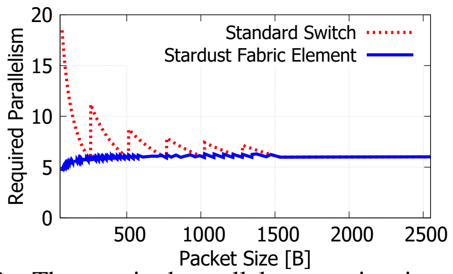


Figure 3: The required parallel processing in a standard switch and in a Stardust Fabric Element. We assume 12.8Tbps switch, 256B wide bus, 1GHz data path frequency.

cores, required to support the device's 12.8Tbps. For small packets, a design optimally packing data (Stardust) outperforms a packet-based design by a factor of $\times 4$. Packing data provides 41% improvement for 513B packets, and 18% for 1025B packets. Increasing the data path width eases the requirements for large packets, but not for small ones.

Packets smaller than the internal data path width (which can be over 50% of the traffic, assuming a 256B wide bus [74]), benefit from data packing even more, as the on-wire inter packet gap (IPG) and preamble are applied only once per packed unit, rather than once per packet, increasing the overall packet rate. From a silicon design standpoint, expecting standard switches within the DCN fabric to process all packets at wire rate requires a huge amount of over design for the task performed, with substantial penalties.

Observation: *Packing data to optimally fit data path width achieves higher throughput, for all packet sizes, at lower clock frequencies. This simplifies switch design and enables better scalability over time.*

3 Architecture

Stardust is a new architectural paradigm for DCNs, addressing the resource, I/O and performance limitations discussed in §2. Stardust scales up the switch-system to the DCN.

In order to support a scalable network, we define two types of devices: the Fabric Adapter device, and the Fabric Element device. The Fabric Adapter device, such as StrataDNX Jericho [22], plays a similar role to a ToR device. The Fabric Element device, such as StrataDNX Ramon [23], is used as the building block of the network fabric. The Fabric Adapter and Fabric Element used in the architecture differ from the commonly used packet switches in at least one important aspect: the Fabric Element is not a packet switch but a simple cell switch, and the Fabric Adapter takes every incoming packet and chops it into data units, which we refer to as cells. Cells are still "packet" units, but with payload and header sizes optimized to fit the Fabric Element pipeline.

3.1 Constructing a Stardust Based Network

The Stardust architecture is Clos based [30]. Fabric Adapters are used as the first and last stage devices, whereas the middle stages are composed of Fabric Elements. For the rest of this paper we limit the discussion to Fat-tree topologies [51], a special case of a folded Clos topology.

The building of a network starts by connecting host machines, (e.g., servers) to a Fabric Adapter device that acts as a ToR switch. Fabric Elements are used in the aggregation and spine layers. The number of uplink connections from the Fabric Adapter to the aggregation layer of the network fabric depends on the desired uplink bandwidth and over-subscription ratio. In Stardust every physical link is an independent entity and is not bundled.

The network fabric is made only of Fabric Elements. Each Fabric Element has k full duplex links. In the aggregation layer, half of the links are connected to the Fabric Adapter and half of the links are connected to a spine layer. In the spine layer, all the Fabric Element's k links are connected to the aggregation layer.

We define the fabric speed up, fs, as the ratio between links' capacity to the network fabric and links' capacity to the hosts. $1/fs$ is the link *utilization*. A network fabric may be under-subscribed with *utilization* < 1 or fully utilized, from any device to any device, though some redundancy is typically applied. Long-term over-subscription from the hosts to the Fabric Adapter is handled as in any ToR, i.e., packets will be dropped in the Fabric Adapter. Short-term over-subscription is absorbed by the Fabric Adapter's buffers. Similarly, over-subscription is allowed between tiers in the network fabric.

3.2 Dynamic Cell Forwarding

Packets arriving from the host to a Fabric Adapter are parsed, and the destination is identified. Every destination is mapped to a destination Fabric Adapter and port. Each Fabric Adapter holds a reachability table indicating which links can be used to reach the destination. The Fabric Adapter collects multiple packets and chops them into bounded-size (e.g., 256B) cells. The cells hold a small header including the destination and a sequence number that allows reassembling cells into packets.

A pivotal idea in Stardust is dynamic cell forwarding: each cell is sent over a different link, while the Fabric Adapter arbitrates over all the links in the reachability table marked as leading to the destination. At each Fabric Element stage, the same process is repeated by load balancing among all available paths converging on the destination Fabric Adapter. This dynamic cell forwarding is a radical departure from traditional static routing architectures, where a flow (or flowlet) is bound to a specific path according to it headers, leading to complex issues of provisioning, congestion avoidance, failure recovery, routing table configuration and more. Load balancing on the links is nearly perfect and enables an optimal use of fabric resources. The fabric behaves as a uniform transport media, agnostic to incoming traffic patterns, or to flow granularities. Dynamic forwarding may introduce out-of-order fabric traversal. However, the load-balancing creates a limit on the extent of out-of-order mismatching, and reordering is managed at the destination Fabric Adapter (§4).

3.3 Buffering And Scheduling

Stardust is based on the combination of packet buffering at the edge and a distributed scheduled fabric. The architecture uses virtual output queues (VOQs) to queue packets arriving to the Fabric Adapter. Each destination port (and priority) has an assigned VOQ. The Fabric Adapter uses a buffering memory, in the order of megabytes to gigabytes per Fabric Adapter, storing the queued packets. Empty VOQs do not consume buffering resources.

Distributed traffic scheduling is used in Fabric Adapter's Egress. Non-empty VOQs request from the destination port permission to send traffic, and the port scheduler at the destination Fabric Adapter is responsible for generating credits without exceeding the port's capacity, as it has a view of all of the VOQs toward its ports. Stardust supports multicast and broadcast, but this support is beyond this paper's scope.

A credit entitles a VOQ to release an amount of data, (e.g., 4KB) to the fabric. The VOQ dequeues packets up to the credit size. The amount of surplus data is stored for later accounting. The next VOQ is selected by the order of credit arrival and a priority. The ingress packet buffer and sufficient egress memory enable absorbing long bursts of data at the ingress and momentary bursts of data at the egress.

3.4 Packet Packing

Using cells can be quite wasteful: sending packets that are just one byte bigger than a cell size can lead to 50% waste of throughput. *Packet packing* avoids such waste, effectively increasing the packet processing rate, but operating on packed packets rather than on individual ones (Appendix B). When a VOQ receives a credit to send packets, it chops the packets in the queue into cells while treating the entire burst of data as a unit. As a consequence, a cell may include multiple packets or multiple packet fragments (Fig. 4). Packet packing is feasible only within the same VOQ.

4 Device Architecture

4.1 Fabric Adapter

The Fabric Adapter (Jericho) is the edge unit interconnecting hosts with the network fabric. The Fabric Adapter resembles a ToR switch, providing (programmable) header-processing, scheduling and switching functionality. The size of its tables resembles a ToR. However, it has additional functionalities:

• Chopping incoming packets from the hosts into cells, and sending them to the network fabric
• Reassembling cells arriving from the fabric into packets
• Providing scheduling services by sending credits

Fig. 5 depicts an architecture of a Fabric Adapter device. On the Fabric Adapter ingress, packets arrive from the host through multiple interfaces, (e.g., 40GE, 100GE). The packet header is parsed and packets are stored in VOQs, according to the destination Fabric Adapter and port. The number of VOQs is determined by the total number of downlink ports on Fabric Adapters and the number of traffic classes (rather than the number of routable IP addresses).

Non-empty VOQs send special control messages to an egress scheduler at the destination Fabric Adapter. The egress scheduler is consequently aware of all VOQs toward its ports. It sends to the ingress Fabric Adapter a "credit", (e.g., 4KB). The total rate of credits matches the egress port's rate. The egress scheduler implements various QoS policies, typically a combination of round-robin, strict priority and weighted among VOQs of different Traffic Classes, allowing a flexible allocation of fabric and egress port capacities. While Jericho uses a proprietary scheduler, similar schedulers have been described before, e.g. in [56, 24]. To keep the egress buffer full, compensating for propagation and processing delays, the credit rate is set slightly above the egress port bandwidth (e.g., 2%) and slightly below the fabric speed-up, to avoid congestion (§ 6.2). When the egress buffer is close to full, the scheduler stops sending credits to the VOQs and resumes as packets are drained.

Packets are dequeued from the ingress VOQ when a credit arrives. The VOQ dequeues a number of packets, matching the credit size. Packets are allocated to cells of a limited size (e.g. up to 256B), distributed across multiple links toward multiple Fabric Elements. The cell size matches the width of the Fabric Element pipeline, and therefore fabric performance is minimally affected by packet size distribution (§6).

At the network's egress, cells arrive from Fabric Elements over multiple links and are reassembled into packets. Cells and packets can arrive out of order, and many hardware-based solutions exist (e.g., [33, 15, 32]). However, as credits are spread evenly over time, the number of concurrent interleaved packets at the egress reassembly is bound by Fabric Element queue size and does not scale with network size, or the number of VOQs. If a packet reassembly timer expires, e.g., due to a link error, the packet is discarded.

The minimum credit size is set by the Fabric Adapter's output bandwidth and the credit generation rate of its scheduler. For example, for a 10Tbps Fabric Adapter, using 1GHz clock frequency and generating a credit every two clocks, the minimum credit size will be $10Tbps/(1GHz/2) = 2000B$. The maximum credit size dictates the amount of memory required at the devices. Consider a flow control to a port: the egress memory will need to absorb the data in flight from source Fabric Adapters, which is a function of credit size and latency across the network (from last credit generated to last bit of data arriving). To minimize the amount of Fabric Adapter's egress memory, credit size should be at the same scale of the minimum credit size, as we validate (§6.2).

4.2 Fabric Element

The network fabric is composed only of Fabric Elements (Ramon). Fabric Elements connect to either Fabric Adapters

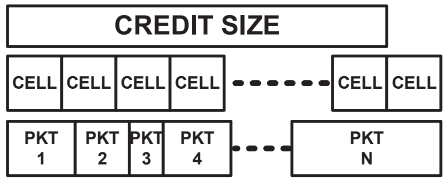


Figure 4: The relations between packets, cells and credits.

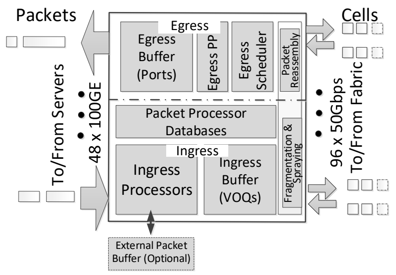


Figure 5: An architecture of a Fabric Adapter device

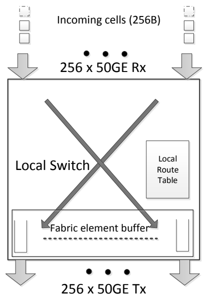


Figure 6: An architecture of a Fabric Element device.

or other Fabric Elements. A Fabric Element handles only fixed size cells, and does not parse packets, while providing bare minimum buffering and scheduling. A possible implementation of a Fabric Element device is shown in Fig. 6.

The Fabric Element is a radical simplification of DCN switches. It eliminates the logic and large tables associated with protocols. There is also no need for complex software configuring routing tables, handling bandwidth provisioning or protection recovery. Additional logic handling congestion avoidance and queue management is redundant as well.

In Stardust, a cell is transmitted on a single link in its entirety (rather than over a link bundle) with a bounded cell size and a small header that indicates the destination Fabric Adapter. Packing packets into cells guarantees that only a very small fraction of the cells are smaller than the maximum cell size (e.g., 256B). Thus, the Fabric Element data path can be throughput optimized, regardless of packet size, only bounded by technology and resource constraints.

Every Fabric Element holds a simple forwarding table that relates every destination Fabric Adapter to an outgoing Fabric Element link. If a Fabric Adapter is accessible through multiple links, the Fabric Element load-balances the traffic among them. As the destination is a Fabric Adapter and not an end-host, the size of the table can be two orders of magnitude smaller than a typical routing table. Beyond this, the Fabric Element is essentially two $k \times k$ crossbars (e.g., 256×256), one for data cells and one for control messages.

The forwarding table is automatically maintained by hardware exchanging special reachability control messages, where each device, Fabric Adapter and Fabric Element, advertises itself to all directly connected network-fabric devices. The reachability messages are sent periodically. If no reachability messages are received on a link periodically, it is considered failed. Topology changes are incorporated in the forwarding tables of the entire fabric in order of hundreds of microseconds (see §5.9). This automatic setup in hardware of forwarding tables, and the relative low latency of the process, sets it apart from past proposals of extending the internal structure of a packet switch to DCN scale.

As cells from different sources may compete on an output link, shallow buffering is required within the device (§4.2.1). If the link's queue is a above a configured threshold, then Fabric-Congestion-Indication (FCI) bits are piggybacked on all cells. The FCI reaches the destination Fabric Adapter and throttles the credit rate. The shallow queuing within the Fabric Element guarantees low latency and jitter within cells' travel time of the network fabric. The shallow buffering within the fabric guarantees shallow buffering at the egress of the Fabric Adapter, and easier packet reassembly.

4.2.1 Fabric Element Queuing Analysis

The probability distribution of queue size per link is a vital characteristic of Stardust. First, it determines latency and jitter characteristics for traffic through the network fabric. Second, it determines the amount of memory the egress Fabric Adapter requires for absorbing in-flight data and packet reassembly. Third, the queue size statistics determine the maximal practical link utilization. Last, it affects the die size of the Fabric Element.

A discrete queuing model represents the Fabric Element queue behavior, as a function of link utilization. The time unit is "fabric cell time", which is the time it takes to transmit a cell on a serial link. Consider a link's queue at the last stage of a network fabric: on average, a cell will be added to a link with probability $1/fs$. However, the cells arrival process is bursty, as multiple Fabric Adapters may be sending cells simultaneously.

Cell arrival is bound by a Poisson arrival process with parameter $1/fs$. The discharge process is a constant one cell every "fabric cell time". With M/D/1 queuing model, the probability of queue build-up on a link of size N can be approximated by $o(fs^{-2N})$ (as corroborated in §6.2). The Poisson arrival model is a worst-case scenario, as it assumes an infinite number of Fabric Adapters, and ignores the distribution effect of credits. As a limited number of Fabric Adapters generates traffic, both the burstiness of the arrival process and queue size's tail probabilities decrease.

5 Stardust on a data center scale

5.1 Scaling, Tiers Reduction and Longevity

The Fabric Element utilizes a lean functionality, allowing it to pack more interfaces into the same area and enables higher radix devices. As Stardust eliminates the need for link bundling logic, then when, e.g., 400GE is used (link-bundle of 8), the n^{th} tier of a Stardust based network can support $\times 8^n$ more ToR devices than a typical DCN, while still being non-blocking.

Since the traffic from a source to a destination Fabric Adapter is load balanced over all upstream links, the network fabric behaves like a single large pipe whose bandwidth is the bisection bandwidth (for example, 12.8Tbps pipe for 256 links of 50Gbps). Consequently, the fabric is agnostic to port rate or flow rate.

Stardust supports a gradual growth of a DCN. A data center operator will typically start with a small instance of a network fabric, then expand it over time. When designing a Stardust fabric, it is not necessary to populate the entire fabric from the start. The auto-construction of reachability tables, combined with dynamic forwarding, enables starting with a partially populated fabric, and adding Fabric Elements over time within a live network.

Stardust scales from switch-systems to data centers with 10K's to 100K's of nodes as the Fabric Elements, queueing and reassembly processes depend on device radix and are independent of network scale. Scheduling scales with the amount of on-chip memory. The number of VOQs in Jericho [22], as well as in other high-end switches [13, 29], is in the order of 100K, and increasing the number of VOQs is feasible (see §C).

5.2 Push Fabric vs. Pull Fabric

The following example shows the fundamental difference between a network fabric based on a "standard" Ethernet switch and a Stardust scheduled fabric. Consider a network fabric as in Fig. 7. On a single device there are two 100GE ports, A and B. From one input device 100Gbps are injected toward A (marked red), and additional 100Gbps are injected toward B (marked green). Additional 100Gbps are injected towards A from a second device (marked red).

In a fabric based on standard Ethernet switches, both of A's 100Gbps flows are pushed into the fabric. Even with multi-path routing and load balancing, local congestion occurs in the middle stage switch and both A's and B's traffic is dropped. At the egress device only 66% of B traffic passes, despite not being congested. The problem is exacerbated if we consider Traffic Classes, where the throughput of the standard Ethernet switch based fabric may be half of Stardust (Appendix F).

Ethernet switch fabric drops are due to local optimization of the ToRs. The network fabric, made of autonomous Ethernet switches, makes local decisions using local congestion, oblivious to the-end-to-end traffic. Thus, congested ports can block non-congested ports. While on a longer time scale packet losses may be reduced by congestion control mechanisms (e.g. ECN marking), this is not the case for short bursts.

In Stardust, the egress port scheduler of B sends 100Gbps worth of credit toward B's flows, and the egress port scheduler of A sends 50Gbps worth of credits toward each of A's sources. These flows, distributed through the fabric, reach the egress, and transmit 100Gbps at B and 100Gbps at A. The surplus 100Gbps of A's traffic is stored at the ingress buffer, and from there it may be sent (if over subscription is momentary) or dropped (if persistent).

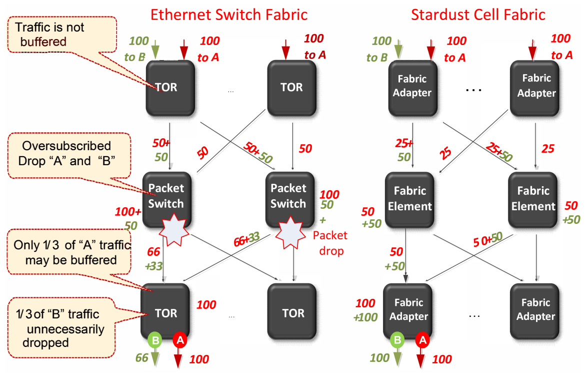


Figure 7: Packet drops in Ethernet switch fabric vs. Stardust.

5.3 Optimal Load Balancing

Network fabric architectures based on typical Ethernet switches use flow hashing to load balance traffic across a network. This has been shown to be highly inefficient, allowing only 40%-80% utilization of the bisection bandwidth [5].

Recent research [41] started investigating packet level load balancing, giving rise to more uniform distribution, lower queues, higher utilization [8], lower latency and jitter, and therefore a much shallower buffer. However, even with packet level load balancing, there is $\times 144$ disparity between the smallest 64B packet and the largest 9KB packet, leading to substantial variance in queue length, and hence latency.

Stardust load balances fixed size cells, effectively achieving a perfect fluid model. At each Fabric Adapter, each packet is segmented to fixed size cells that are distributed in a round robin manner across all links leading to the destination port. Thus, the same amount of data is sent down each link. There are only two aspects that cause insignificant variations in link loads. First, cells at the credit-worth tails may be shorter. Their relative amount is negligible as multiple packets from the same VOQ are accumulated up to a threshold (e.g., 4KB), before fragmentation. Second, recurrent synchronization between packet transmit times and the load balancing mechanism bias load on some links for some destinations. However, the round robin arbiter traverses the Fabric Element links in a random permutation order, that is replaced every few rounds. Thus, the probability of a persistent synchronization is negligible.

5.4 Effective Buffer and Incast Absorption

Stardust provides a large and distributed packet buffer, enabling it to absorb incast. These packet buffers may be managed by smart queue management (SQM). When a destination port is oversubscribed, the data accumulates only at source Fabric Adapters and the packet buffers available for storing data bursts are large. Even if the packet buffers are not sufficient, the source Fabric Adapter can avoid packet loss by sending flow control messages back to the host, as in a standard ToR.

Let us compare a "standard" unscheduled network fabric and a Stardust network, and assume a configuration of 128 ToR switches/ Fabric Adapters with $50Gbps \times 128$ ports, 64 switches/ Fabric Elements with $50Gbps \times 128$ ports, and 32MB packet buffer per device. Consider an incast event of 1MB from each ToR toward a single 50Gbps port (total of 128MB). In the Ethernet switch fabric, all traffic will reach the egress ToR, and whatever is not transmitted at 50Gbps, will fill the ToR's packet buffer or be dropped. This happens regardless of the load balancing granularity (flow, flowlet, packet). Activating PFC flow control at the destination may propagate and block the entire fabric. In Stardust, the incast traffic from all Fabric Adapters will be admitted to the fabric at an aggregate rate of 50Gbps. The rest (99%) accumulates in the ingress Fabric Adapters, occupying 0.99MB of each packet buffer. No packet is lost. The available packet buffer memory per destination is effectively $\times 128$ larger, distributed across all Fabric Adapters. The destination's egress scheduler distributes bandwidth (credits) to incast sources evenly, guaranteeing an even draining rate of the VOQs, and an optimal flow completion time of the incast event.

5.5 Lossless Transmission

The Fabric Element is lossless. As shown in §6, when undersubscribed, the probability of a queue reaching it maximal level is very low. The FCI reduces this probability further, as credit rate is reduced when a queue starts building. This control loop is activated with a very low probability, and affects only traffic heading to the congested destinations. When the network fabric is intentionally over-subscribed, the FCI mechanism is activated only when the fabric is actually oversubscribed with traffic. Queue overflow is further reduced using a shared queue-memory pool, permitting dynamic balancing between congested and non-congested links. As a last resort, the Fabric Element pipe can be paused to avoid dropping a cell. The probability of this event is infinitesimal, thus the net effect on the entire fabric throughput is negligible.

5.6 Latency and Jitter

The packet latency and jitter through the network fabric are governed by the queue size probability distribution within the Fabric Element. The latency is at the scale of 0.5-4 microseconds per hop. In a multi-tier network fabric, the latency is governed by fiber length as much as it is by the Fabric Element latency, as every 100m of fiber translates to a half microsecond propagation delay.

First packet latency depends on VOQ configuration. Receiving a credit will take about a microsecond, but a *low latency* VOQ starts transmitting immediately. We assume a limited aggregate bandwidth of all low latency VOQs, bounded by Egress Fabric Adapter and Fabric Element memory resources, else packets may be dropped (as in a ToR).

5.7 Traffic Pattern Agnosticism

A valuable property of Stardust fabric is its agnosticism to the traffic pattern at the edges (ToR/Fabric Adapter). The single parameter that affects the fabric performance and latency is the fabric speed-up fs[2]. If we consider the worst-case scenario for latency and drop probability, it is for many (e.g., 1000) sources sending traffic to a single output port. This is bound by the M/D/1 queuing model. Other scenarios would have a shorter-tailed queue size (hence latency) distribution. If the total bandwidth toward a port exceeds its bandwidth, as in incast, it does not matter; the egress distributes credits at the egress port's rate and no more, and excess traffic remains queued in the source Fabric Adapters, while the fabric experiences the same cell loads, and hence, will have the same or better queue size distributions than that of the M/D/1 queuing model.

5.8 Dynamic Routing

Routing within Stardust's fabric requires no routing protocol or SDN controller inputs, and is dynamic per cell. When a cell needs to be sent to a destination, there are multiple paths that can be taken. Cells are load balanced on all available paths, even if they are part of the same flow or packet. Consequently, a large set of paths is taken between every pair of source and destination Fabric Adapters. There is no set of fabric routes per destination that is saved in a memory: for each destination Fabric Adapter there is only the set of links outgoing from the current device that can be used. This forwarding database has a negligible size of *Number-of-Fabric-Adapters* entries of size *Number-of-Links*.

Each device periodically transmits its Fabric Adapter reachability table to its upstream neighbors, leading to an automatic update of routing tables. Any topology change due to link failure is transmitted from its source to the routing tables of all upstream devices, and affects their forwarding decision.

5.9 Improved Resilience, Self-Healing Fabric

The use of non-bundled links improves the resilience of the network. While for a 400GE port, a link failure means that $8 \times 50Gbps$ links go down, in the Stardust, a link failure affects only this single link. Therefore, the improvement in resilience equals the link bundling between stages.

The reachability messages exchange, used for building the destination's reachability table, is also used to establish a link's status and health, making the network self-healing and increasing its resilience. When a link is down, the reacha-

[2]And even then, as long as the fabric is $< 90\%$ utilized the dependence of latency distribution on fabric speed-up is very weak

bility tables in all devices are automatically updated, and the load balancing is adjusted to exclude dependencies on this link. The load balancing, starting at the first stage's Fabric Adapter, eliminates hot spots of congestions. Recovery time is determined by the rate allocated for reachability messages, i.e., interval between successive reachability cells. The recovery time is around hundreds of microseconds with negligible reachability cell overhead (Appendix E). While the values are configurable, assume for example, a Fabric Element reporting the reachability of 128 Fabric Adapters on a link every $1ns \times 10,000$ clocks. It takes the Fabric Element seven messages to report the status of a network connecting $32K$ hosts (40 hosts per Fabric Adapter). It takes 3 hops in a Fat-tree to reach the ingress Fabric Adapter from the last Fabric Element, amounting to $1ns \times 10000 \times 7 \times 3 = 210us$. Updating the status based on multiple (e.g, three) consecutive updates will amount to roughly $630us$. The mechanism precedes BFD [47] and, as implemented in a dedicated hardware, has a lower overhead.

5.10 Handling Failures

Load balancing cannot compensate for a link's failure; if the link is connected to a destination Fabric Adapter, the bandwidth capacity toward the destination is reduced. Resilience can be improved by adding redundant links. With traffic automatically spread across all links, and as there is no management overhead for adding links, contrary to typical networks, only the physical cost applies.

Stardust provides multiple mechanisms to handle failures, from the link level to end-to-end detection and isolation. For example, packets with Ethernet CRC errors are dropped at the Fabric Adapters. On the link level, traditional mechanisms are applied to protect against errors (e.g., Forward error correction). If the error rate on a link crosses a threshold, the link marks itself as faulty on reachability cells, and is excluded from cell forwarding. A link is declared valid only after the number of good reachability cells received crosses a threshold. The probability of an error on a link is unaffected by the use of cells, as the number of bits on the wire is approximately maintained. As Stardust minimizes the number of hops in a network, by saving tiers, the probability of an error on a packet is reduced.

When a device fails, it stops sending reachability messages. Consequently, links leading to the device are removed from the reachability tables, and the traffic bypasses the faulty device. In a multi-tier network fabric, this information is also propagated upstream to the Fabric Adapters. If the load per link increases due to a device's removal, the FCI mechanism readjusts the credit rate toward affected devices.

Unlikely scenarios, such as a faulty Fabric Adapter sending unlimited credits, will not collapse the network. Instead, it will degrade the network to the performance of a standard "Push" Ethernet network. While switch-systems are mature solutions, where failure mechanisms have been thoroughly debugged over the years, DCNs face additional reliability challenges, which we intend to study in future work.

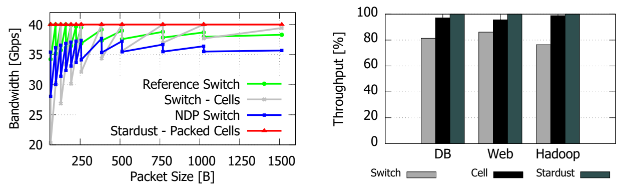


Figure 8: Throughput of switches running at 150MHz and using (a) single packet size, (b) traces from [74].

6 Evaluation

We take a three stage approach to the evaluation. First, running experiment-driven measurements on stand-alone devices and in an existing single-tier Stardust system. Second, we conduct a precise simulation of a two-tier network. Last, in §7 we analyze a large scale network, attending to aspects of cost, power and scalability. The simulation and analysis are based on the results of our measurements.

6.1 Experiment-Driven Measurements

6.1.1 Packet Packing

Measuring the effect of packet packing on throughput is not possible on existing silicon devices. Instead, we use a programmable platform, NetFPGA SUME [84] to demonstrate this effect. We compare four different architectures, all originating from the same source code: NetFPGA $4 \times 10GE$ Reference Switch (Release 1.7.1), NDP switch [41, 40], Stardust, and Stardust-based using non-packed cells. The NDP design originated from the NetFPGA Reference Switch, and treats non-NDP packets similar to it. The Stardust design is similar to the Reference Switch, but instead of switching packets it switches packed cells. The Stardust-based non-packed cells design is identical in implementation to the Stardust design, yet we inject non-packed cells. The credit size is 2KB, and each cell is 64B, as the NetFPGA data path is 32B wide with 2 clock cycles per table lookup. We compare the designs' throughput at a data path clock frequency of 150MHz (The NetFPGA platform is limited at lower frequencies [63]). The traffic, generated using OSNT [12], is a stream of TCP/NDP packets of various sizes. As a reminder, our goal is to explore device-level performance and scalability, rather than protocol-level properties.

As Fig. 8(a) shows, Stardust achieves full line rate for all packet sizes, up to 15%, 30% and 49% better than the Reference Switch, NDP, and non-packed cells, respectively. The Reference Switch achieves full line rate for all packet sizes only at a clock frequency of 180MHz, while NDP fails to reach full line rate for some packet sizes (65B, 97B, 129B) even at 200MHz [3]. Using real world traces [74], Stardust

[3]NDP's TCP performance is identical to the Reference Switch, and only NDP packets experience performance loss. NDP's authors confirmed this performance loss is expected in their design.

maintains its performance edge, as shown in Fig. 8(b). NDP is omitted as it performs worse than the standard switch.

These results indicate the power of packet packing: achieving higher throughput, for all packet sizes, at lower clock frequencies, simplifying network silicon design and enabling better scalability over time.

6.1.2 Throughput and Latency

A number of relevant experimental results from a wider evaluation of Fabric Adapters and Fabric Elements within a single-tier Arista 7500E system are reported here. An extended set of results appears in [53]. The test platform contained $1152 \times 10GE/288 \times 40GE$ ports, connecting 24 Fabric Adapters with a single tier of 12 Fabric Elements, and using Ixia XG12 as a traffic source. The Fabric Adapter devices used, older generation Arad, do not support packet packing.

In a configuration of 960 ports, the platform achieves full line rate on all ports, for all packet sizes. In a configuration of 1152 ports, full line rate is achieved for packets of 384B or more (as packing was not supported), and no packet loss occurred within the network fabric. In the same 1152 port configuration, under full load, the minimum latency is almost independent of packet size (between 2.8μs and 3.5μs), and for the average (3.3μs to 9.1μs) and maximum (5.6μs to 13.5μs) it increases with packet size, as the Fabric Adapter used store-and-forward architecture. The maximal latency for small packets decreases with packet size until reaching the cell size. The average latency variance is in the order of nanoseconds (1ns-11ns).

6.2 Simulation of a 2-Tier Network

To understand the queuing intricacies of Stardust in a two tier network, we use a proprietary packet-level C++ simulator, constructing a network of Fabric Elements and Fabric Adapters, generating traffic and emulating scheduler and control traffic exchange. We use this experiment to discuss aspects of load balancing, queue size distribution and memory requirements.

The network is evaluated using 128K simultaneously active flows. The number of Fabric Adapters is 256, each of them connected using $t = 32$ links to the network fabric. There are 128 Fabric Elements in the first fabric tier, each with 64 links connected to the Fabric Adapters and 64 links connected to the second tier. In the second tier, there are 64 Fabric Elements, each connected using all 128 links to Fabric Elements in the first tier. The length of each link is 100m. We simulate two flows from each Fabric Adapter to every other Fabric Adapter, creating 128K simultaneous flows. The number of connected hosts does not affect the simulation, but assuming the number of uplinks is no more than the number of downlinks, it is in the range of 8K to 64K hosts. The setup size is limited by simulator run times.

Fig. 9(left) presents the latency distribution of network fabric traversal for different fabric utilizations. The latency distribution is tight, and even at 95% utilization, the latency

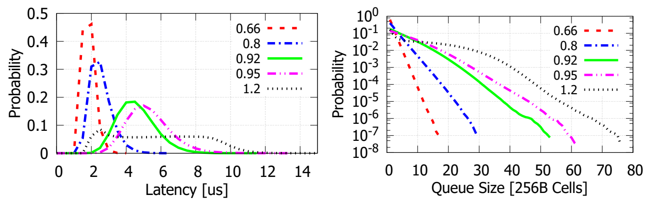


Figure 9: Probability Distribution of latency (left) and link queue size (right) under different loads.

is bound by 13 microseconds. Fig. 9(right) presents queue size distribution in the last stage of the network fabric, for different fabric utilizations. The fabric utilization refers to raw data utilization, after deducting physical protocol overheads, cell headers and control cell overheads. A distribution of an oversubscribed fabric (120%) is included, using FCI to control credit rate. In all runs no cells were lost with the network fabric. With *utilization<1*, queue size probability is an exponential function of fabric utilization, conforming to the theoretical M/D/1 model. In the oversubscribed case, the FCI reduced the effective fabric utilization to 0.9. [4]

We extrapolate the simulation's results to recent network devices, equipped with $256 \times 50Gbps$ links. For a cell size of 256B and a speed up of 1.05 the respective memory will be $128 \times 256B \times 256$, i.e. only 8MB. Given the 50Gbps links, this stands for at most 5μs latency within the Fabric Element.

6.3 Comparisons to Existing Works

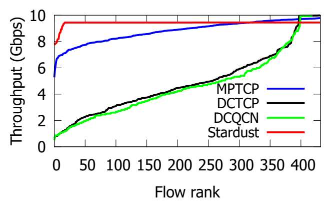
We compare Stardust with Multipath TCP [72], DCTCP [7] and DCQCN [82]. We use *htsim* [72] for our evaluation, and reproduce the experimental environment of [40, 41], implementing a Stardust network, and running TCP. The simulated environment uses a 432-node Fat-tree (see Appendix G).

We run a permutation experiment, where each node in a Fat-tree continuously sends traffic to one node and receives from another, fully loading the data center. Stardust uses 512B packets, 4KB credit size, and 3% credit speed up. Other solutions use 9000B packets. Fig. 10(a) shows the throughput achieved by each node in the network, in an increasing order. Stardust achieves 9.44Gbps on 96% of the flows, and a mean utilization of 94%, compared with 49%, 47% and 90% for DCTCP, DCQCN and MPTCP, respectively.

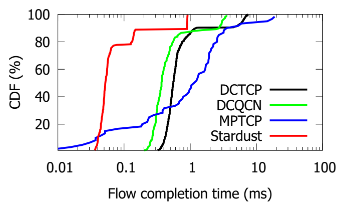
We use the Web workload from [74] to exchange traffic between a pair of nodes and measure the flow completion time (FCT). The same 432-node setup is used, and all other nodes source four long-running connections to a random destination, thus testing the effect of queuing within the network on short flows. Fig. 10(b) shows that Stardust significantly outperforms all other schemes, as the fabric is scheduled. Even flows of 1MB have a FCT of less than a millisecond.

Next, we test an incast traffic pattern. A frontend server

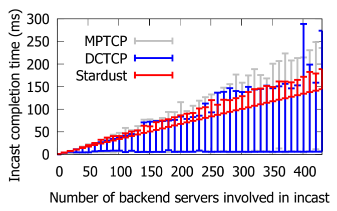
[4] Representing a single trade-off point between latency and throughput. Higher utilization can be gained using less aggressive FCI reaction.



(a) Throughput



(b) FCT



(c) Incast

	B/A
Header Processing	13%
Network Interface	30%
Other logic	60%
I/O	87.5%
Relative area/Tbps	66.6%
Relative power/Tbps	64.8%

(d) Relative Area

Figure 10: Performance comparison, 432-node FatTree (a) Per-flow throughput (b) FCT in an over-subscribed network (c) Incast performance vs number of senders (d) Relative device area of a Fabric Element device (B) and a standard switch (A)

fans out work to many backend servers and then receives their replies, creating an incast scenario. We use a constant response size (450K) and vary the number of backend servers, measuring the FCT. Fig. 10(c) shows the first and last FCT, a measure both of performance and "fairness". Stardust's last FCT is the same as DCTCP and better than MPTCP, but its fairness is considerably better. Furthermore, no packets are dropped within the Stardust fabric.

7 Cost Analysis

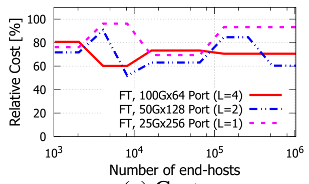
The absolute cost of a Stardust based system is difficult to calculate, as it depends on multiple factors, such as manufacturing costs and platform implementation. Thus, we take a comparative approach and analyze the complexity of Stardust and closely related Fat-tree networks, showing that Stardust is more cost efficient.

Silicon cost Our silicon cost comparison is based on two Broadcom devices, manufactured using the same process. Device A is a standard Ethernet switch (ToR), whereas device B is a Fabric Element [23]. The table in Fig. 10(d) compares the area of these two devices, normalized to their bandwidth, accounting for a difference in the number of I/O (which is favorable for the ToR), detailed further in Appendix C. The relative area/Tbps of a Fabric Element is only 66.6% of that of a ToR. The overall area of a Fabric Adapter is very similar to device A, and is referred as identical henceforth. The area is a good indicator of silicon cost, dominating yield, packaging costs and others factors.

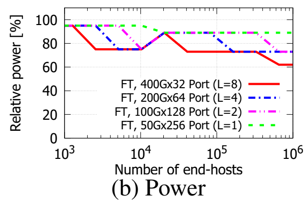
Switch-Platform Cost A Fabric Adapter based switch-platform will cost the same as a standard ToR, while a Fabric Element box will be less expensive. A switch platform is composed of many components, yet the high-end switch silicon dominates the cost. In a Fabric Element platform, the board complexity is reduced and a lower-cost CPU can be used, as only basic control is needed[5]. The reduced complexity also means that more than a single Fabric Element can be used within a single platform, with two being the common case for emerging OCP designs. While in switch-systems the cost ratio between Fabric Adapter and Fabric Element modules of the same generation is high [6], we use the conservative silicon-area ratio as a cost indicator.

[5]From discussions with leading cloud providers, some will prefer the cost-driven CPU option, whereas others will prefer the same type of CPUs across all platforms, for better manageability

[6]Comparison of modules from same manufacturer and platform, based on e.g., https://h22174.www2.hpe.com/SimplifiedConfig/Welcome



(a) Cost



(b) Power

Figure 11: The (a) cost and (b) power of building a Stardust DCN relative to Fat-tree.

Transceivers and Fibers Stardust always opts for the minimal number of transceivers in a DCN. If a network can be constructed with a similar number of tiers using, e.g., 400G and 100G transceivers, then Stardust supports building the network using the least expensive option. The devices are oblivious to whether bundling was used in the transceiver. Furthermore, breakout cables can be used to connect transceivers of different bundling, easing a gradual growth of a network. As every additional tier doubles the number of links required ($\times 4$ transceivers, Table 2), a Stardust based solution becomes more attractive based on cost.

System Cost The cost of a large scale DCN can be cut in half using Stardust. We analyze the relative cost of building a DCN based on current generation data center components, detailed in Appendix D. We assume 25G as the link speed and its link bundles (50G, 100G). ToR and Fabric Adapter platforms are of an equal cost. We use the relative silicon area (0.67) as a relative cost indicator for Fabric Element platforms (See switch-platform cost). Each ToR/Fabric Adapter is assumed to connect conservatively to 40 servers using direct-attach cables, with no over-subscription from the ToR/Fabric Adapter to the network fabric. 100m fibers are used in the last tier (except for a 1-tier network), and 10m fibers within any other tier. The number of devices and links is calculated as detailed in Appendix A. Fig. 11(a) shows the overall cost of building the network. Stardust is always the most cost effective solution.

Power Power consumption significantly varies between switch vendors, e.g., from 150W to 310W [57]. Fig. 11(b) examines the relative power consumption of Stardust, compared with Fat-tree networks with different link bundlings. The calculation is as in Appendix A, using the power ratio indicated in Fig. 10(d). Power per link and equal crosssection bandwidth are considered. The biggest power saving is in networks of up to ten thousands nodes: up to 25% of the entire network's power, and 78% saving within the network fabric, a result of both the reduction in the number of devices, and power saving per device.

The number of network tiers and devices required to build a network (Fig. 2b), and the power consumption of each device, affect the physical space required. The amount of physical space also impacts performance: smaller networks require shorter fibers, reducing the latency, and in turn improving throughput. In Stardust, Fabric Element based chassis can still be used within the network, further reducing space requirements.

8 The Case for Future Data Centers

Using current Fabric Adapter and Fabric Element devices, Stardust naturally fits within regional data centers, and can similarly be used to build efficient clusters of 10K end-hosts (see §6). However, while Stardust reduces the complexity within the network's fabric, the Fabric Adapter maintains the complexity and resource consumption of current ToRs (§6). The challenge becomes the scalability of ToR devices.

We make the case for future Stardust-based data centers, made entirely of Fabric Element devices within the network and of Fabric Adapter-like network interface cards (NICs) at the edges. Here, the concept of a simple network-core is further extended, and the complex network-edge diminished.

The divide-and-conquer approach adapts the end-host's NIC to Stardust through a *reduction* of the Fabric Adapter: combining VM-facing functionality with the handling of cells and scheduling, but at a smaller scale. The number of VOQs will match host-scale requirements [34, 77]. The host's memory will be used for further buffering [54, 55, 58]. The NIC's MAC will become lighter: a fabric interface. A programmable header processor, as in the Fabric Adapter, will also support acceleration in the NIC. Connecting a NIC to a Fabric Element is the same as to a ToR, while the reachability table required is smaller than in a Fabric Adapter, by a factor of *Num-of-Fabric-Adapter-Uplinks/Num-of-NIC-ports*, or can be entirely eliminated if the NIC connects to a single Fabric Element.

This is a natural evolution, supporting trends advocating smarter edge devices [52, 85, 34] and moving scheduling close to the host [71, 78]. We estimate the power consumption of such a NIC to be on-par with current smart NICs [34]. As Jericho [22] already supports a PCIe interface and a direct memory access (DMA) engine to send and receive host traffic, this vision is realizable.

9 Related Work

Switch systems today scale to thousands of ports and to multiple chassis [27, 43, 14, 65]. Their chipsets vary from commodity devices [20] to custom-made solutions [66, 61]. The closed nature of the system, has allowed building upon concepts that have failed in wide area networks (WAN), and in particular in cell based solutions such as ATM [46, 50, 11], while optimizing for system-scale needs (e.g., [48, 24, 10]). While DCNs differ from supercomputing in many ways, supercomputing research also explored aspects of credit-based scheduling and cell-switched cores on a medium scale (e.g., [49, 3, 1]). Stardust is deeply rooted in switch systems, which inspired supercomputing networks.

Stardust is not a circuit switch. Circuit switching (e.g., recent [70, 36, 58]) allows a single path at a time between source-destination pairs and has strong (and sometimes high-latency) scheduling requirements, whereas Stardust allows any-to-any communication at all times and applies distributed end-to-end scheduling.

This paper focused on Clos [30], and Fat-tree topologies [51, 4, 75]. Our link bundling observation is also applicable to other topologies (e.g., [2, 38, 18]).Unlike other solutions, Stardust makes no assumptions about traffic patterns or network utilization, and its routing is trivial [18, 80, 76]. It also remains a single network [59], providing better manageability and resilience. Aspects of network reconfigurability combining different topologies (e.g., [58, 81]) are beyond the scope of this contribution.

Congestion control and load balancing research has tried to optimize performance, from protocol and flow level (e.g., [5, 8, 73, 9, 41]), through flowlets (e.g., [45, 6]) and flowcells [42] to packets (e.g., [25, 37]). Stardust uses evenly distributed dynamic forwarding, combined with end-to-end scheduling. Stardust is refined by combining ingress buffering with congestion avoidance mechanisms (e.g., ECN), providing a lossless solution for short-term congestion.

10 Conclusion

We presented Stardust, a scalable fabric architecture for data center networks, separating the simple network-fabric from the complex network-edge. Stardust applies system-switch architecture on a data center scale, while attending to the scalability limitations of network devices: resources, I/O and performance. The resulting network fabric devices are a radical change from commodity Ethernet switches, eliminating significant overheads in DCNs. Our demonstrated *divide and conquer* approach is practical, power-efficient, cost-effective, scalable, and, critically, a deployable approach.

11 Acknowledgement

We would like to thank the many people who contributed to this work, particularly to the team at Broadcom Yakum. We would like to thank Andrew W Moore and Jeff Mogul for their useful advice. We thank the anonymous reviewers and our Shepherd, Xin Jin, who helped us improve the quality of this paper. This work was partly funded by the Leverhulme Trust (ECF-2016-289) and the Isaac Newton Trust.

References

[1] ABTS, D. Cray xt4 and seastar 3-d torus interconnect. In *Encyclopedia of Parallel Computing*. Springer, 2011, pp. 470–477.

[2] AHN, J. H., BINKERT, N., DAVIS, A., MCLAREN, M., AND SCHREIBER, R. S. Hyperx: topology, routing, and packaging of efficient large-scale networks. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis* (2009), ACM, p. 41.

[3] AJIMA, Y., INOUE, T., HIRAMOTO, S., TAKAGI, Y., AND SHIMIZU, T. The tofu interconnect. *IEEE Micro 32*, 1 (Jan 2012), 21–31.

[4] AL-FARES, M., LOUKISSAS, A., AND VAHDAT, A. A scalable, commodity data center network architecture. In *ACM SIGCOMM Computer Communication Review* (2008), vol. 38, ACM, pp. 63–74.

[5] AL-FARES, M., RADHAKRISHNAN, S., RAGHAVAN, B., HUANG, N., AND VAHDAT, A. Hedera: Dynamic flow scheduling for data center networks. In *NSDI* (2010), vol. 10, pp. 19–19.

[6] ALIZADEH, M., EDSALL, T., DHARMAPURIKAR, S., VAIDYANATHAN, R., CHU, K., FINGERHUT, A., MATUS, F., PAN, R., YADAV, N., VARGHESE, G., ET AL. Conga: Distributed congestion-aware load balancing for datacenters. In *ACM SIGCOMM Computer Communication Review* (2014), vol. 44, ACM, pp. 503–514.

[7] ALIZADEH, M., GREENBERG, A., MALTZ, D. A., PADHYE, J., PATEL, P., PRABHAKAR, B., SENGUPTA, S., , AND SRIDHARAN, M. Data center TCP (DCTCP). In *Proc. ACM SIGCOMM* (Aug. 2010).

[8] ALIZADEH, M., GREENBERG, A., MALTZ, D. A., PADHYE, J., PATEL, P., PRABHAKAR, B., SENGUPTA, S., AND SRIDHARAN, M. Data center TCP (DCTCP). In *ACM SIGCOMM computer communication review* (2010), vol. 40, ACM, pp. 63–74.

[9] ALIZADEH, M., YANG, S., SHARIF, M., KATTI, S., MCKEOWN, N., PRABHAKAR, B., AND SHENKER, S. pfabric: Minimal near-optimal datacenter transport. In *ACM SIGCOMM Computer Communication Review* (2013), vol. 43, ACM, pp. 435–446.

[10] ALVERSON, R., ROWETH, D., AND KAPLAN, L. The gemini system interconnect. In *IEEE 18th Annual Symposium on High Performance Interconnects (HOTI)* (2010), IEEE, pp. 83–87.

[11] ANSI. T1. 105-1991, digital hierarchy optical interface rates and formats specifications (SONET)". *American national standard for telecommunication* (1991).

[12] ANTICHI, G., SHAHBAZ, M., GENG, Y., ZILBERMAN, N., COVINGTON, A., BRUYERE, M., MCKEOWN, N., FEAMSTER, N., FELDERMAN, B., BLOTT, M., ET AL. OSNT: Open source network tester. *IEEE Network 28*, 5 (2014), 6–12.

[13] ARISTA. *7500R Series Data Center Switch Router, Datasheet*. https://www.arista.com/assets/data/pdf/Datasheets/7500RDataSheet.pdf.

[14] ARISTA. *Arista 7500 Switch Architecture ("A day in the life of a packet")*, 2016. https://www.arista.com/assets/data/pdf/Whitepapers/Arista_7500E_Switch_Architecture.pdf.

[15] ARMITAGE, G. J., AND ADAMS, K. M. Packet reassembly during cell loss. *IEEE Network 7*, 5 (1993), 26–34.

[16] BARROSO, L., MARTY, M., PATTERSON, D., AND RANGANATHAN, P. Attack of the killer microseconds. *Communications of the ACM 60*, 4 (2017), 48–54.

[17] BARWICZ, T., TAIRA, Y., LICHOULAS, T. W., BOYER, N., MARTIN, Y., NUMATA, H., NAH, J.-W., TAKENOBU, S., JANTA-POLCZYNSKI, A., KIMBRELL, E. L., ET AL. A novel approach to photonic packaging leveraging existing high-throughput microelectronic facilities. *IEEE Journal of Selected Topics in Quantum Electronics 22*, 6 (2016), 455–466.

[18] BESTA, M., AND HOEFLER, T. Slim fly: A cost effective low-diameter network topology. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC14)* (2014), IEEE, pp. 348–359.

[19] BOSSHART, P., GIBB, G., KIM, H.-S., VARGHESE, G., MCKEOWN, N., IZZARD, M., MUJICA, F., AND HOROWITZ, M. Forwarding metamorphosis: Fast programmable match-action processing in hardware for sdn. In *Proceedings of the ACM SIGCOMM Conference* (New York, NY, USA, 2013), SIGCOMM '13, ACM, pp. 99–110.

[20] BROADCOM. *StrataDNX Switch Solutions*. https://www.broadcom.com/products/ethernet-connectivity/switching/stratadnx.

[21] BROADCOM. *BCM56980 series, 12.8 Tbps StrataXGS Tomahawk 3 Ethernet switch series*, dec 2017. https://www.broadcom.com/products/ethernet-connectivity/switching/strataxgs/bcm56980-series.

[22] BROADCOM. *BCM88690, StrataDNX 10 Tb/s Scalable Switching Device, Product Brief*, 2018. https://docs.broadcom.com/docs/88690-PB100 [Online, accessed January 2019].

[23] BROADCOM. *BCM88790, Scalable Fabric Element 9.6 Tbps Self-Routing Switching Element, Product Brief*, 2018. https://docs.broadcom.com/docs/88790-PB00 [Online, accessed January 2019].

[24] CHRYSOS, N., AND KATEVENIS, M. Scheduling in non-blocking buffered three-stage switching fabrics. In *INFOCOM* (2006), vol. 6, pp. 1–13.

[25] CHRYSOS, N., NEESER, F., CLAUBERG, R., CRISAN, D., VALK, K. M., BASSO, C., MINKENBERG, C., AND GUSAT, M. Unbiased quantized congestion notification for scalable server fabrics. *IEEE Micro 36*, 6 (2016), 50–58.

[26] CISCO. *Cisco Nexus 9000 Series Switches*. https://www.cisco.com/c/en/us/products/switches/nexus-9000-series-switches/index.html.

[27] CISCO. *Cisco CRS Carrier Routing System Multishelf System Description*, jul 2014. http://www.cisco.com/c/en/us/td/docs/iosxr/crs/hardware-install/crs-1/multishelf/system-description/b-crs-multishelf-sys-desc.html.

[28] CISCO. *Cisco Global Cloud Index: Forecast and Methodology, 2015–2020*, nov 2016.

[29] CISCO. *Cisco Nexus 9500 R-Series Line Cards and Fabric Modules White Paper*, 2018. https://www.cisco.com/c/en/us/products/collateral/switches/nexus-9000-series-switches/white-paper-c11-738392.pdf.

[30] CLOS, C. A study of non-blocking switching networks. *Bell Labs Technical Journal 32*, 2 (1953), 406–424.

[31] CUMMINGS, U., DALY, D., COLLINS, R., AGARWAL, V., PETRINI, F., PERRONE, M., AND PASETTO, D. Fulcrum's focalpoint fm4000: A scalable, low-latency 10GigE switch for high-performance data centers. In *IEEE Symposium on High Performance Interconnect (HOTI)* (2009), IEEE, pp. 42–51.

[32] DHARMAPURIKAR, S., AND PAXSON, V. Robust tcp stream reassembly in the presence of adversaries. In *USENIX Security* (2005).

[33] ESCOBAR, J., AND PARTRIDGE, C. A proposed segmentation and reassembly (sar) protocol for use with asynchronous transfer mode (atm). *High Performance Network Research Report* (1990).

[34] FIRESTONE, D., PUTNAM, A., MUNDKUR, S., CHIOU, D., DABAGH, A., ANDREWARTHA, M., ANGEPAT, H., BHANU, V., CAULFIELD, A., CHUNG, E., ET AL. Azure accelerated networking: Smartnics in the public cloud. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)* (2018).

[35] FRANK, D. J., DENNARD, R. H., NOWAK, E., SOLOMON, P. M., TAUR, Y., AND WONG, H.-S. P. Device scaling limits of Si MOSFETs and their application dependencies". *Proceedings of the IEEE 89* (2001), 259–288.

[36] Ghobadi, M., Mahajan, R., Phanishayee, A., Devanur, N., Kulkarni, J., Ranade, G., Blanche, P.-A., Rastegarfar, H., Glick, M., and Kilper, D. Projector: Agile reconfigurable data center interconnect. In *Proceedings of the ACM SIGCOMM Conference* (2016), ACM, pp. 216–229.

[37] Ghorbani, S., Yang, Z., Godfrey, P., Ganjali, Y., and Firoozshahian, A. Drill: Micro load balancing for low-latency data center networks. In *Proceedings of the ACM SIGCOMM Conference* (2017), ACM, pp. 225–238.

[38] Greenberg, A., Hamilton, J. R., Jain, N., Kandula, S., Kim, C., Lahiri, P., Maltz, D. A., Patel, P., and Sengupta, S. Vl2: a scalable and flexible data center network. *Communications of the ACM 54*, 3 (2011), 95–104.

[39] Gurevich, V. Barefoot networks, programmable data plane at terabit speeds. In *DXDD* (2016), Open-NFP.

[40] Handley, M., Raiciu, C., Agache, A., Voinescu, A., Moore, A. W., Antichi, G., and Wójcik, M. *NDP Switch NetFPGA*, 2017. Repository, https://github.com/nets-cs-pub-ro/NDP [accessed December 2017].

[41] Handley, M., Raiciu, C., Agache, A., Voinescu, A., Moore, A. W., Antichi, G., and Wójcik, M. Re-architecting datacenter networks and stacks for low latency and high performance. In *Proceedings of the ACM SIGCOMM Conference* (2017), ACM, pp. 29–42.

[42] He, K., Rozner, E., Agarwal, K., Felter, W., Carter, J., and Akella, A. Presto: Edge-based load balancing for fast datacenter networks. In *SIGCOMM '15* (2015), ACM, pp. 465–478.

[43] Huawei. *CloudEngine 12800 Series Data Center Switches*. http://e.huawei.com/us/products/enterprise-networking/switches/data-center-switches/ce12800.

[44] Intel. *Intel Xeon Processor E7 v4 Family*, 2017. https://ark.intel.com/products/series/93797/Intel-Xeon-Processor-E7-v4-Family [accessed January 2018].

[45] Kandula, S., Katabi, D., Sinha, S., and Berger, A. Dynamic load balancing without packet reordering. *ACM SIGCOMM Computer Communication Review 37*, 2 (2007), 51–62.

[46] Katevenis, M., Sidiropoulos, S., and Courcoubetis, C. Weighted round-robin cell multiplexing in a general-purpose atm switch chip. *IEEE Journal on selected Areas in Communications 9*, 8 (1991), 1265–1279.

[47] Katz, D., and Ward, D. *Bidirectional forwarding detection (BFD)*, 2010. RFC 5880.

[48] Keslassy, I., Chuang, S.-T., Yu, K., Miller, D., Horowitz, M., Solgaard, O., and McKeown, N. Scaling internet routers using optics. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications* (2003), ACM, pp. 189–200.

[49] Kumar, S., Sabharwal, Y., Garg, R., and Heidelberger, P. Optimization of all-to-all communication on the blue gene/l supercomputer. In *2008 37th International Conference on Parallel Processing* (Sept 2008), pp. 320–329.

[50] Le Boudec, J.-Y. The asynchronous transfer mode: a tutorial. *Computer Networks and ISDN systems 24*, 4 (1992), 279–309.

[51] Leiserson, C. E. Fat-trees: universal networks for hardware-efficient supercomputing. *IEEE transactions on Computers 100*, 10 (1985), 892–901.

[52] Li, H. Introducing "Yosemite": the first open source modular chassis for high-powered microservers. https://code.facebook.com/posts/1616052405274961/introducing-yosemite-the-first-open-source-modular-chassis-for-\high-powered-microservers-/, 2015. [Online; accessed January 2017].

[53] Lippis Report. *Arista 7500E Software-Defined Cloud Network Switch Performance Test*, 2014. https://www.arista.com/assets/data/pdf/7500E-Lippis-Report.pdf[Online, access January 2019].

[54] Liu, H., Lu, F., Forencich, A., Kapoor, R., Tewari, M., Voelker, G. M., Papen, G., Snoeren, A. C., and Porter, G. Circuit switching under the radar with reactor. In *NSDI* (2014), vol. 14, pp. 1–15.

[55] Manihatty Bojan, N., Zilberman, N., Antichi, G., and Moore, A. W. Extreme data-rate scheduling for the data center. In *Proceedings of the ACM SIGCOMM Conference* (2015), pp. 351–352.

[56] McKeown, N., Izzard, M., Mekkittikul, A., Ellersick, W., and Horowitz, M. Tiny tera: a packet switch core. *IEEE Micro 17*, 1 (1997), 26–33.

[57] Mellanox. *Mellanox Spectrum Ethernet Switch*. http://www.mellanox.com/page/products_dyn?product_family=218&mtag=spectrum_ic.

[58] Mellette, W. M., McGuinness, R., Roy, A., Forencich, A., Papen, G., Snoeren, A. C., and Porter, G. Rotornet: A scalable, low-complexity, optical datacenter network. In *Proceedings of the ACM SIGCOMM Conference* (2017), ACM, pp. 267–280.

[59] Mellette, W. M., Snoeren, A. C., and Porter, G. P-fattree: A multi-channel datacenter network topology. In *HOTNETS* (2016), ACM, pp. 78–84.

[60] Moore, G. E. Cramming more components onto integrated circuits. *Electronics 38* (April 1965), 114–117.

[61] Morgan, T. P. *Homegrown 25G Chips Help Cisco Hold The Switch Line*, 2016. https://www.nextplatform.com/2016/03/01/homegrown-25g-chips-help-cisco-hold-the-switch-line/.

[62] Morgan, T. P. *Flattening Networks And Budgets With 400G Ethernet*, jan 2018. https://www.nextplatform.com/2018/01/20/flattening-networks-budgets-400g-ethernet/ [accessed January 2018].

[63] NetFPGA. *NetFPGA-SUME-live, issue 36: 10G port - Attachment unit - Rx side - inefficiency*, 2017. https://github.com/NetFPGA/NetFPGA-SUME-live/issues/36.

[64] Networks, B. *Barefoot Tofino, World's fastest P4-programmable Ethernet switch ASICs*, 2016. https://barefootnetworks.com/products/brief-tofino/.

[65] Networks, J. *QFabric System, Scalable Fabric Switching System.* https://www.juniper.net/uk/en/products-services/switching/qfabric-system/.

[66] Networks, J. *Juniper Networks Unveils Industry's Most Powerful Core Routing Platform*, 2015. http://investor.juniper.net/investor-relations/press-releases/press-release-details/2015/Juniper-Networks-Unveils-Industrys-Most-Powerful-Core-Routing\-Platform/default.aspx.

[67] Ohring, S. R., Ibel, M., Das, S. K., and Kumar, M. J. On generalized fat trees. In *Proc. Ninth International Parallel Processing Symposium* (1995), IEEE, pp. 37–44.

[68] Patterson, D. A., and Hennessy, J. L. *Computer organization and design: the hardware/software interface*. Newnes, 2013.

[69] Popescu, D. A., and Moore, A. W. Ptpmesh: Data center network latency measurements using ptp. In *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), 2017 IEEE 25th International Symposium on* (2017), IEEE, pp. 73–79.

[70] Porter, G., Strong, R., Farrington, N., Forencich, A., Chen-Sun, P., Rosing, T., Fainman, Y., Papen, G., and Vahdat, A. *Integrating microsecond circuit switching into the data center*, vol. 43. ACM, 2013.

[71] Radhakrishnan, S., Geng, Y., Jeyakumar, V., Kabbani, A., Porter, G., and Vahdat, A. Senic: Scalable nic for end-host rate limiting. In *NSDI* (2014), vol. 14, pp. 475–488.

[72] Raiciu, C., Barre, S., Pluntke, C., Greenhalgh, A., Wischik, D., and Handley, M. Improving datacenter performance and robustness with Multipath TCP. In *Proc. ACM SIGCOMM* (Aug. 2011).

[73] Raiciu, C., Barre, S., Pluntke, C., Greenhalgh, A., Wischik, D., and Handley, M. Improving datacenter performance and robustness with multipath tcp. In *ACM SIGCOMM Computer Communication Review* (2011), vol. 41, ACM, pp. 266–277.

[74] Roy, A., Zeng, H., Bagga, J., Porter, G., and Snoeren, A. C. Inside the social network's (datacenter) network. In *ACM SIGCOMM Computer Communication Review* (2015), vol. 45, ACM, pp. 123–137.

[75] Singh, A., Ong, J., Agarwal, A., Anderson, G., Armistead, A., Bannon, R., Boving, S., Desai, G., Felderman, B., Germano, P., et al. Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network. *ACM SIGCOMM Computer Communication Review 45* (2015), 183–197.

[76] Singla, A., Hong, C.-Y., Popa, L., and Godfrey, P. B. Jellyfish: Networking data centers, randomly. In *NSDI* (2012), vol. 12, pp. 17–17.

[77] Solarflare. *Ultra Scale, Scaling from two to two thousand containers on a single server*, 2018. https://www.solarflare.com/ultra-scale.

[78] Stephens, B., Singhvi, A., Akella, A., and Swift, M. Titan: Fair packet scheduling for commodity multiqueue nics. In *2017 USENIX Annual Technical Conference (ATC'17)* (2017), pp. 431–444.

[79] Technologies, M. *Mellanox Spectrum-2 Ethernet Switch*, 2017. http://www.mellanox.com/related-docs/prod_silicon/PB_Spectrum-2.pdf.

[80] Valadarsky, A., Dinitz, M., and Schapira, M. Xpander: Unveiling the secrets of high-performance datacenters. In *Proceedings of the 14th ACM Workshop on Hot Topics in Networks* (2015), ACM, p. 16.

[81] Xia, Y., Sun, X. S., Dzinamarira, S., Wu, D., Huang, X. S., and Ng, T. A tale of two topologies: Exploring convertible data center network architectures with flat-tree. In *Proceedings of the ACM SIGCOMM Conference* (2017), ACM, pp. 295–308.

[82] Zhu, Y., Eran, H., Firestone, D., Guo, C., Lipshteyn, M., Liron, Y., Padhye, J., Raindel, S., Yahia, M. H., and Zhang, M. Congestion control for large-scale rdma deployments. In *SIGCOMM* (2015), pp. 523–536.

[83] Zilberman, N. An evaluation of NDP performance . Tech. Rep. UCAM-CL-TR-933, University of Cambridge, Computer Laboratory, Feb. 2019.

[84] Zilberman, N., Audzevich, Y., Covington, G., and Moore, A. W. NetFPGA SUME: Toward 100 Gbps as Research Commodity. *IEEE Micro 34*, 5 (September 2014), 32–41.

[85] Zilberman, N., Moore, A. W., and Crowcroft, J. A. From photons to big-data applications: terminating terabits. *Phil. Trans. R. Soc. A 374*, 2062 (2016), 20140445.

A The Math Behind Network Size

In this section, we provide the mathematical justification for the number of elements used in a Fat-tree based network. The number of devices required to build a Fat-tree network is long known [51, 67], and we build upon it to demonstrate the dependency on link bundling. For the clarity of the discussion, Table 1 lists the parameters used to describe the network topology, and Table 2 provides the number of different elements within a multi-tier, Fat-tree based, DCN and the parameters used to describe the network topology. Note that Table 2 shows the number of network elements, and not the number of connected end hosts, and that unless noted otherwise, downlinks from a ToR to end hosts are not counted.

In a 1-Tier network, the number of switches equals, at most, the number of uplink ports from a ToR (t). The number of ToRs equals, at most, the number of downlinks from a switch, which is in this case the number of ports on the switch (k). The number of link bundles in this network will equal the number of uplink ports from a ToR times the number of ToRs, which equals the number of downlinks from a switch times the number of switches ($t \times k$). As this is a 1-Tier network, there is only one layer of links within it, therefore the number of links per ToR will be the same as the number of uplinks ($t \times l$).

In a 2-Tier network, the number of downlink ports from a switch in a second tier will still be k, but a switch in the first layer will have $k/2$ downlink ports and $k/2$ uplink ports. The number of uplink ports from a ToR is the same. This leads to a network of $k^2/2$ ToRs (at most). The number of link bundles in the first tier will equal the number of ToRs times the number of uplink ports per ToR ($t \times k^2/2$). With a naive construction, assuming a fully provisioned network, this will equal the number of link bundles between the switches in the first and the second tier, bringing the overall number of link bundles in the network to $t \times k^2$. The number of links per ToR (excluding ToRs' downlinks) equals the number of link bundles in the network, multiplied by the number of links per link bundle and divided by the number of ToRs ($2 \times t \times l$). The same logic is used to continue and build networks of 3, 4 and n tiers.

So far we have limited the discussion to the network devices within the DCN. Next, we consider the number of end hosts connected to this network. The number of end hosts will always be $O(k^n/2^{n-1})$ in a fully provisioned n tier network. The exact number of end-hosts will depend on the number of downlink ports from the ToR switch. This number is not necessarily t: the link bundle from a host to the ToR can be different than the link bundle from the ToR to the first switch. Furthermore, a ToR can be over-subscribed, in which case the number of uplink and downlink ports will be different. Given d downlink ports from a ToR, the number of end hosts connecting to an n tier network will be $d \times k^n/2^{n-1}$.

While over-subscription is most common at the ToR level,

Parameter	Description
k	switch radix
t	number of ToR uplink ports
l	number of links in a link-bundle

Table 1: The parameters used to describe a Fat-tree based network.

it is also possible within the network. This does not change the math of a 1-Tier network, but in a two (or n) tier network, each switch will use u uplink ports, and $k-u$ downlink ports, and the math will be adjusted accordingly: the maximum number of ToRs will be $k \times (k-u)$, and the number of switches will be $t \times (k+u)$.

B Parallel Processing of Packets

§2.3 discussed the number of processing cores required in a device. In this appendix we explain this calculation. Here we focus not on bandwidth, but rather on packet processing rate.

Let us assume a packet of size S bytes, running through an Ethernet switch of bandwidth B bits/sec. As this is Ethernet, there are preamble, start of frame delimiter (SFD) and inter frame gap between packets on the wire, G. The packet rate R required from a device is therefore:

$$R = B/(8 \times (S+G)) \tag{1}$$

Note that Equation 3 makes no assumption on the architecture and design of the switch.

Most packet switches today are pipelined (e.g. [19]) as a way to increase performance. The length of the pipeline does not affect the rate at which packets are being processed, rather the latency through the pipeline. Let us assume that each stage through the pipeline takes c clock cycles, and that the clock frequency of the device is f. This means that the number of packets r processed in the pipeline every second is:

$$r = f/c \tag{2}$$

Under optimal conditions, $c = 1$, as in a pipelined design the minimum time required per stage is one clock cycle. Note that an *operation* within a stage may take less than one clock cycle, but a stage will always be clocked (and sampled) in order to avoid metastability and close timing. This leads to $r = f$.

We define the amount of parallelism required in a switch P as:

$$P = R/r \tag{3}$$

P defines the ratio between the number of arriving packets that needs to be processed every second, and the rate at which the pipeline can process packets. Where $P > 1$ it means that more packets need to be processed every second by the device than a single pipeline can process.

For example, let us assume packet size $S = 64B$, switch bandwidth of $B = 12.8Tbps$, gap $G = 12B + 8B = 20B$, clock frequency of $f = 1GHz$ and $c = 1$ clock cycle per packet. The parallelism required is 19.047:

$$\begin{aligned} R = B/(8 \times (S+G)) = 19.047E+9 \\ r = f/c = 1E+9/1 = 1E+9 \\ P = R/r = 19.047 \end{aligned} \tag{4}$$

In a similar manner, a packet size of 256B will require $P = 6.06$.

The way to handle $P > 1$ may vary between chip architectures. One common solution is to use more than one pipeline within a switch [39, 62]. Another solution is to process more than one packet within the pipeline every clock (in a manner not unlike CPU's pipelines) - however such a solution is much more complex. Packet packing effectively increases the packet processing rate, but it operates on packed packets rather than on individual ones.

Recent devices have used four pipelines as a way to increase parallelism [39, 62], while processing one packet every clock ($c = 1$). This means that such devices likely support full line rate only above a certain packet size [7].

C Silicon Area

§7 briefly introduced the silicon area comparison of Stardust devices and a standard ToR switch. In this section we extend this discussion, based on the Table in Fig. 10(d).

Device A is a standard Ethernet switch (ToR), whereas device B is a Fabric Element [23]. A significant difference between devices A and B is in the area consumed by header processing. Device A supports advanced (programmable) header processing, whereas device B supports only simple cell header parsing. There is also a notable difference in the resources required for lookup tables: Device B only requires a reachability table, whereas device A requires standard forwarding tables, such as for IPv4 lookup. Assuming a network with N end hosts, and only 40 servers per rack, device A will require an exact match IPv4 table size $N \times (32 + \log_2 k)$, where k is the device's radix, whereas Device B will only need to point to Fabric Adapters, thus its table size will be $(N/40) \times log_2 k$. Using hashing will reduce the required memory size, but this simplified explanation gives a notion of the difference between the two devices.

In terms of I/O, both devices use the same libraries and therefore have a relatively close I/O area ratio[8]. However, the network interface modules beyond the I/O, e.g. the MAC, are significantly different. Device A requires a full-blown MAC

[7]This is an estimation based on current manufacturing processes, as f and c of switches are not published by vendors.

[8]The difference is for practical implementation reasons.

Tiers	Max ToRs	Max of Switches	Switches per ToR	# of Link Bundles	Links per ToR
1	k	t	t/k	$t \times k$	$t \times l$
2	$k^2/2$	$3/2 \times t \times k$	$3 \times t/k$	$t \times k^2$	$2 \times t \times l$
3	$k^3/4$	$5/4 \times t \times k^2$	$5 \times t/k$	$3/4 \times t \times k^3$	$3 \times t \times l$
4	$k^4/8$	$7/8 \times t \times k^3$	$7 \times t/k$	$7/8 \times t \times k^4$	$7 \times t \times l$
n	$k^n/2^{n-1}$	$(2 \times n-1)/2^{n-1} \times t \times k^{n-1}$	$(2 \times n-1) \times t/k$	$(1-1/2^{n-1}) \times t \times k^n$	$(2^{n-1}-1) \times t \times l$

Table 2: The number of different elements within a multi-tier, Fat-tree based network. The maximum size of a network of n tiers using a switch with port radix k is $O((k/2)^n)$.

supporting different Ethernet standards, whereas Device B requires a simpler module to extract the cells. The gain in area per port is 70%.

For the remaining functionality of the device, the savings using a Fabric Element amounts to 40%. While in a standard switch, the amount of packet buffering is a compromise between silicon area and packet drop under different traffic scenarios, in the Fabric Element, cell buffering is agnostic to traffic scenarios.

In a Fabric Adapter [22], functionality supporting Stardust (e.g., cell generation, load balancing, and credit generation) takes about 8% of the device's area. This area is largely compensated by the saving on network fabric facing interfaces, a gain of 70% per port. The number of VOQs supported is directly mapped to memory resources, where 128K VOQs consume roughly 4MB, an order of magnitude less than consumed by a header processing module [19]. The overall area of the Fabric Adapter is very similar to device A.

The area of a device can vary based on a number of factors. For example, an Ethernet switch can trade-off buffer space for packet processing functionality or lookup table sizes, and a Fabric Element may trade-off cell buffering for area. A Fabric Element based on the design choices in [19] will require just 45% of the area of an RMT switch[9]. Table 10 does represent, however, existing devices and not a theoretical case.

D Cost Estimation

Cost estimation is conducted using list prices collected from online resellers on September 12, 2018. The prices are listed in Table 3. The websites are authorized resellers of the listed components. Prices are used to calculate cost ratio, rather than as indicative costs. Our calculations do not take into account other real-world considerations that affect cost, such as the pricing for high quantities or a buyer's bargaining power.

E Resilience

Recovering from a link failure follows a local detection of a link failure, and the propagation of this information to all other devices. Local detection of a failure is done on a nanosecond scale, as a result of a loss of signal or high bit error rate (BER). The time it takes to propagate the information using reachability messages is configurable. This appendix formulates this time, and Table 4 lists the parameters used below.

Item	Price [USD]	Source
Edgecore AS7816-64X 64-Port 100GbE	\$16200	Colfax
Edgecore Wedge 100BF-65X 65-Port 100GbE	\$16200	Colfax
Passive Copper Cable 100GbE, 2 meters	\$84	Colfax
100G QSFP28 Optical Module Short Range	\$435	Colfax
50G QSFP28 Optical Module Short Range	\$280	Estimated
25G SFP28 Optical Module Short Range	\$125	Colfax
Fiber, 10m Fiber, 100m	\$8 \$62	FS FS

Table 3: Indicative component costs - List prices.

Let f be the core frequency of a device, and c the configurable number of clock cycles between messages, per link. The time t' between every pair of messages will be:

$$t' = \frac{c}{f}$$

This means a message is generated every c/f seconds. Assume that each reachability message is of B bytes and includes a bitmap of b bits indicating the reachability of b Fabric Adapter devices. Assume that each Fabric Adapter connects to h hosts. The number of messages M required to propagate the reachability table of a DCN connecting N hosts will be:

$$M = \left\lceil \frac{N}{h \times b} \right\rceil$$

If a network has n tiers, then at the worst case, the link that failed was between a Fabric Adapter and a Fabric Element, which means that the information needs to propagate through the entire network ($2 \times n$ hops), minus one hop (the failed link). The time it would take to propagate the updated reachability table through the entire network would be:

$$t = t' \times M \times (2 \times n - 1) =$$
$$\frac{c}{f} \times \left\lceil \frac{N}{h \times b} \right\rceil \times (2 \times n - 1)$$

[9] [19] has insufficient data to compare to the Fabric Adapter.

Parameter	Description	Example Value
f	Core frequency	1GHZ
c	Number of cycles between messages	10,000
t'	Time between reachability messages	10μs
b	Reachability bitmap size	128
B	Reachability message size	24B
h	Number of hosts per Fabric Adapter	40
N	Number of hosts connected to the DCN	32,000
n	Number of tiers	2
M	Number of messages propagating a full reachability table	7
th	Reachability status change threshold	3
pd_i	Propagation delay through link i	50ns, 500ns
s	link speed	50Gbps

Table 4: The parameters used to describe the propagation of reachability updates.

To avoid momentary effects and account for a potential loss of a reachability message during the updates, one would usually opt for multiple updates of a link's value (th) before updating its state, which means the time it takes to react to a link failure would be $t \times th$.

A slightly more accurate calculation will also take into account the propagation delay on each link (pd_i):

$$t \times th = \sum_{i=1}^{2\times n-1} \left((t' + pd_i) \times M \times th \right) =$$
$$\sum_{i=1}^{2\times n-1} \left(\frac{c}{f} + pd_i \right) \times \left\lceil \frac{N}{h \times b} \right\rceil \times th$$

The value of c is determined such that the effect of reachability messages on the available bandwidth will be minimal. Assume that the link speed is s. The relative bandwidth consumed by reachability messages would be:

$$\frac{B \times 8}{t' \times s} = \frac{B \times 8 \times f}{c \times s}$$

Using the example values indicated in Table 4, the overhead of reachability updates is 0.04% of the bandwidth, and the time it takes to recover from a failed link, which is the time it takes to propagate an update to the farthest Fabric Adapter, is 652μs. Note that the difference from the illustrative example in §5.9 is the propagation delay on the links.

F Push Fabric vs Pull Fabric

§5.2 made the case for Pull Fabric vs Push Fabric, with no traffic classes. Here we illustrate the case of traffic classes on an Ethernet based network fabric, compared with Stardust. The scenario is depicted in Fig. 12. On a single device there are two 100GE ports, A and B. From one input device, 100Gbps of a *high* traffic class are injected toward A (marked red), and additional 100Gbps of a *low* traffic class are injected toward B (marked green). Additional 100Gbps of a *high* traffic class are injected towards A from a second device (marked red). As a result, B's traffic will be pushed to the fabric, where all of it will be dropped at the packet switch. The throughput of the destination ToR will be 100Gbps, all toward port A.

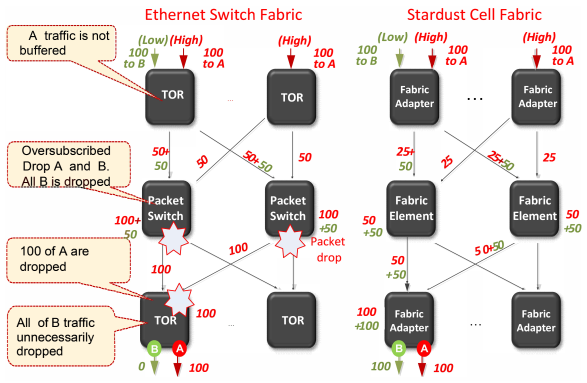


Figure 12: Packet drops in Ethernet switch fabric vs. Stardust scheduled fabric with traffic classes

In Stardust, credits are sent separately from port A and from port B to each ingress Fabric Adapter. Each input device receives credits sufficient for just 50Gbps toward port A, and the first device received credits sufficient to send 100Gbps toward port B. As cells are distributed across all links, the traffic toward A and B is evenly divided between the links, meaning that no link is over-subscribed and no traffic is dropped within the network. The eventual throughput from Stardust is twice the throughput using the standard Ethernet switch.

G Performance Simulations

The simulator of choice for performance comparison is *htsim* [72]. Htsim is used as it provides, under an open-source license, implementations of several data center protocols (TCP, MPTCP, DCTCP, DCQCN and NDP), along with scripts allowing the experiments of [41] to be reproduced. The simulator used in §6.2 is unsuitable for this purpose, as it is an architecture-specific, low-level simulator, that would take days to run experiments that are conducted in htsim within minutes.

The setup used in §6.3 reproduces the setup used in [41], using a 432-node Fat-tree. All links in the system are of 10Gbps, and the same number of tiers is used across protocols and the experiments. Although the htsim simulator is fast, it still takes many hours to simulate a permutation matrix of Stardust in a 432 node Fat-tree.

Our simulations reproduce as-is the experiments from [41] using DCTCP with ECN, DCQCN and MPTCP . We implement a model of Stardust into this environment. We use an unmodified TCP (New Reno) on top of Stardust, which is the least favorable scenario. While the Fabric Adapter supports sending congestion notifications to the host, the presented results do not take advantage of such features, presenting 'raw' Stardust performance. DCTCP and MPTCP use 100 packet output queues, and DCQCN uses 100 buffers per port, shared between interfaces. Stardust uses similar buffering resources, providing an apple-to-apple comparison.

The Stardust configuration used in the experiment uses 512B cell size, 4KB credit size and 3% credit speed up. The use of 512B cell size and 4KB credit size is intended to reduce simulation time. A simulation using 256B cells and 2KB credits takes 12 hours per permutation on a Xeon E5-2660 v4 server. Both settings are realistic, though a smaller credit size will improve fairness in the given setup. The scheduler at the egress Fabric Adapter uses a simple round robin between all flows, intended to show fairness. Other scheduling schemes are also supported by the Fabric Adapter (§4).

The performance simulations use different workloads. The throughput simulation (Fig. 10(a)) uses continuous flows and 9000B packets, which favors existing data center protocols. The FCT simulation uses the Facebook Web workload [74] flow size distribution, which was implemented by [41]. Background traffic is similar to the throughput simulation. The Incast experiment, sending traffic from an increasing number of sources to a single destination, uses 450KB flow size and 9000B packets. In all the simulations, the packets are chopped into cells by Stardust.

The simulation results presented in this paper do not capture all the minutiae of the Stardust design, for example, avoiding synchronization issues when distributing cells across links. Consequently, the actual performance of Stardust, e.g., as measured in §6.1, is better than simulated in §6.3. Most of the aforementioned features are not implemented due to the level of abstraction provided by the simulator. Htsim treats the network as a collection of pipes and queues, and is oblivious to hardware limitations, such as described in §2.

DCQCN is not included in the incast experiment depicted in Fig. 10(b) as it is missing from the reproducibility package of [40], and due to insufficient resources to accurately re-implement it. We opt to omit NDP from our performance comparison as we find that it is very sensitive to the experimental setup and configuration. For example, a very realistic scenario of using 1500B packets rather than 9000B reduces its mean utilization by 14%. In a different case, changing the simulation's seed resulted in performance collapse. We refer the reader to [83] for a detailed analysis.