

BACKGROUND

Between buildings, data already travels as light



Fiber

A hair-thin glass strand. Cities already have hundreds of spare fibers in the ground, so leasing more is cheap.

Wavelength

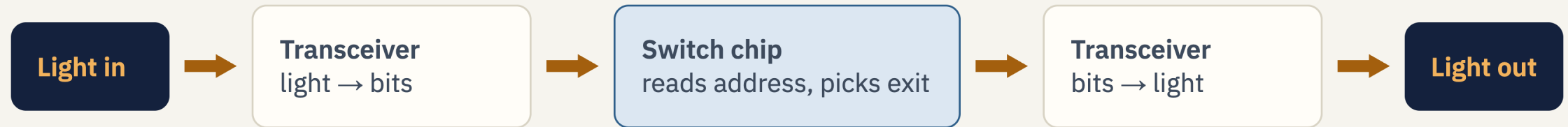
One "color" of light. A single fiber carries 40 to 64 colors at once, each about 400 Gbps.

Transceiver

Turns electrical bits into light and back. One per color, per end. This is the expensive part.

ELECTRICAL PACKET SWITCH

An electrical switch reads every packet



Like a post office that opens every envelope to read the address. This round trip is called **O-E-O**: optical → electrical → optical.

Good at

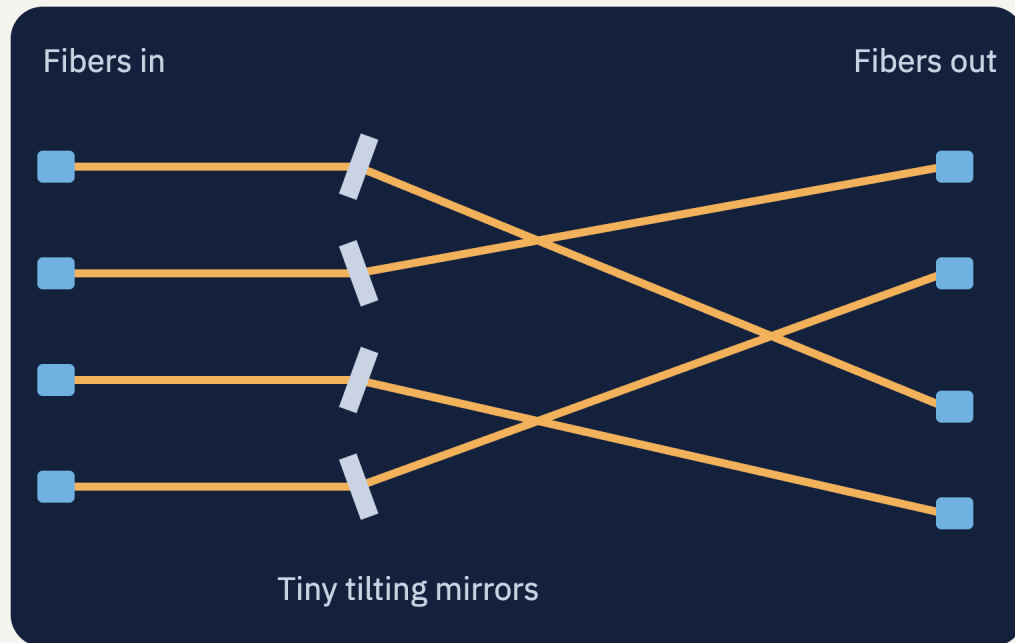
Fine-grained sharing: each packet can go anywhere, and queues absorb bursts.

Costly because

Every wavelength at every switch needs its own transceiver, plus power-hungry chips.

OPTICAL CIRCUIT SWITCH

An optical switch just steers the light



Like a railway track switch: it sets a path, then every train on that track goes the same way. It never looks at the cargo.

Good at

No conversion, so no transceivers inside the network. Little power. Works at any data rate.

Gives up

Moves whole beams, not packets. Changing a path takes milliseconds, not nanoseconds.

KINDS OF OPTICAL SWITCHES

Switch the whole fiber, or one color at a time



Fiber switch

Optical space switch (OSS)

- Moves all of a fiber's colors together
- Hundreds of ports, little light lost
- Simple and cheap: \$100–200 a port
- Coarse: rounds up to whole fibers



Wavelength switch

WSS; combined into a cross-connect (OXC)

- Sends each color to its own output
- Only tens of ports per device
- Loses more light, costs more
- Finer: one fiber, many destinations

SIDE BY SIDE

Flexibility versus cost

	Electrical packet switch	Optical circuit switch
What it moves	Individual packets	Whole beams of light
Reads the data?	Yes, every header	No, it is blind to content
Light-to-bits conversion	At every port	Only at the two ends
Time to change paths	Per packet (nanoseconds)	About 20 milliseconds
Cost per port	Port plus a costly transceiver	\$100–200, no transceiver
Power	High	Very low, mostly passive
Speed upgrade (100G → 400G)	Replace hardware	Unchanged
Capacity use	Shared packet by packet	Rounded up to whole circuits, so extra fiber

WHERE OPTICAL SWITCHING FITS

It works best when traffic changes slower than the switch

HARD FIT

Inside one data center

Rack-to-rack traffic shifts in microseconds. Designs like Helios and c-Through keep electrical switches alongside.

DIFFERENT GOAL

Across continents

Long fiber is scarce and costly, so optics are used to squeeze the most out of every link, e.g. OWAN.

GOOD FIT · IRIS

Between DCs in one city

Huge, steady traffic over tens of km, and spare fiber is cheap. Slow switching is fine.

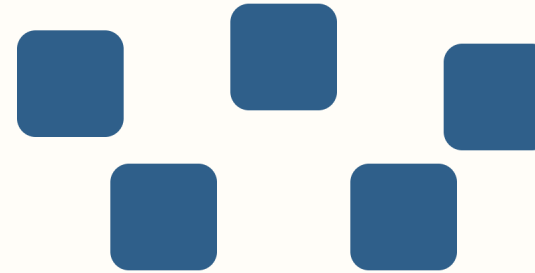
THE SETTING

Clouds now build regions, not mega data centers



Before: one mega-DC

Hundreds of thousands of servers in one place



Now: a region

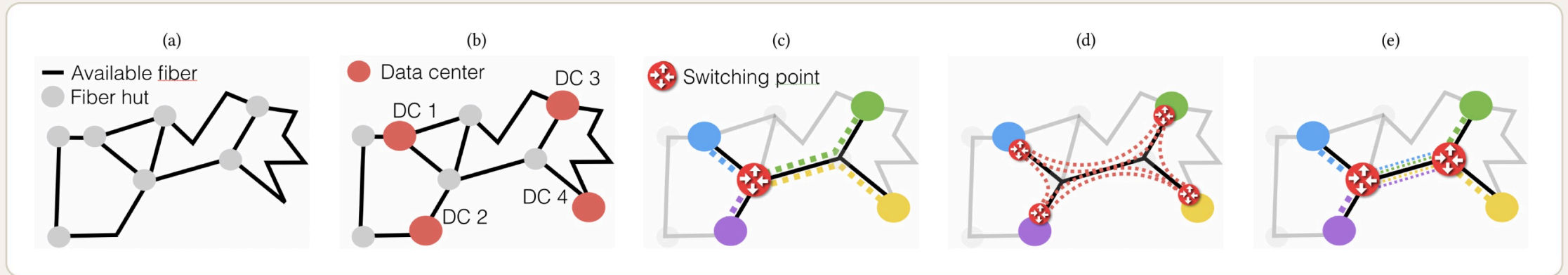
5–20 smaller DCs, tens of km apart

Why: big sites can't find land, power and fiber near cities, and one disaster shouldn't take out a whole region.

The network between them is the DCI. It has about 100× more ports than the links to the wide-area backbone.

TWO WAYS TO WIRE A REGION

Through a central hub, or directly



(c) Centralized

Every DC connects to hubs that act as one big switch. Simple and cheap. Azure today.

(d) Fully distributed

Every pair of DCs gets its own direct connection along the shortest path.

(e) In between

Groups of DCs share a local hub, and the hubs connect. AWS appears to do this.

WHY DISTRIBUTED IS BETTER

Shorter paths and more places to build

6×

Lower delay, one example

Tokyo-like region: 1.2 ms round trip via the hubs, 0.2 ms over a direct 19 km link.

>2×

Lower delay, broadly

For more than 20% of DC pairs across 22 Azure regions. Delay drops for at least 60%.

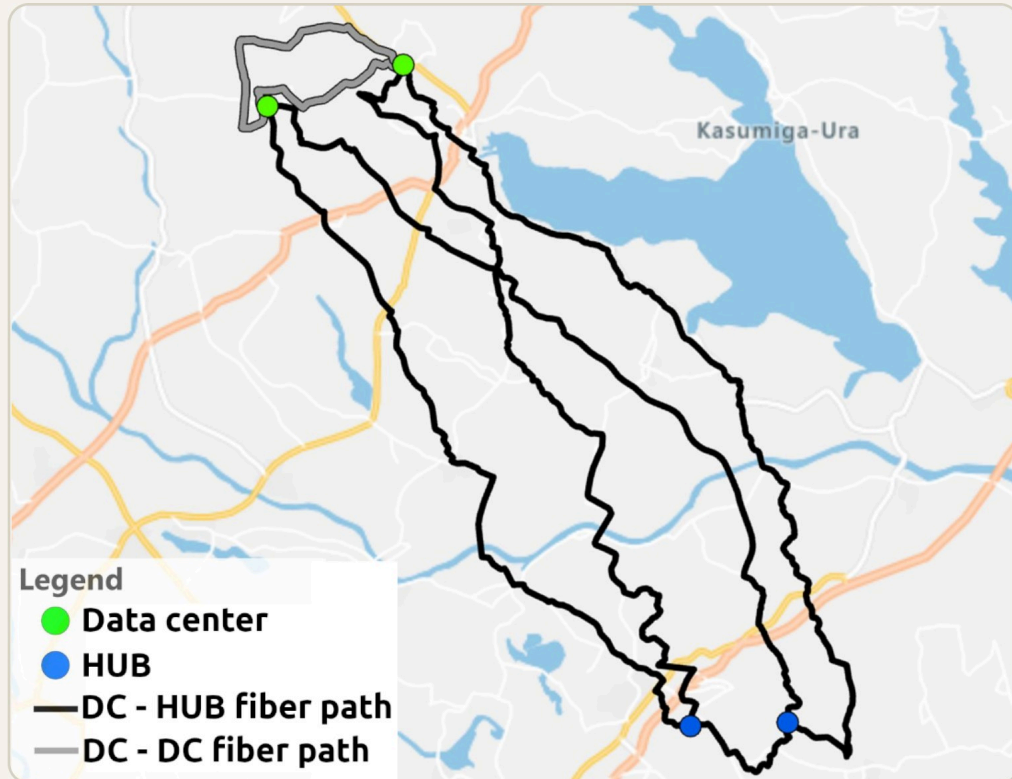
2–5×

More area for a new DC

Across 33 regions. Without hubs, a new DC only has to be near the other DCs.

Bonus: hubs are a reliability trap. Put them close together for more siting area, and one disaster can hit both.

Going through a hub can be a long detour



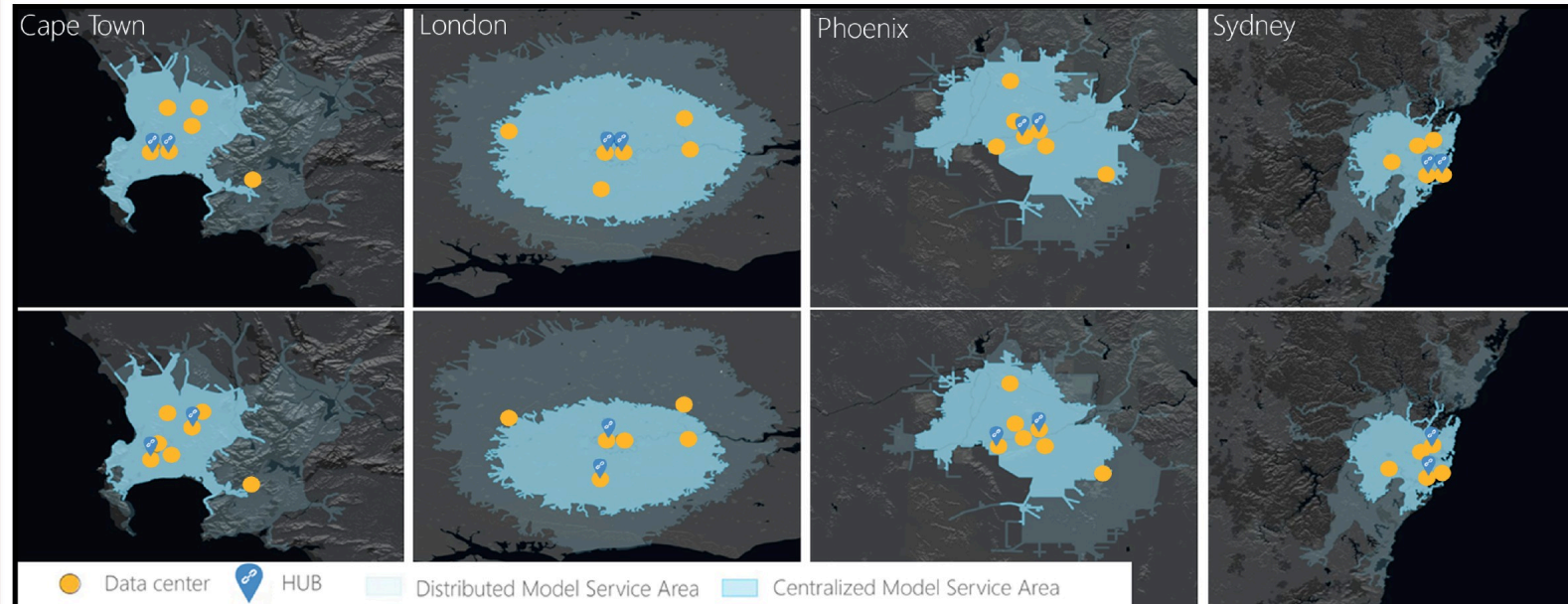
1.2 ms

Round trip via a hub. Black paths: 53–60 km down to the hubs and back up.

0.2 ms

Round trip over the direct 19 km gray path. Six times faster.

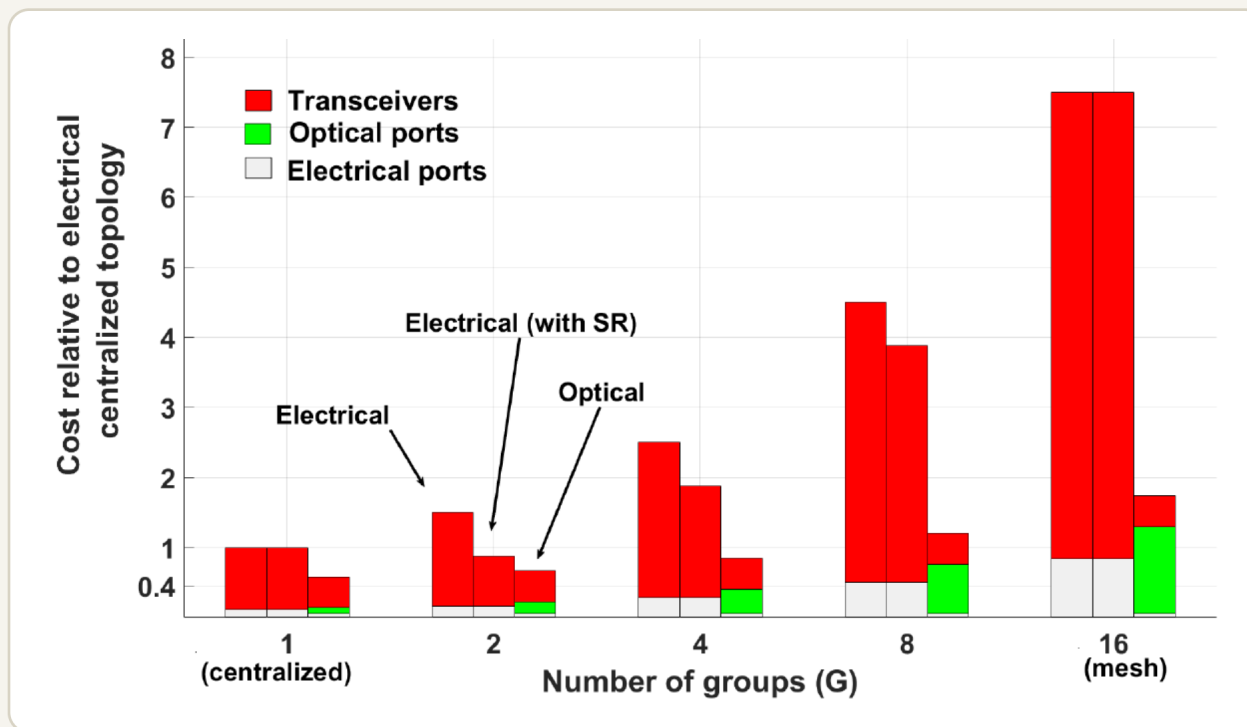
Without hubs, there is far more room to build



Light blue: where a new DC may go with hubs. Wider gray shading: the extra area the distributed design allows. Top row: hubs 4–7 km apart; bottom row: 20–24 km apart.

THE PROBLEM

With electrical switches, distributed costs about 7× more



Every hop is O-E-O

Each switching point needs transceivers for every wavelength passing through.

Red is transceivers

They dominate. One costs about 10× an electrical switch port.

Figure 7 from Dukic et al., §2.4 (16-DC example region)

THE PROBLEM, PART 2

Distributed is also harder to build and run

Hubs: one big switch

- Every DC plugs all its capacity into the hubs
- Same recipe works in any region, whatever the fiber map
- Few sites to manage

Distributed: custom per region

- Plan which DC pairs connect, and with how much capacity
- Switching equipment spread over many huts
- Growing the region means upgrading many sites

Iris's answer

About 10× fewer ports in the middle of the network to manage. Planning runs once, offline. Reconfiguring is just drain, switch, resume, with no constant tuning of light power.

IRIS'S KEY INSIGHT

Transceivers are expensive. Fiber is cheap. So keep the data as light.

~\$1,300/yr

One DCI transceiver, amortized. Carries one color.

~\$3,600/yr

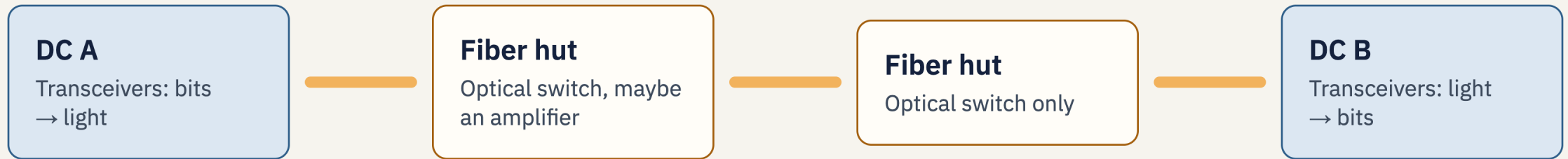
One leased fiber pair. Carries 40–64 colors.

\$100–200

One optical switch port. Switches a whole fiber.

HOW IRIS WORKS

Bits become light once, at the edge



The orange path stays light the whole way: an end-to-end optical circuit.

Switch whole fibers

Optical space switches at huts and DCs point each fiber, with all its colors, toward one destination DC.

Plan once, offline

An algorithm picks links, fiber counts, amplifier spots and shortcuts so every pair uses its shortest path, even after 2 fiber cuts.

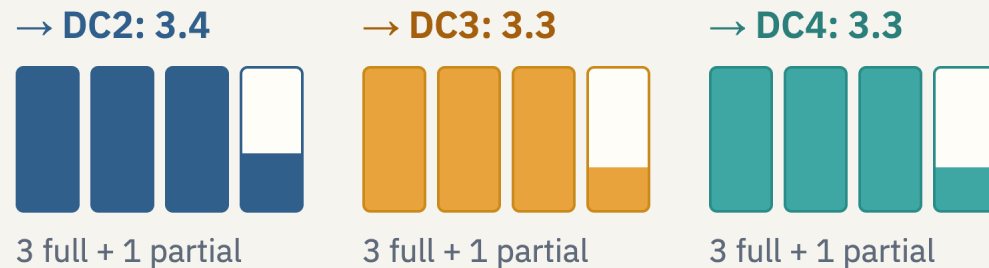
Reconfigure rarely

A central controller moves traffic off a path, flips the switches, then uses the new path. Only on failures or slow traffic shifts.

WHY $C + N - 1$ FIBERS PER DC

A fiber points at one DC, so leftovers can't share

Worst case for DC1 ($C = 10$, $N = 4$)



10 fibers' worth of traffic, but **12 fibers**: the three slivers each need their own fiber.

The fiber budget



$C = 10$ full fibers. Full fibers carry real traffic, so never more than DC1's capacity.

$N - 1 = 3$ partial fibers. At most one leftover per destination.

So 13 fibers cover any traffic pattern. The 3 extra fibers need no extra transceivers.

CAPACITY MOVES WITH DEMAND (OC2: ANY TRAFFIC MATRIX)

DC1 keeps the same total, but re-splits it

Morning: normal mix



To	Demand (fibers)	Fibers used
DC2	4.5	5
DC3	3	3
DC4	2.5	3

Night: big backup to DC2



To	Demand (fibers)	Fibers used
DC2	8.2	9
DC3	1.3	2
DC4	0.5	1

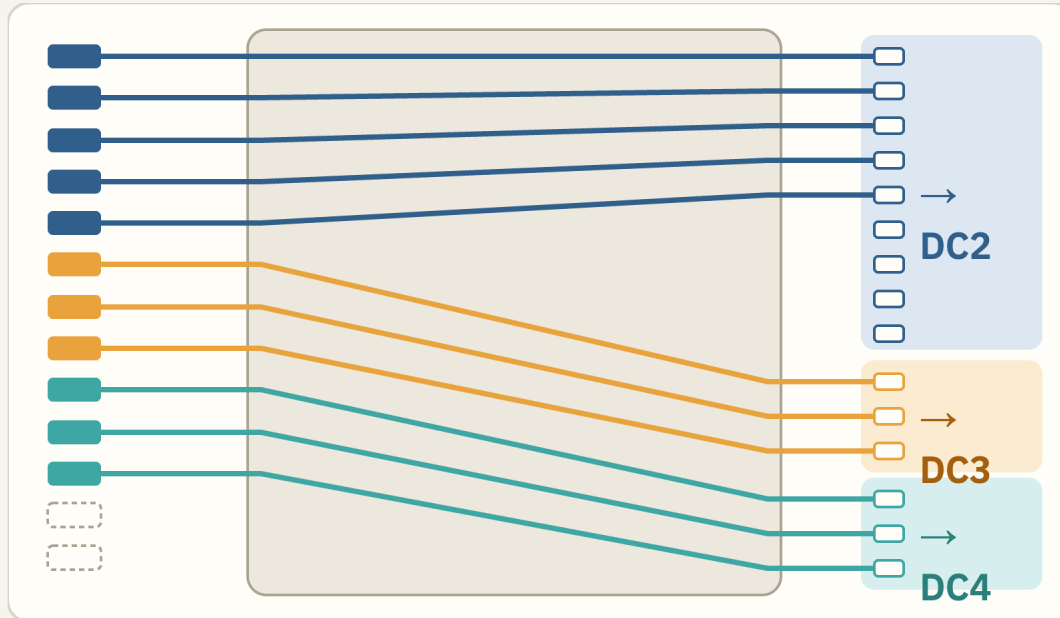
Both times: 10 fibers' worth in total, the same 400 transceivers, out of 13 fibers (10 + one spare per destination). A fixed mesh ready for any pattern would need ~30.

Illustrative numbers; structure from Dukic et al. §3.1 (OC2), §4.3, §5.1

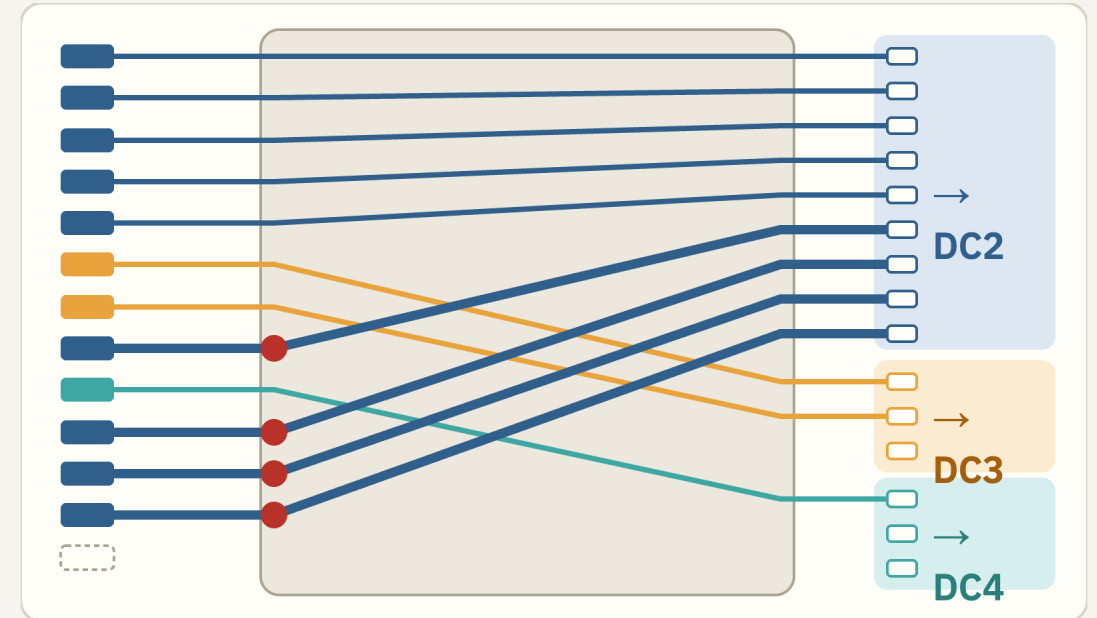
INSIDE THE OPTICAL SWITCH NEXT TO DC1

Re-splitting = re-pointing a few fibers

Morning: 5 / 3 / 3 fibers



Night: 9 / 2 / 1 fibers, 4 re-pointed



Left: DC1's 13 outgoing fibers (dashed = unused). Middle: the switch's tiny mirrors. Right: fibers toward each DC. **Red dots** mark the 4 fibers re-pointed for the night; the other 8 never move.

HOW IT SCALES WITH N DATA CENTERS

The n^2 lands on cheap fiber, not on transceivers

Design	Grows like	Formula	$n = 10, f = 16$
Fixed full mesh: fibers	$n^2 \times f$	$n(n-1) \cdot f$	1,440 fibers
Iris: fibers	$n \cdot f + n^2$	$n \cdot f + n(n-1)$	250 fibers
Electrical full mesh: transceivers	$n^2 \times f \cdot \lambda$	$(n+1) \cdot n \cdot f \cdot \lambda$	70,400
Electrical hubs: transceivers	$n \times f \cdot \lambda$	$2 \cdot n \cdot f \cdot \lambda$	12,800
Iris: transceivers	$n \times f \cdot \lambda$	$n \cdot f \cdot \lambda$	6,400

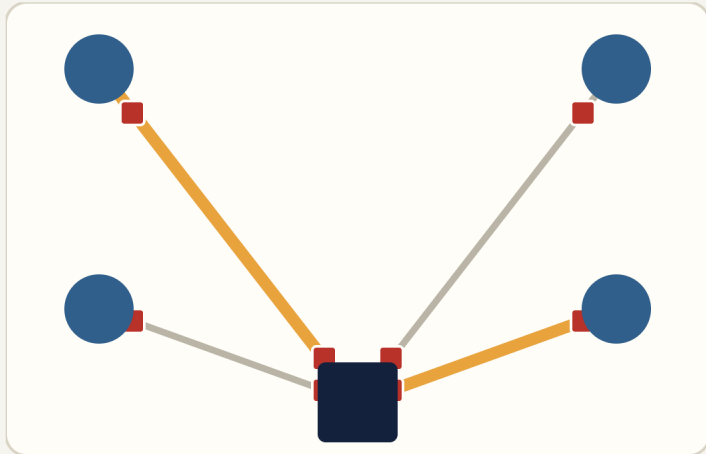
$n \cdot f$ = each DC's own capacity. Switches move it to wherever demand is.

$+ n(n-1)$ = one spare fiber per DC pair for rounding up. Needs no transceivers.

$f \cdot \lambda$ = transceivers per DC ($\lambda = 40$ colors per fiber). Only at the DCs in Iris.

BEFORE AND AFTER IRIS

Same region, three ways to wire it

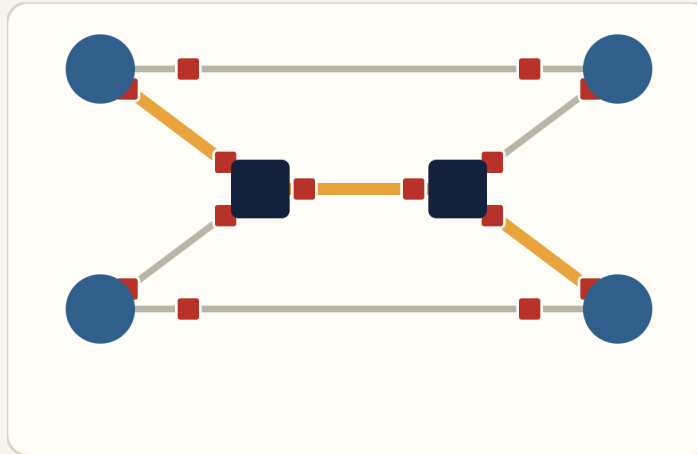


BEFORE · TODAY

Hub-and-spoke, electrical

Cheap and simple, but every trip detours through the hub.

Path: long · Cost: 1×

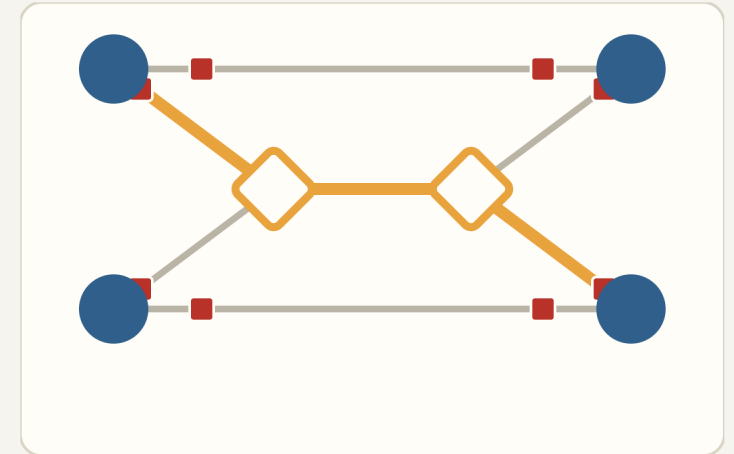


THE OBVIOUS FIX

Distributed, electrical

Short paths, but every hut converts light to bits and back.

Path: short · Cost: up to ~7×



AFTER · IRIS

Distributed, optical

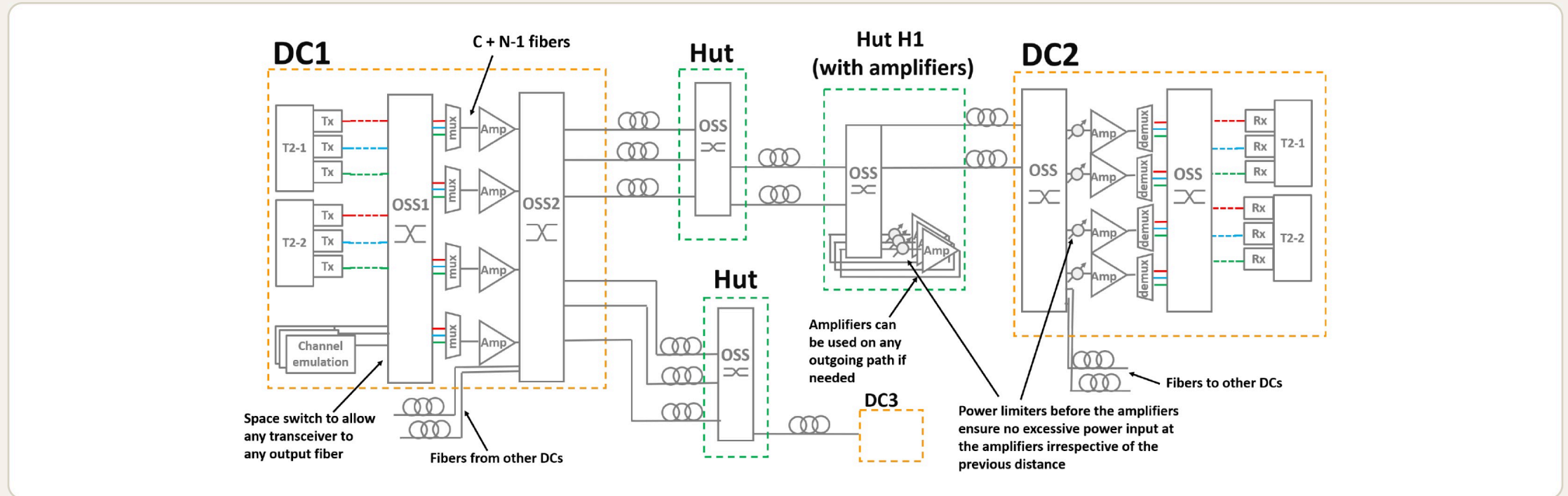
Huts just steer light. Transceivers only at the DCs.

Path: short · Cost: about 1× (at most ~1.1×)

● Data center ■ Electrical switch ◇ Optical switch ■ Transceiver — One DC-to-DC path

Simplified illustration. ~7×: Dukic et al. §2.4, Fig. 7. ~1.1×: claimed in the paper's intro, no figure

The full Iris picture

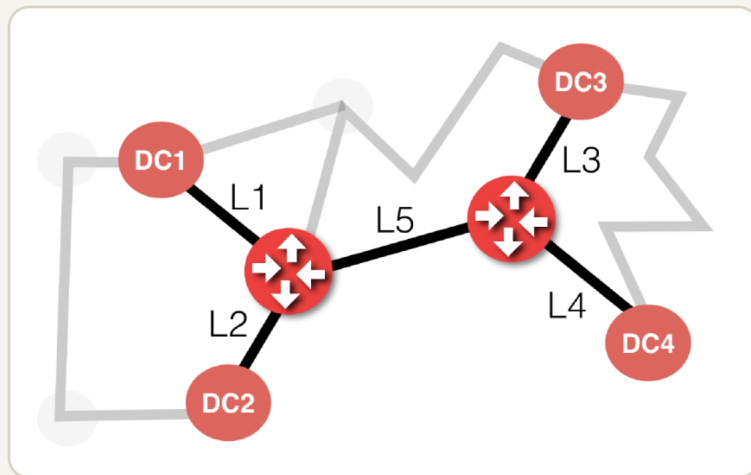


Transmitters (Tx) only in DC1, receivers (Rx) only in DC2. Everything in between is optical switches (OSS), fiber and amplifiers.

A WORKED EXAMPLE

More fiber, far fewer transceivers

4 DCs, each 160 Tbps (10 fiber pairs of 40 colors × 400 Gbps). Two hubs, each serving two DCs, linked to each other.



	Electrical	Iris (optical)
Fiber pairs	60	78 (+30%)
Transceivers	4,800	1,600 (−67%)
Optical switch ports	0	312
Relative cost	2.7×	1×

Transceivers are only needed where data enters and leaves, so Iris pays for a little extra fiber and saves two thirds of the transceivers.

RESULTS

Distributed latency at roughly hub prices

$\geq 5\times$

Cheaper than electrical

Vs. the same distributed design with electrical switches. In 80% of 240 setups; about 7× in the median.

1.1×

Of hub-and-spoke cost

Vs. today's hubs. Stated in the paper's intro; no figure shows this comparison.

$>2\times$

Cheaper, even with backup

Vs. distributed electrical. Iris survives 2 fiber cuts; the electrical design survives none.

Testbed: the optics work

Real switches, amplifiers and 5–50 km fiber spools. Path changes take 50–70 ms; error rates match a fixed link.

Simulation: apps barely notice

Even with big, frequent traffic shifts, the slowest 1% of transfers are at most 2% slower.

The cost curve and the real testbed

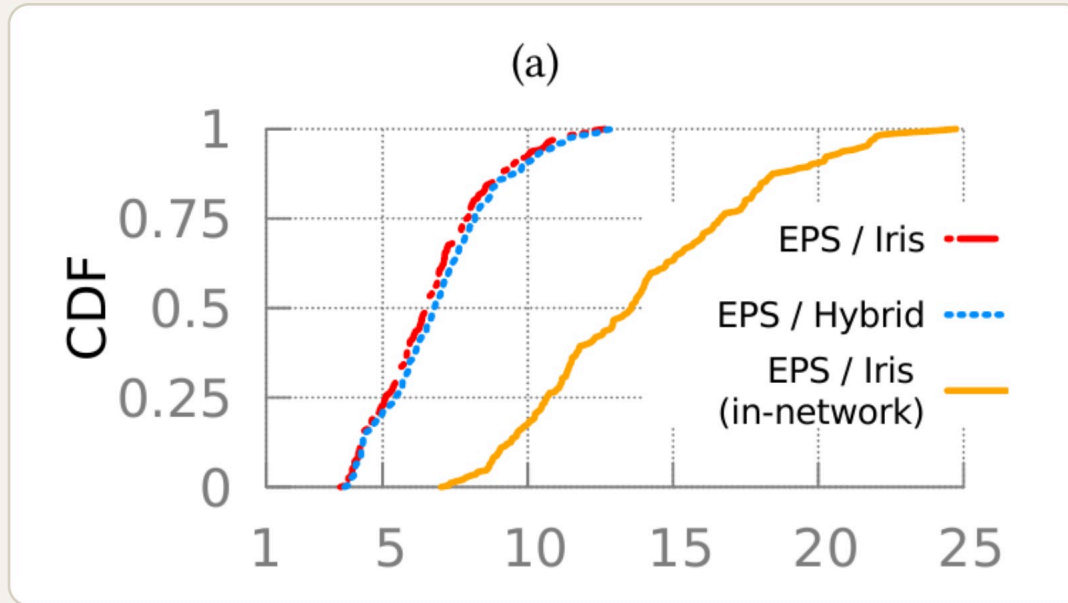


Fig. 12a: how many times pricier electrical is than Iris, across 240 setups. Red and blue overlap: wavelength switching adds almost no savings.

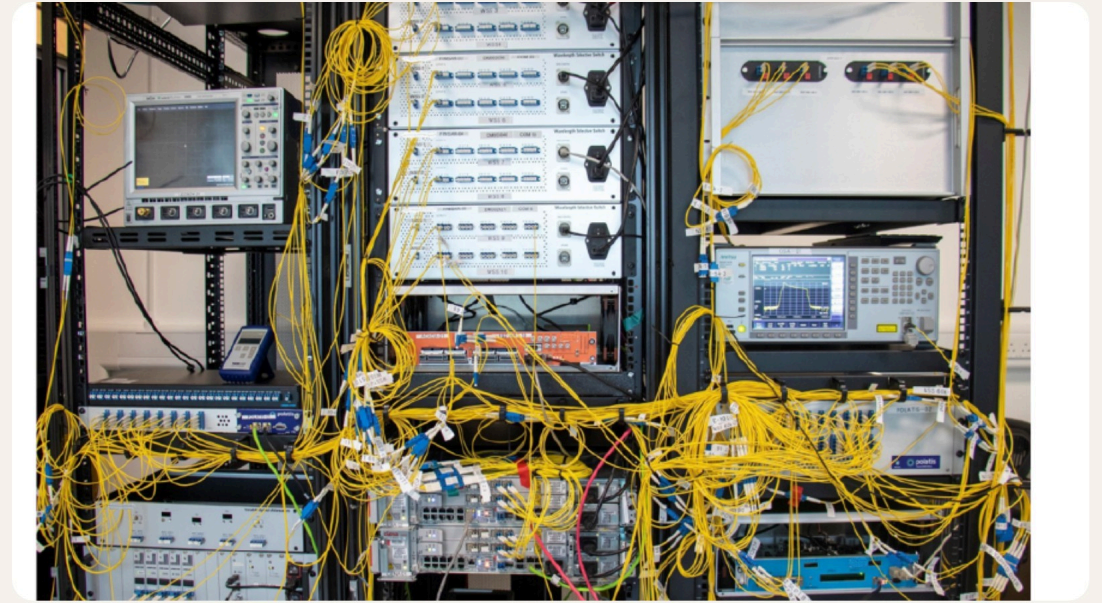


Fig. 13a: the testbed. Real optical switches, amplifiers, transceivers and spools of 5–50 km fiber.

ASSESSMENT

What the paper does well, and what to question

Strengths

- Names a real, new problem with real Azure data
- One simple idea: trade cheap fiber for costly transceivers
- Off-the-shelf parts and no fast, fragile control loop
- Checks optics, cost and application impact

Open questions

- Small testbed and simulation, not a deployment
- Real prices are hidden; the result depends on them
- Assumes DC-to-DC traffic changes slowly
- Still many sites with equipment to run and repair

Fast optical switching, untested at scale

In production: slow

Optical circuit switches already run in cloud and ML systems. They take milliseconds to change, which suits slowly changing traffic.

In research: fast but tiny

Microsecond and nanosecond switches exist. Prototypes have under ten ports, often custom transceivers, and many need network-wide traffic data.

Missing: a real test

Would fast optical switching work at practical scale, with commodity servers and standard TCP? Yield, timing and compatibility were open.

This paper: a 128-port rotor switch, 16 servers, and an unmodified Linux stack.

Fast or compatible: rotors aim for both

Optical switch type	Time to change paths	Works with any link or rate?
Tunable laser + wavelength router	Nanoseconds	No: needs a fixed wavelength plan
Semiconductor amplifier (SOA)	Nanoseconds	No: wavelengths interfere
MEMS mirrors	Milliseconds	Yes: moves light of any color
Rotor switch	Microseconds	Yes: moves light of any color

Logical transparency

Acts like a packet network. This depends on how fast the switch can reconfigure.

Physical transparency

Ignores link type and data rate, so one switch can serve many generations of optics.

Cycle through fixed links; never schedule



A matching is a fixed pairing of inputs to outputs. Over one cycle, every node gets a direct link to every other node once.

No scheduler

The rotor repeats the same few pairings forever. Nobody collects traffic demand. The price: up to half the ideal bandwidth.

Bulk traffic

Waits for the slot when its destination is linked, or takes one extra hop to spread load. Gets 50–100% of ideal, however skewed.

Urgent traffic

Pairings are chosen so live links always form a well-connected (expander) graph. A short multi-hop path always exists.

The rotor switch is a spinning disk

128

Ports in a 3U rack box

Every port worked: 100% yield on all 256 fibers.

7 μ s

Worst-case switch time

About 1,000 $\times$ faster than MEMS mirror switches.

10G

Per port, off-the-shelf optics

Commercial transceivers, no amplifiers needed.

How it switches

One disk, patterned in sectors, bends all 256 light beams at once. Spinning from one sector to the next changes the matching.

Why it scales

No moving part per port. The pairings are passive grooves in metal, and the motor is a repurposed hard-drive spindle.

Making it work with ordinary servers

Shared clock

PTP over the normal GbE management port keeps hosts within 1 μ s. A control loop locks the disk's spin to the same clock, to $\pm 5 \mu$ s.

Smart NIC

An FPGA NIC sends each packet only in its timeslot, using the clock and the fixed schedule. Linux, TCP and ping run unmodified.

Defect masking

Flaws in the disk cause short dropouts at fixed, repeating times. The NIC stays silent in those slots, like bad-sector mapping on a hard drive.

89%

WITHOUT THE MASK
6,780 TCP retransmissions

99.5%

WITH THE MASK
0 TCP retransmissions

RESULTS

Near link rate, with ordinary Linux

99.5%

Of achievable TCP rate

Direct-link transfers, with defects masked. Switching overhead is about 2%.

93%

Of link rate, multi-hop

9.3 of 10 Gb/s using all four rotors. A few retransmissions come from packet reordering.

0

Packets dropped

Even when bulk TCP traffic shares paths with latency probes.

Latency matches a normal NIC

Median ping is comparable, and the tail past the 99th percentile is better. Under heavy load the worst 15% rise by about 15 μ s.

The testbed is small

16 servers on one switch, at 10 Gb/s. At 25 Gb/s, about a quarter of the 512 connections performed poorly.

What the paper shows, and what to question

Strengths

- A real 128-port switch, not a mock-up
- No scheduler and no traffic measurements
- Commodity optics and an unmodified Linux stack
- Turns a hardware flaw into a software fix

Open questions

- 10 Gb/s links and 16 servers, not a data center
- Bulk load balancing is not built yet
- Needs precise clocks and custom FPGA NICs
- No production deployment, and up to 2× bandwidth lost by design

Questions for the class

- 1 Iris switches in about 20 ms between data centers. RotorNet switches in microseconds inside one. What does each design give up?
- 2 A demand-oblivious network can lose up to half the bandwidth. When is that a fair price for simplicity?
- 3 The disk's defects were hidden by the NIC rather than fixed in manufacturing. When is pushing a problem to the edge a good call?
- 4 Would you deploy this for AI training clusters, where traffic follows a few known patterns?

Pod-to-pod bandwidth: costly or inflexible

Full bandwidth: costly

A non-blocking core for 64 pods needs 1,024 switches and 65,536 wires. Optical transceivers alone would cost \$26M.

Oversubscribed: inflexible

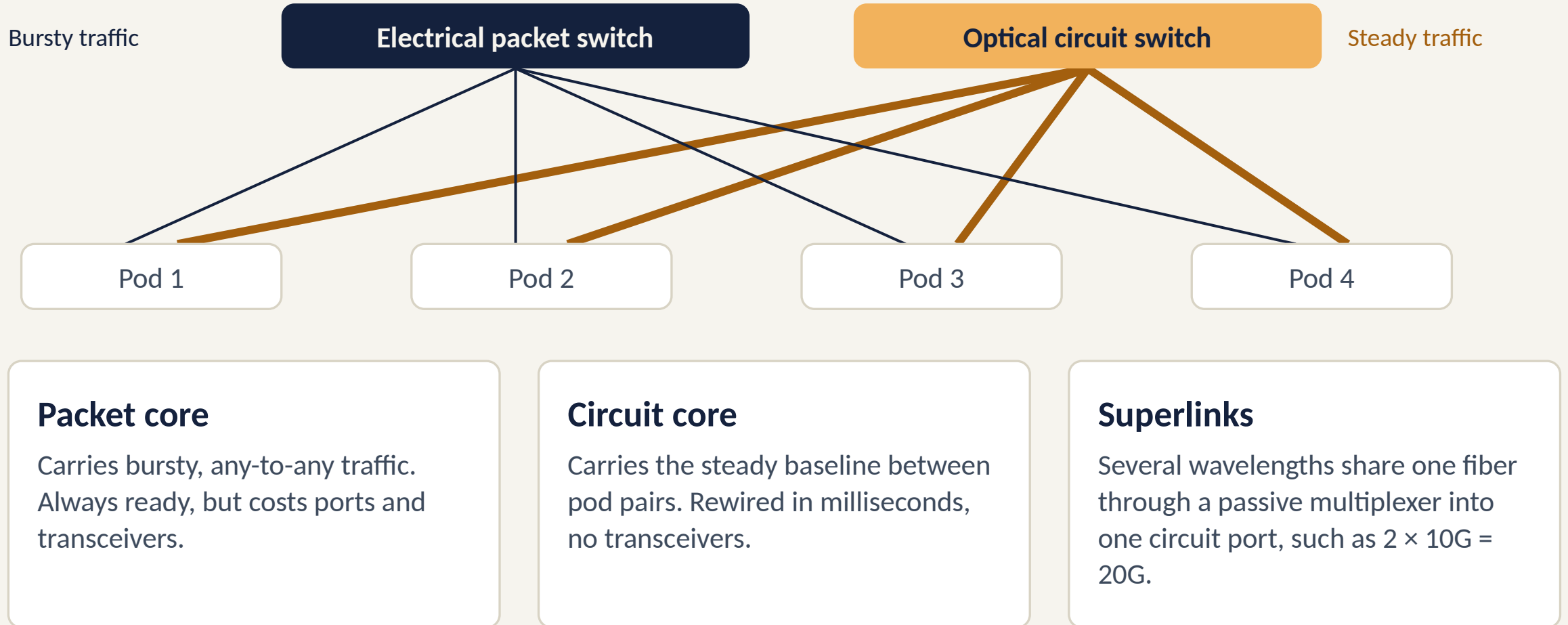
With 100 uplinks for a 1,024-server pod, worst-case traffic gets 1 Gb/s per server. The fixed capacity cannot move to where it is needed.

The opening: aggregation

Traffic summed over a whole pod shifts more slowly than traffic from single hosts, so optical circuits can carry part of it.

Helios: a core with both packet and circuit switches, rewired as traffic shifts.

A two-level tree with two kinds of core



Circuit ports match cost and cut power

Component (2010 prices)	Cost	Power
Packet switch port	\$500	12.5 W
Circuit switch port	\$500	0.24 W
Optical transceiver (up to 8 wavelengths)	\$200	1 W
Fiber	\$50	0

3× lower cost

About \$40M saved at 64 pods of 1,024 hosts, even with circuits that never change.

9× lower power

About 800 kW saved at the same scale.

6.5× fewer ports

About 55,000 fewer ports, a stand-in for cables and upkeep.

A control loop rewires the core

1 Measure

Read flow counters on pod switches. Ignore small “mice” flows under 15 Mb/s.

2 Estimate

Work out what TCP flows would get on an ideal switch, not what today’s topology allows.

3 Compute

Pick circuits with maximum-weight matching, one circuit switch at a time.

4 Notify down

Pod switches move traffic off circuits about to disappear, avoiding black holes.

5 Change

The controller tells the optical switch to move its mirrors.

6 Notify up

Pod switches add the new circuits back into their forwarding groups.

Steps 4–6 run only when the topology changes. Hosts need no changes.

Ordinary switch features slowed the prototype

43

Gb/s as first built

A 2 s “debouncing” delay left flows idle after each circuit change.

87

Gb/s, debouncing off

Next bottleneck: a 600 ms signal-cleanup step (EDC) in the optical PHY chip.

142

Gb/s, EDC off too

The all-electrical tree reached 171 Gb/s on the same test.

Optical switch time

The mirrors moved in about 12 ms (25 ms advertised).
The PHY then needed about 15 ms to lock.

Control loop time

About 96 ms to measure and plan, plus 168 ms to change topology, mostly switch software.

RESULTS

Near electrical when traffic is stable

142

Gb/s vs 171 electrical

Same 4-second test, with Helios using one-way circuits.

2×

One-way vs two-way circuits

When pod demand is lopsided, one-way circuits used capacity twice as well.

265 ms

Per topology change

About 96 ms plus 168 ms. The authors see room to get well under 100 ms.

Stability decides everything

Throughput rises as flows last longer, tested from 0.5 s to 16 s. Gradual shifts beat abrupt ones.

WDM costs no throughput

Two wavelengths per superlink matched one, so it can cut cabling and cost.

What the paper shows, and what to question

Strengths

- First working hybrid core, from commercial parts
- No changes to hosts or switch hardware
- Simple split: circuits steady, packets bursty
- Honest lessons about slow, built-for-static gear

Open questions

- 24 servers; the scale results are simulated
- Needs pod-level traffic stable for seconds
- Central controller with a loop of about 0.26 s
- Cannot split one destination over both cores

Questions for the class

- 1 Helios needs pod-level traffic to hold steady for seconds. Is that still true of today's workloads, such as large ML training jobs?
- 2 Iris and RotorNet drop the packet network from the core. What does Helios's packet side buy, and what does it cost?
- 3 Helios needed a central controller reading flow counters every cycle. Which of the three designs would you rather operate?
- 4 Debouncing and EDC were built for static links and broke fast circuits. What other assumptions in networking gear might break?

Three ways to build the core

ELECTRICAL

Read every packet

Baseline in all three papers, built from packet switches

- Handles any traffic, bursts included
- Ports and transceivers at every hop
- High power; faster links mean new hardware

OPTICAL

Steer whole beams of light

Iris (about 20 ms) and RotorNet (7 μ s)

- Cheap, low power, ignores link speed
- Blind to data: needs a controller or a fixed cycle
- Pays in spare capacity (fiber or bandwidth)

HYBRID

Packets for bursts, circuits for the rest

Helios: a packet core plus an optical circuit core

- Cheaper and lower power than all-packet
- Works when pod-level traffic is stable
- Two networks to run, plus a controller

Electrical, optical and hybrid compared

	Electrical	Optical	Hybrid
Decides by	Reading each packet	A schedule or controller, never the data	A controller watches traffic, then rewires circuits
Path changes	Per packet (nanoseconds)	20 ms (Iris) down to 7 μ s (rotor)	Milliseconds for circuits; packets are instant
Cost	Port and transceiver at every hop	Cheap, no transceivers	Between the two; about 3 $\times$ below all-packet
Power	High	Very low	Up to 9 $\times$ below all-packet (simulated)
Sudden shifts	Handled well	Poor, unless aggregated or cycled	Packet side absorbs them
Faster links later	Replace hardware	Unchanged	Optical side unchanged