A Brief History of Router Architecture

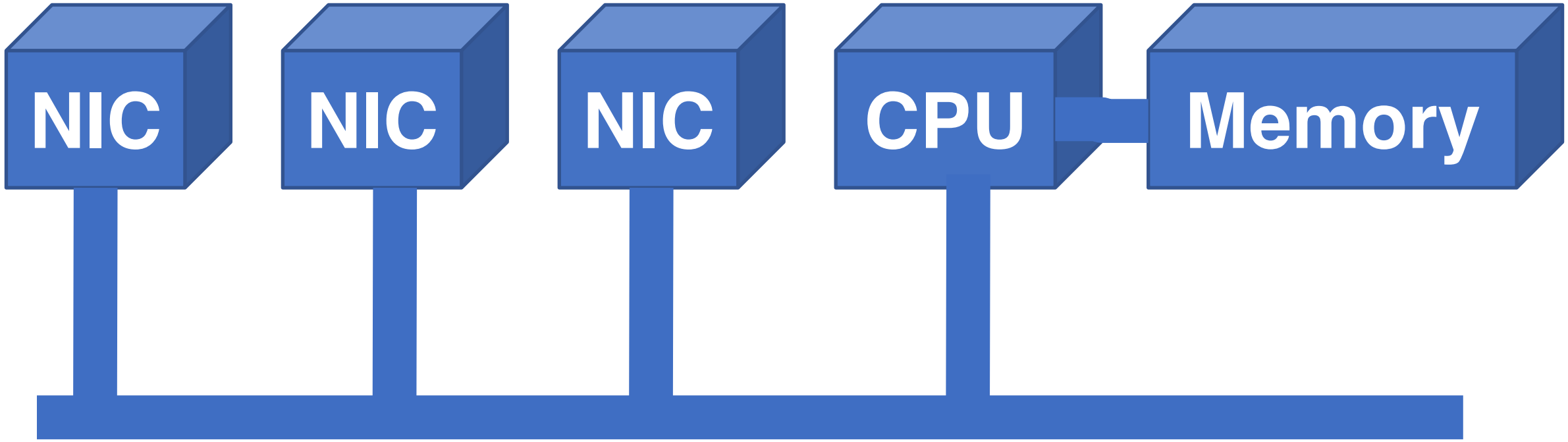
Tony Li

ARISTA

Disclaimer

- The goal of this talk is education and perspective — not dumping on anyone, innocent or guilty.
- This is all based on non-ECC protected neurons, and details are blithely ignored. Omissions and errors are all mine.

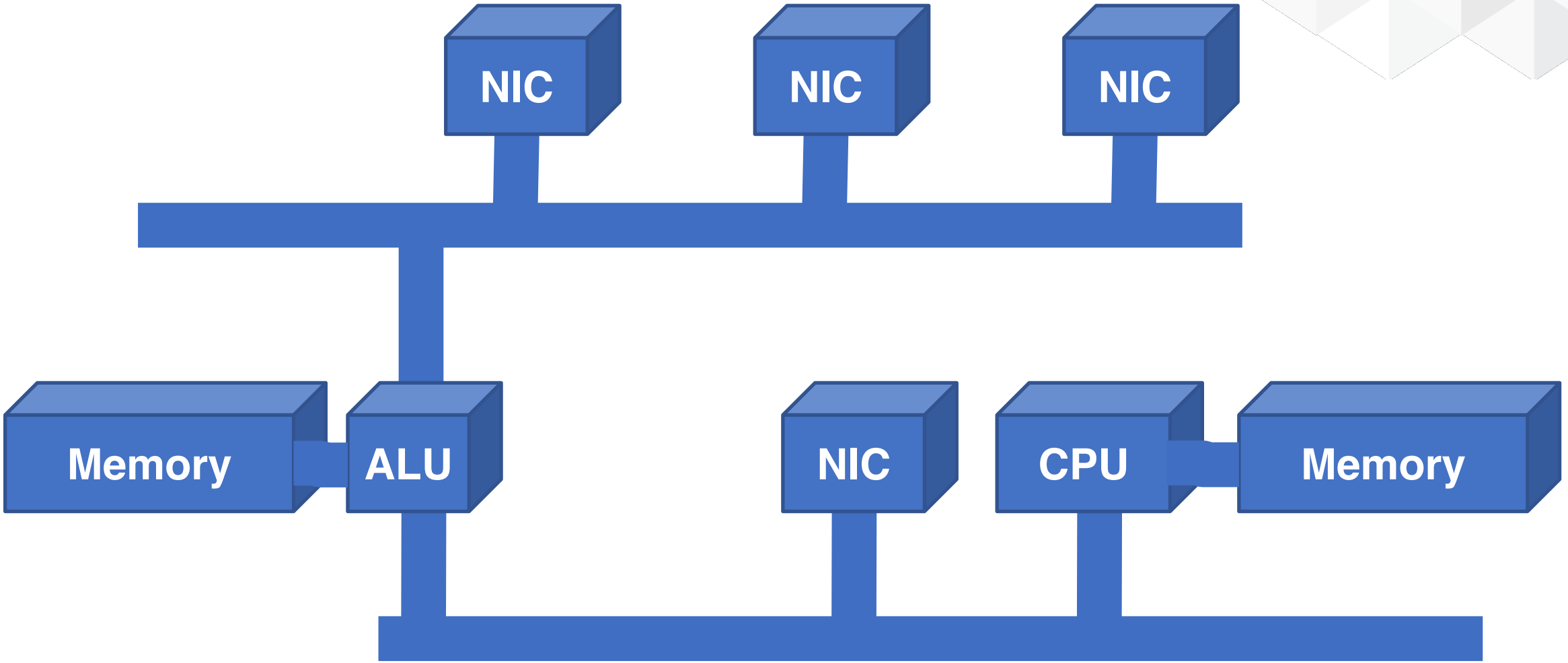
In the beginning, there was a bus …



… and it sucked

- Bus and CPU are centralized resources that scale linearly
- Cost of the bus interface is proportional to the bus speed and the bus speed is the aggregate for the entire system
  - Bigger box? More expensive NICs!
- We need a scalable architecture

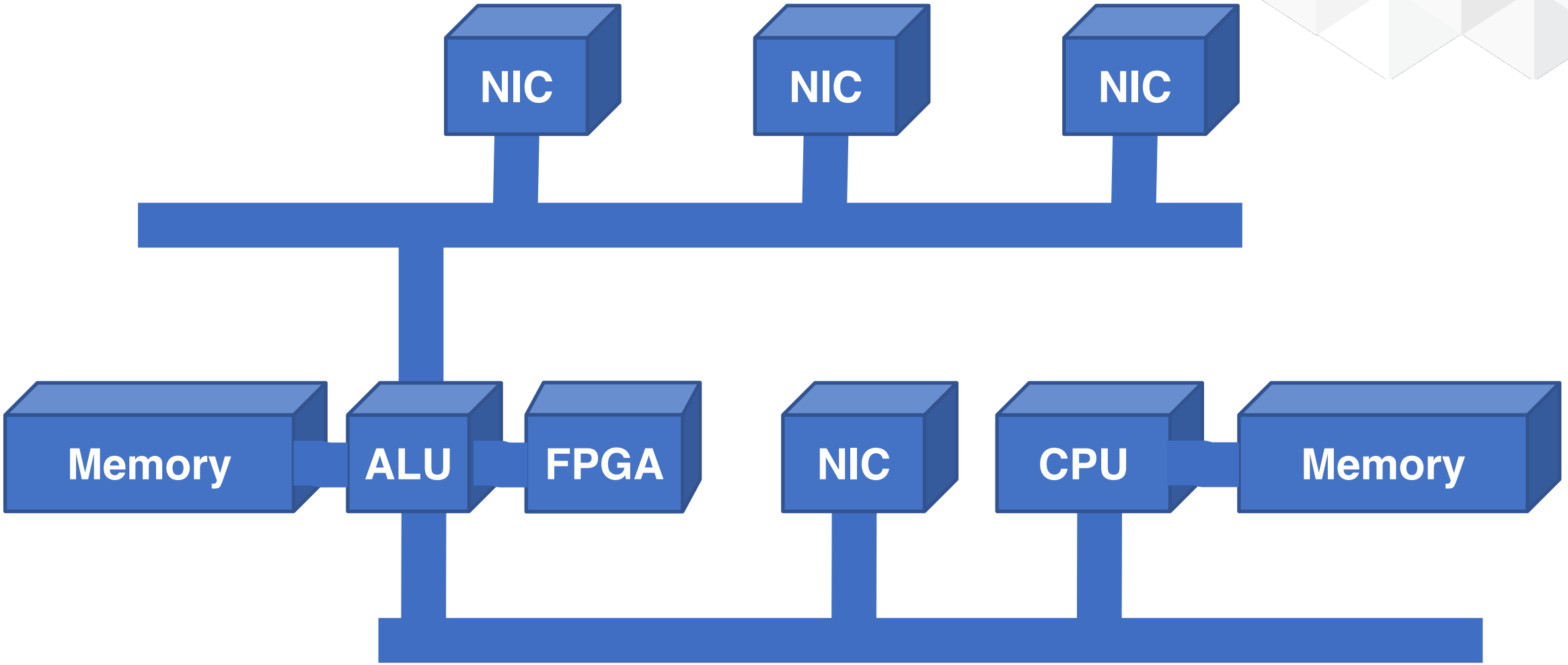
If one bus is good…



… two should suck a little less

- A second, faster bus gives more bandwidth
- An ALU (DSP) gives a few more cycles per packet
- But it still doesn’t scale

What if we add hardware acceleration?



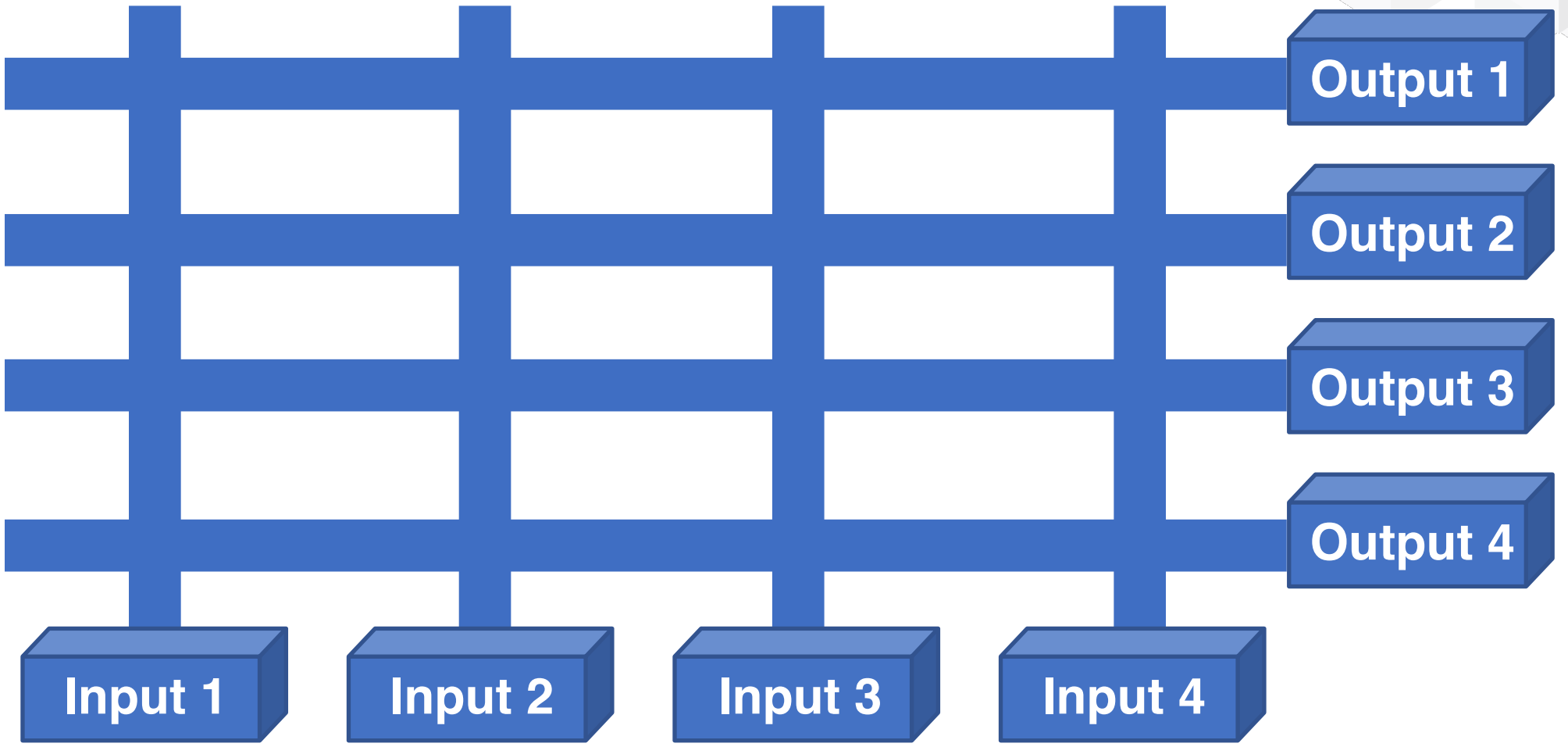
Better, but it still sucks

- Proof-of-concept that you can do packet forwarding in hardware. (So we abandoned that direction.)
- But the IP address lookup is NOT the only bottleneck, so adding lookup hardware only helps a little.
- The real issue is centralized bandwidth. We need distributed bandwidth and processing.

And then, the web…

- Carriers buy up NSFnet regionals
- Real Money starts to flow — time to get serious
- Everyone and their brother wants to build a router
- ASICs become credible
- Creativity blossoms

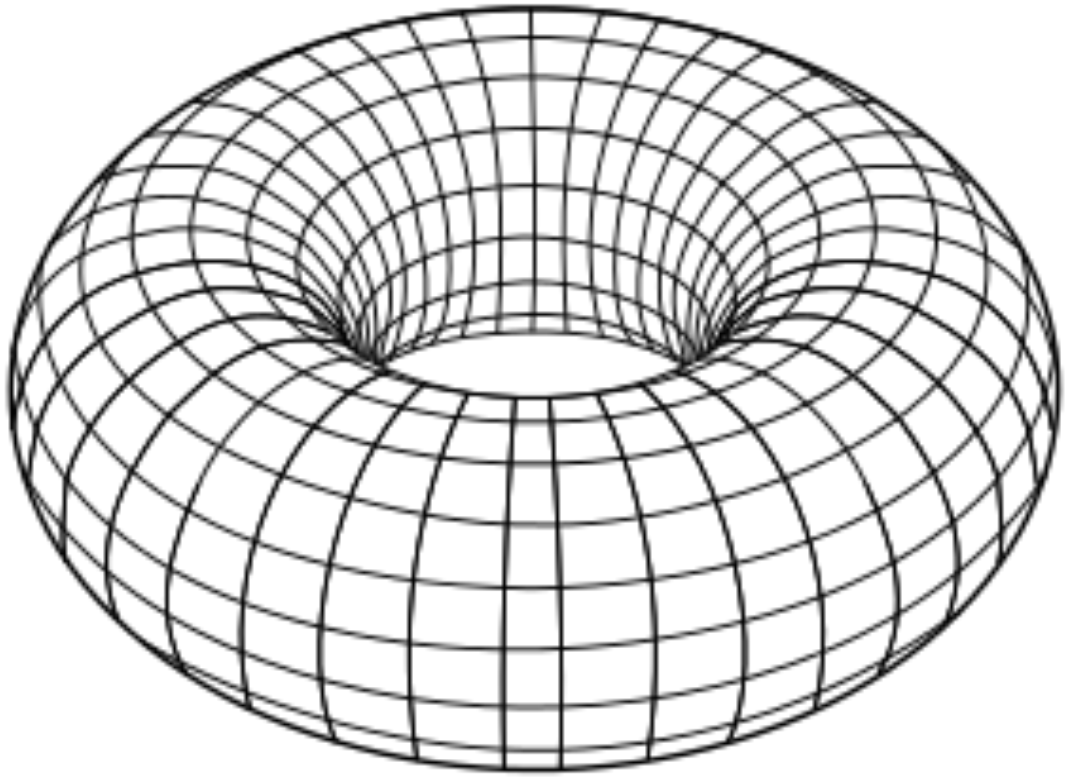
The scheduled crossbar



Distributed bandwidth, but it still sucks

- Scheduling is hard. Contention for outputs means that the switch lanes have to be much faster than the outputs.
- Worse, because you can't always drain the inputs, you end up with Head-Of-Line-Blocking (HOLB). Throughput suffers.
- Inputs need a queue per output.
- And it doesn't scale.

What about a torus?

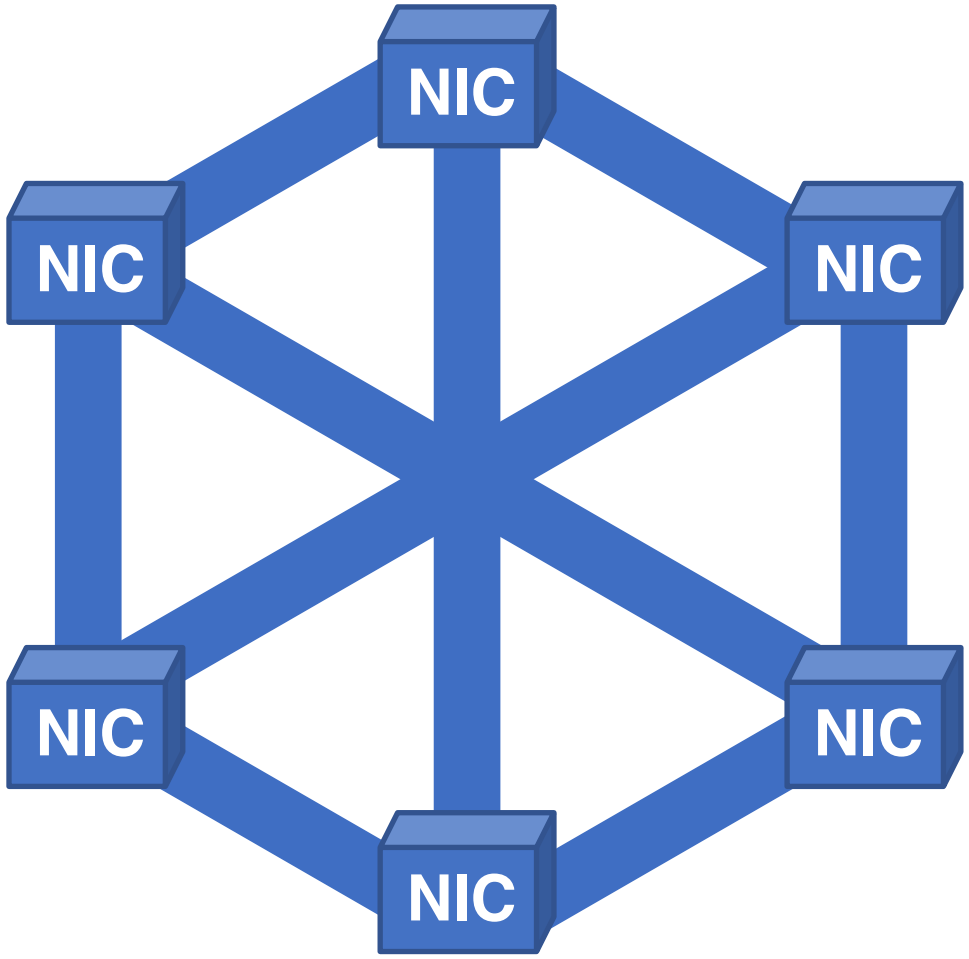


Donuts make lousy routers

- Non-uniform bandwidth means that the fabric can congest depending on the traffic pattern.
- Card removal causes more bandwidth issues.

- Bandwidth needs to be distributed and **uniform**

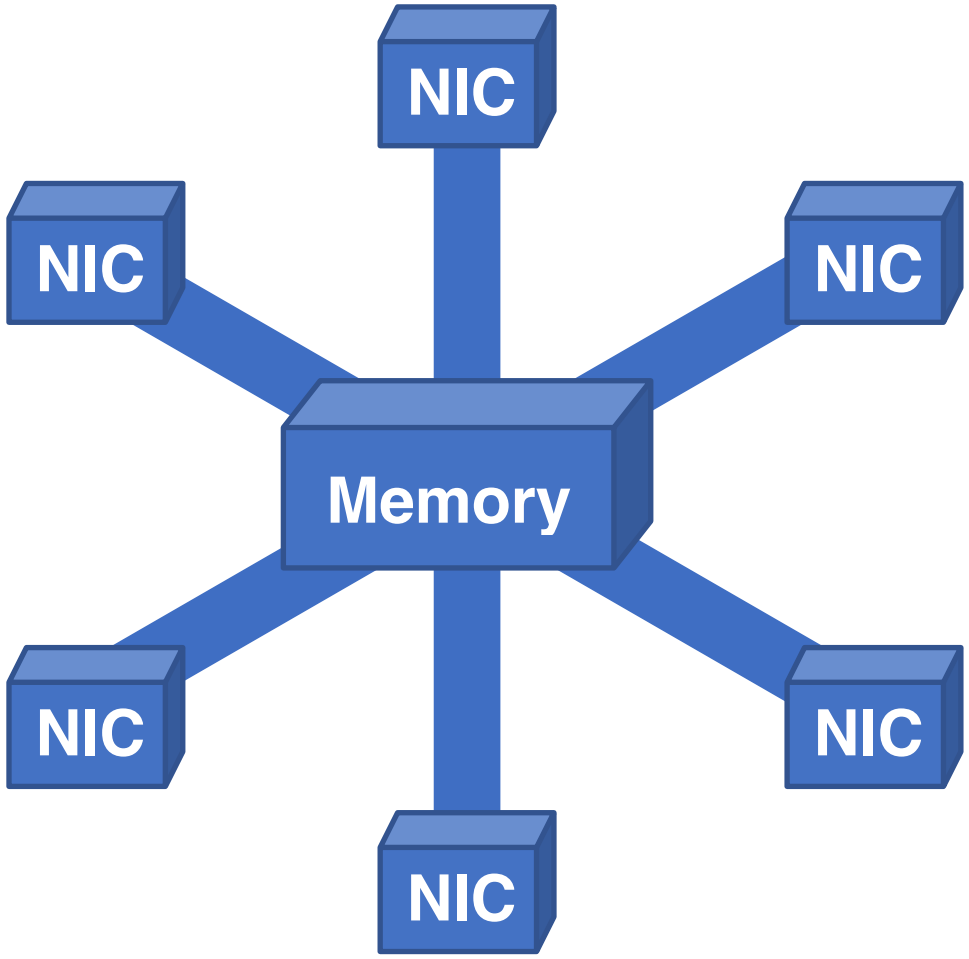
Distributed cell memory and a full mesh?



That still sucks…

- When a card is hotswapped, all packets are lost.
- Bandwidth of system is a fixed multiple of the bandwidth of the card.
- That's still not scalable.

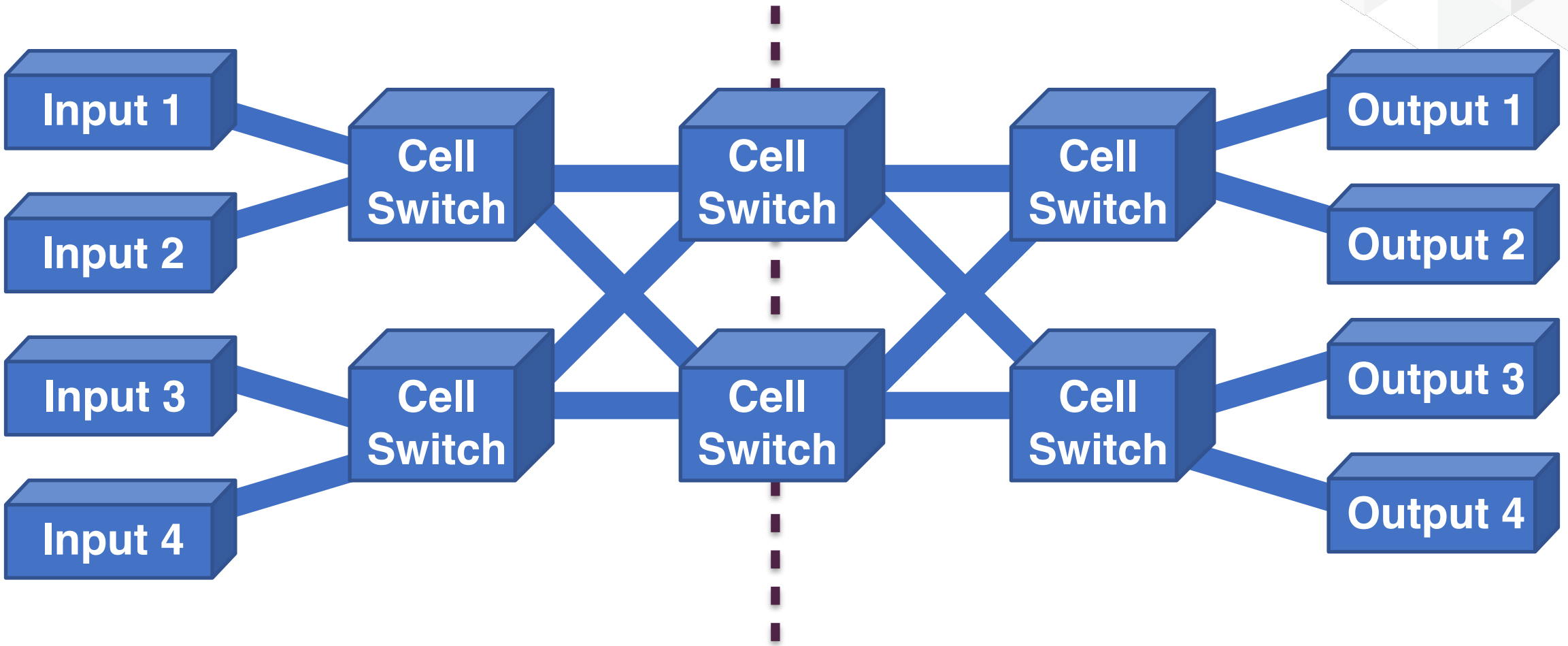
Centralized shared memory?



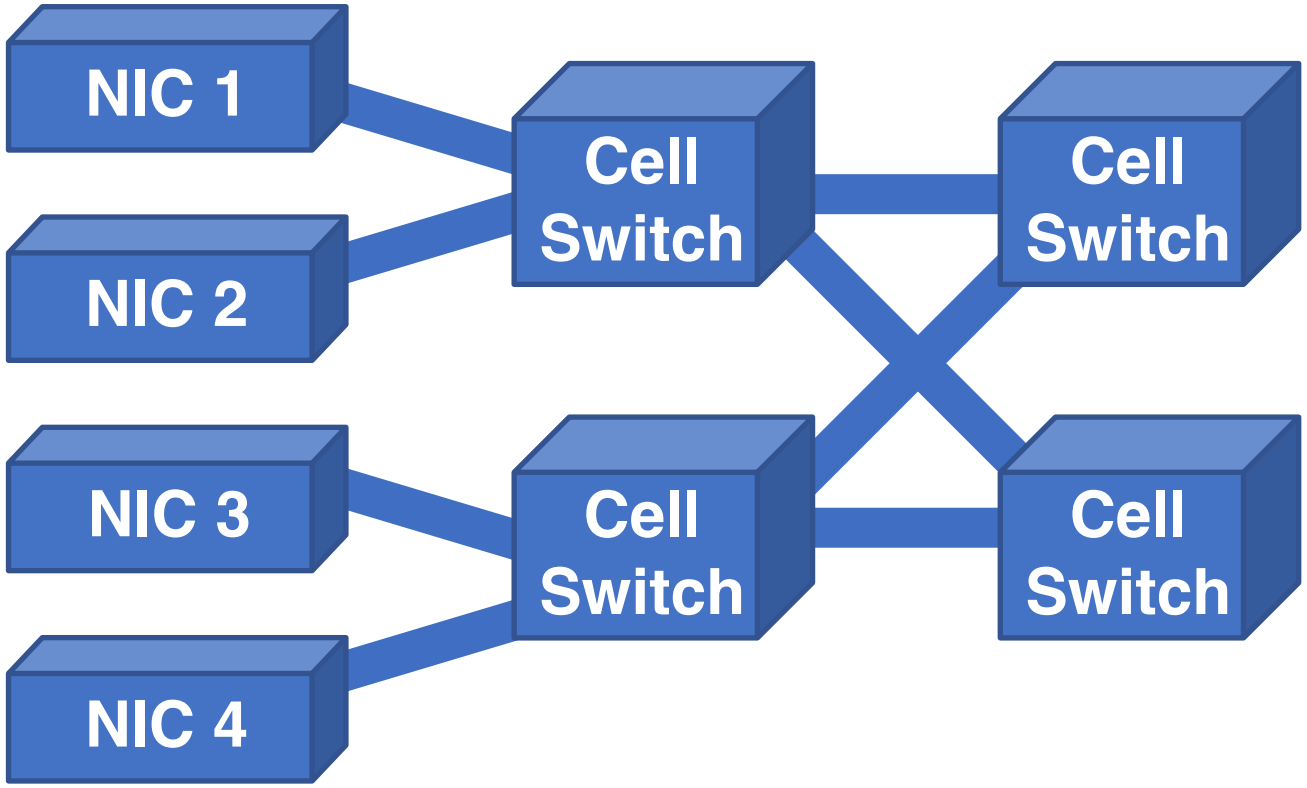
Better, but still sucks

- Lots of nice properties, but
  - It can’t be just one memory, need many banks
  - Need multiple memory controllers
  - Limited by controller bandwidth
- It still doesn’t scale

Cell based Clos networks



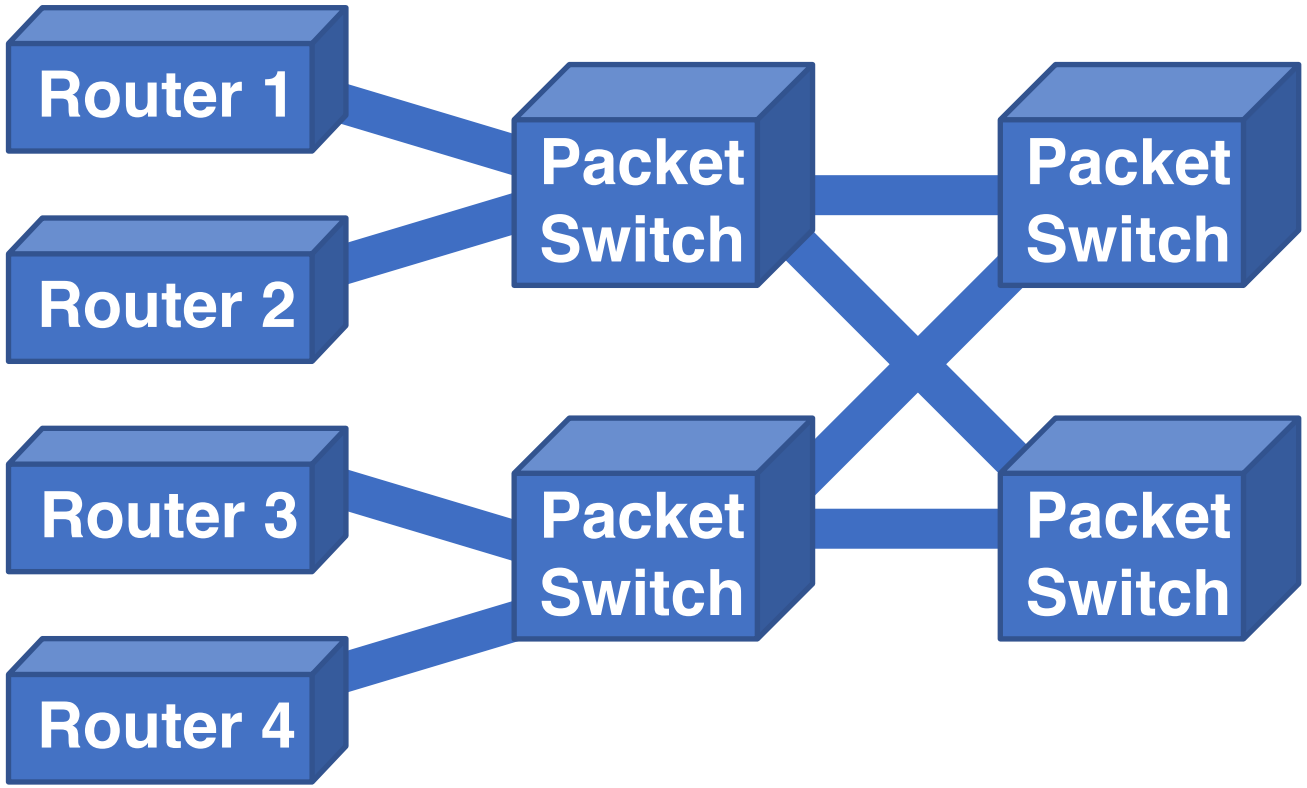
Folded Clos networks



That sucks less

- Bandwidth is distributed, uniform, and redundant
- But scalability is still limited:
  - Inputs need a queue per output
  - Cell addressing is finite: fabric can only be so big
  - Cell fabric is proprietary and fixed
  - Technology upgrades are difficult to roll-in
- Vendor lock-in or chipset lock-in are issues

Supernode architecture



- Same Clos topology
- Industry standard links (Ethernet) and switches
  - Easy incremental upgrades
- True vendor independence
- Packet, not cell based, so needs a bit more internal bandwidth
- Fabric scales with more stages or more width

Supernode software issues

- Needs to look like a single router on Control & Management planes
- Work in progress:
  - IGP abstraction - Supernodes looks like a single IGP node
  - Management plane abstraction - Supernode looks like a single node to top level management