Towards Modern Networking System

现代网络系统结构展望

2026.8.15

Zhihao Wang <wangzhihao9@hotmail.com>

I Behind the Monolith II Inside the Monolith III Beyond the Monolith

Contents

Core Vocabulary

术语表 A · 对象与结构

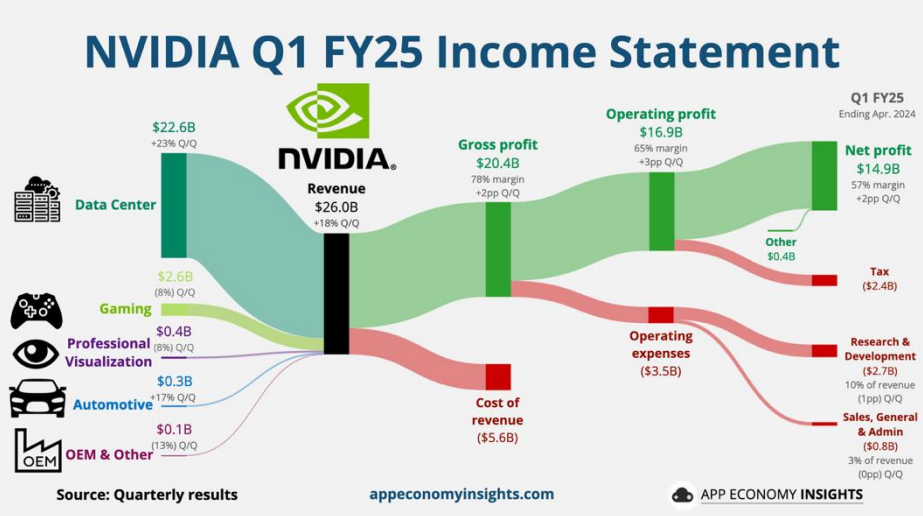
术语	一句话	注解
Management Domain	最终裁决的资源范围	≠ 一致性域；一个域可含多个一致性岛
Software Connection	两端点间的管理关系：一对映射 + 一对 ACL	建连=写表、断连=清表、运行=查表；不绑定传输机制
Reliable Tunnel	NIC 内共享的包级可靠传输资源（PSN 窗口、ACK/NAK、限速）	不是封装隧道；不承载事务语义
Physical Path	报文实际经过的物理路线	Path 是承载 Tunnel 的物理资源，映射可变
Move	完整的搬运任务（数 MiB 乃至更大）	由 DMA Core 拆成有界事务；无单一完成语义
Transaction	有稳定身份、有限大小、一次退休的工作单元	不是数据库原子事务；大小有协议上限
Transaction Fragment	**事务内部一段自描述、幂等、可独立放置的片段**	跨 Tunnel 重放身份不变；片段不同于某一次实际发包
Instance	一个 Fragment 的一次发送执行	换一条 Tunnel 重传，即为一次新实例；{TunnelID, PSN} 标识
DMA Context	以 PASID 为粒度的调度与策略单元，跟踪事务语义状态、源数据义务与退休	调度按租户不按连接；QP 是它的兼容外壳
Aperture	**远端资源映射到本地地址空间的一段窗口，其归属节点是 NIC**	OS 按设备空间分配，无共享页/别名

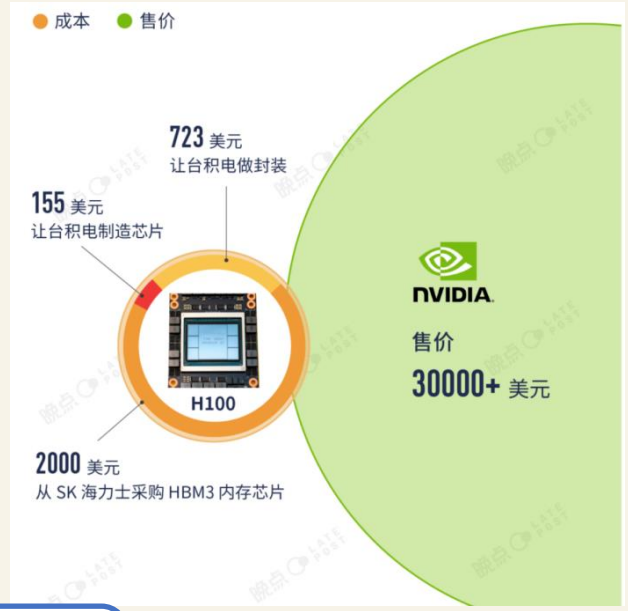
Core Vocabulary

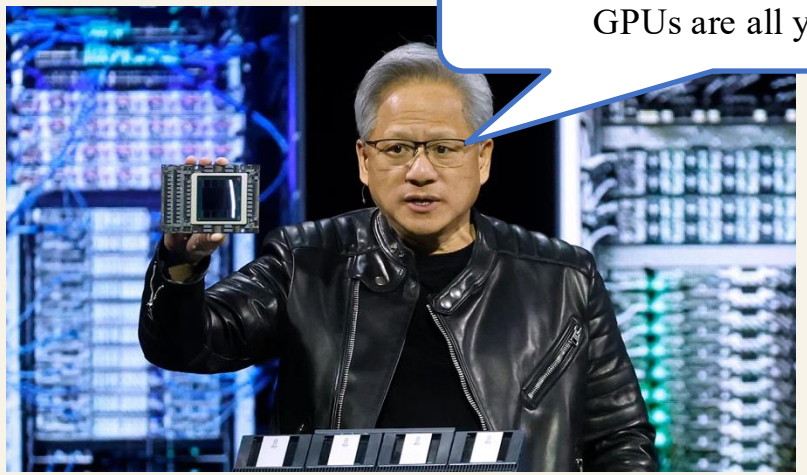
术语表 B · 地址、完成与聚合

术语	一句话	注解
Egress Mapping	把 {PASID, IOVA} 映射为 {DstNode, DstIOVA} 的出口查表	权威表项在 Host DRAM，NIC 只持有缓存
Direct Mode 直通	远程永远是权威；本地写进 NIC 即成为远端事务	PCIe 场景下的唯一模式
Mirror Mode 镜像	本机永远是权威；远端只是投机性副本	单写者；允许一对多扇出；依赖 CXL/CHI 一致性接入
Local Retirement	收齐全部 Fragment 的 ACK，即告退休，解除源数据义务	完成与否只由发送端判定，远端没有事务概念。确认收货前，发送方须保留源数据。
Terminal Result	成功（收齐 ACK）/ 确定拒绝 NAK(reason) / 未知 TRANSPORT_FAILURE	超时不是结果；「未知」不等于「远端未执行」
Send Fence	发送侧的串行化：前一事务发射完毕，后继事务才发射	保序发射，乱序执行
Execute Fence	**接收侧的串行化：发送端记录依赖，接收方据此恢复顺序**	乱序发射，保序执行
Aggregation Engine	占用一段地址空间的流量聚合/分散点	网络层次化的节点，把一对一的通信关系组织成图

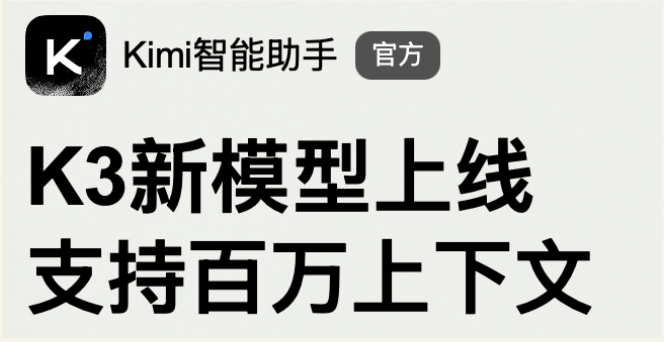
Many Perspectives on NVIDIA GPU



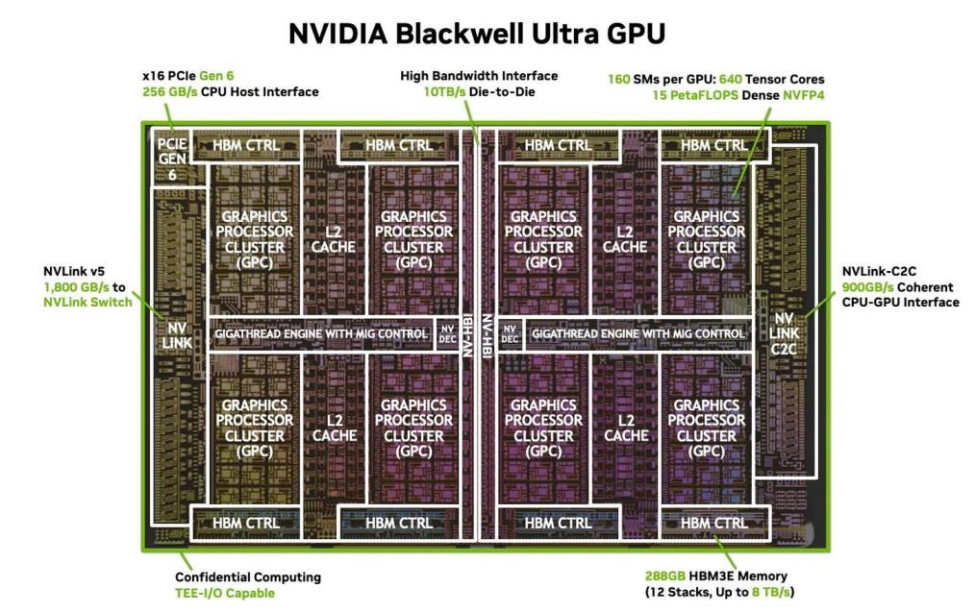


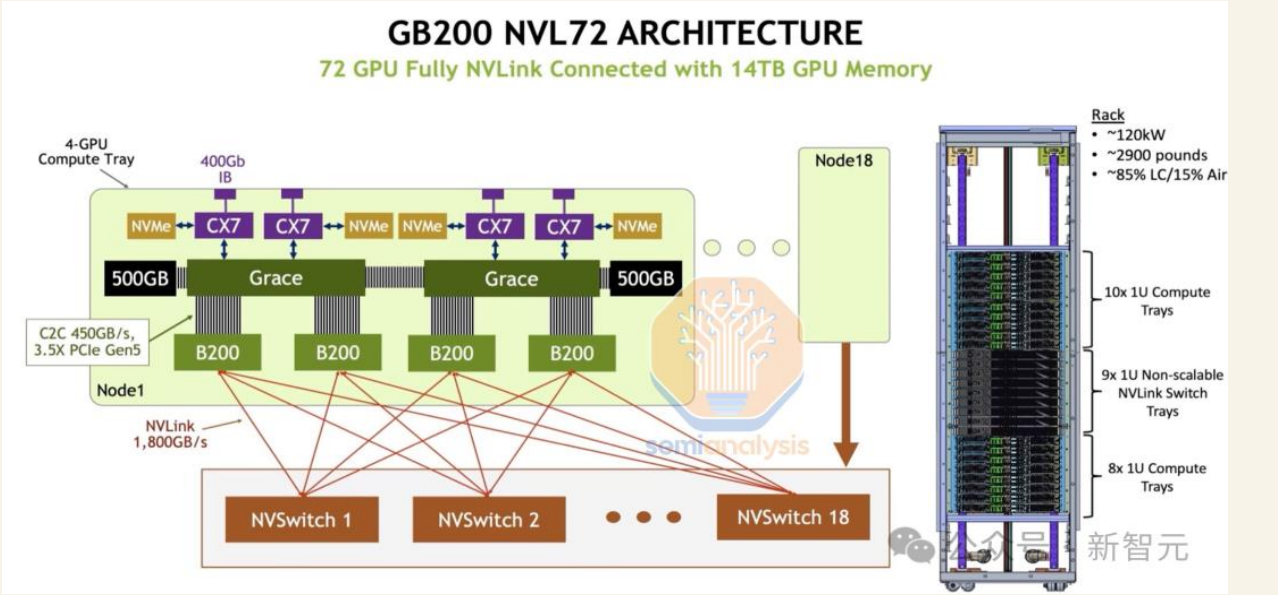


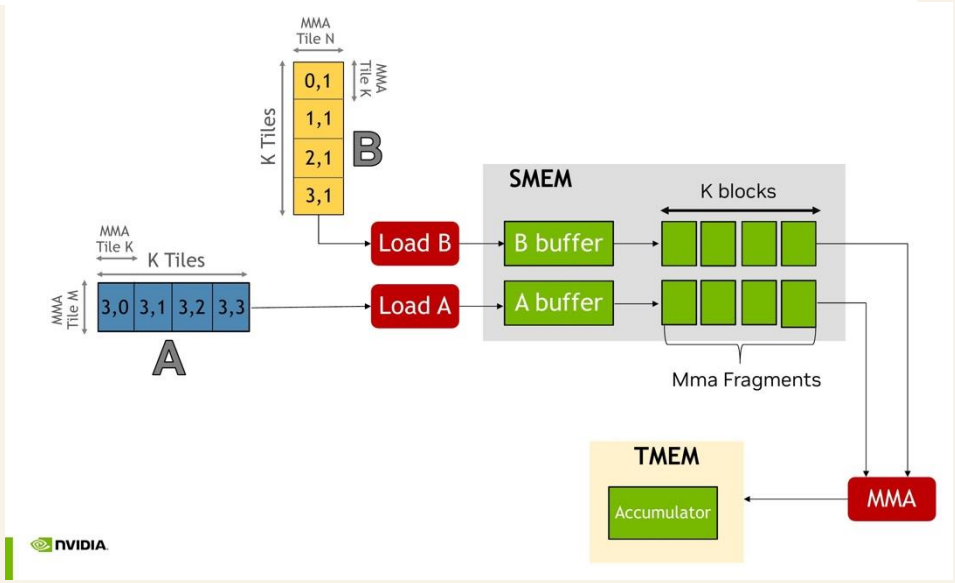
Money => GPU => Token => Money
GPUs are all you need!

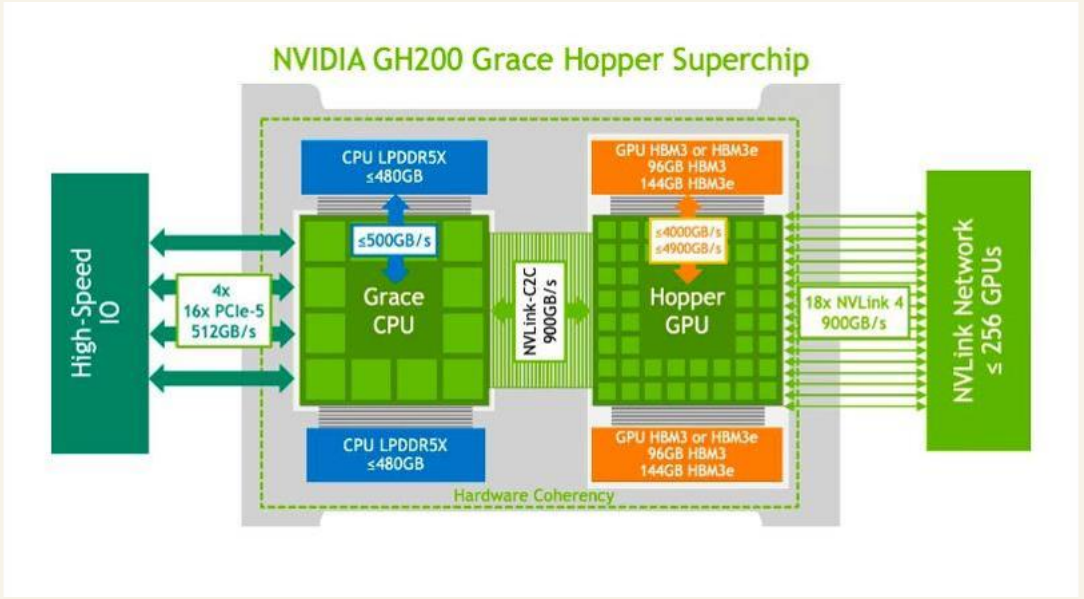


How Systems Researchers See NVIDIA GPU



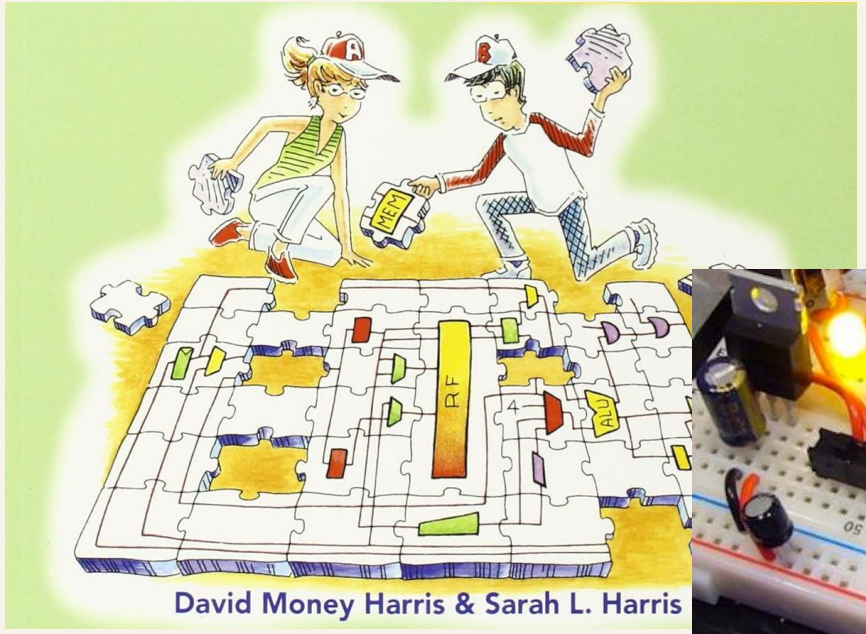


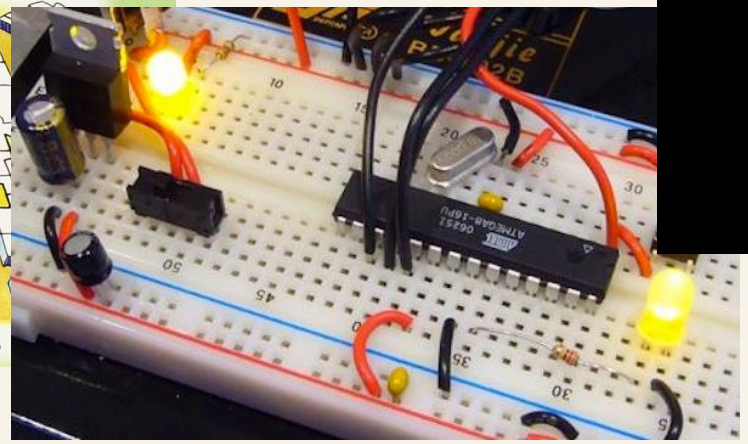


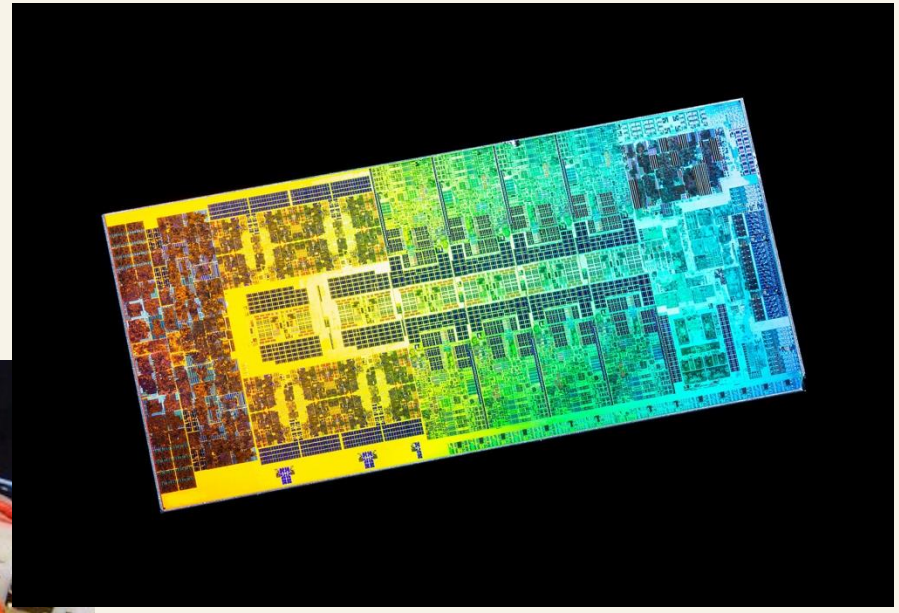


A Breadboard and an NVIDIA B200: Same Ingredients

- 画一堆小方块，用线连起来；写些字涂上点颜色；最后一烧，烧出一个亮晶晶的小方块
- 我们究竟需要知道什么？



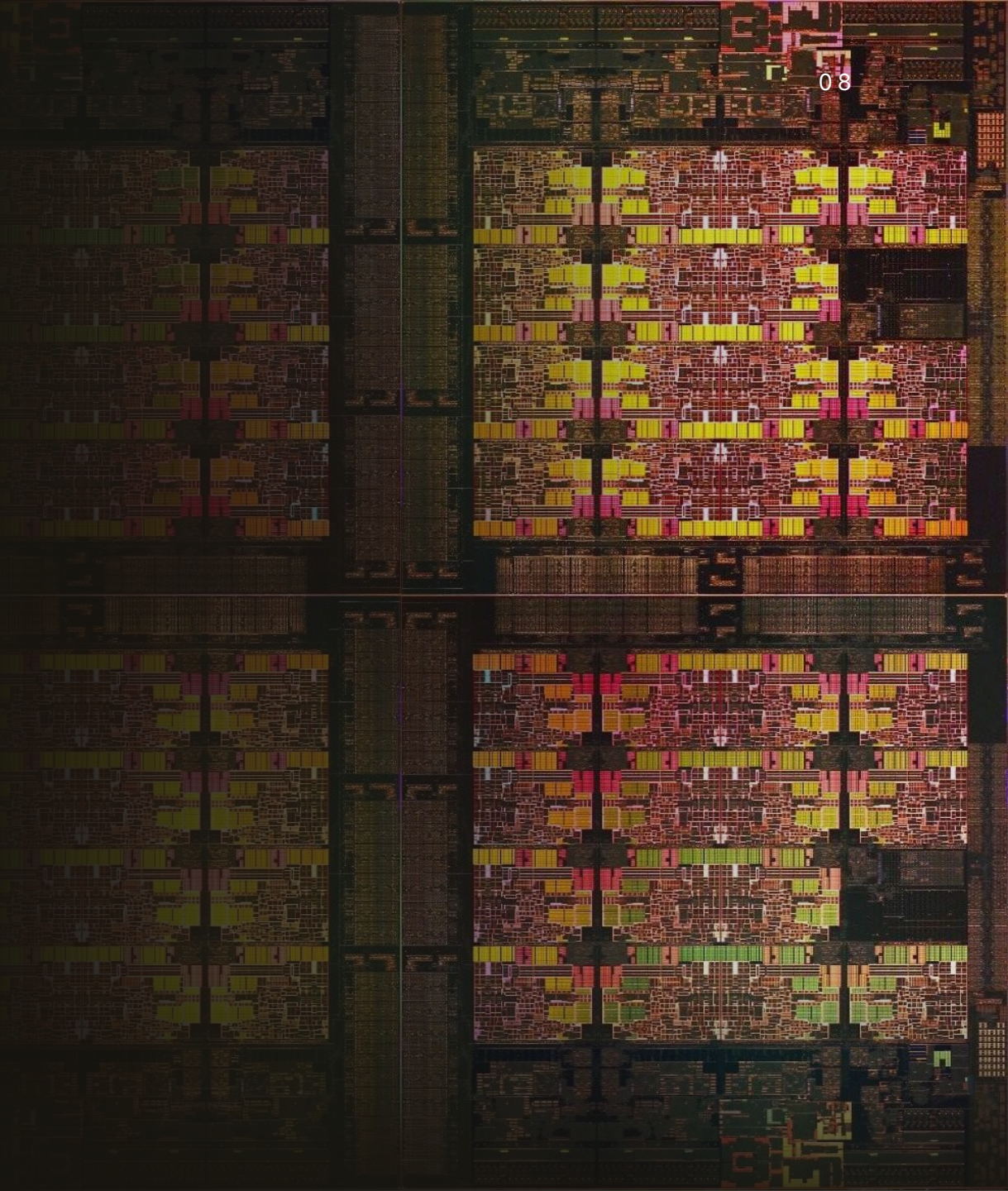




PART I

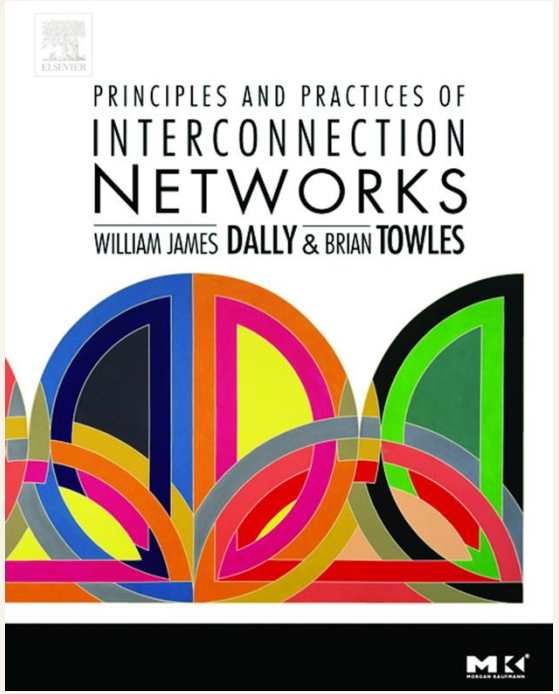
I

Behind the Monolith: From Circuits to Managed Resources



Reference

本讲义参考书目



William J. Dally · Brian Towles

Morgan Kaufmann, 2004

作者 · AUTHOR

William “Bill” Dally

NVIDIA 首席科学家

斯坦福大学前计算机科学系主任

美国工程院院士 · 美国艺术与科学院院士 · IEEE / ACM Fellow；Eckert-Mauchly、Seymour Cray、Maurice Wilkes 奖得主；论文 250 余篇、专利 120 余项；Velio、Stream Processors 联合创始人。Caltech 计算机科学博士。

关于 · ABOUT

本讲义的概念骨架——握手、缓冲、**credit 流控**、**虫洞路由**与**虚拟通道**复制了源自 Dally 的学术脉络：他在 Caltech 参与设计的 Torus Routing Chip 推动了 Wormhole Routing 与 Virtual-Channel Flow Control；MIT 的 J-Machine 与 M-Machine 探索了低开销通信与「硬件机制 / 编程模型分离」；斯坦福时期的信号、路由与同步研究成为大规模并行机的公共基础。

这本 2004 年的教材是上述工作的系统化总结，至今仍是互连网络的标准参考。

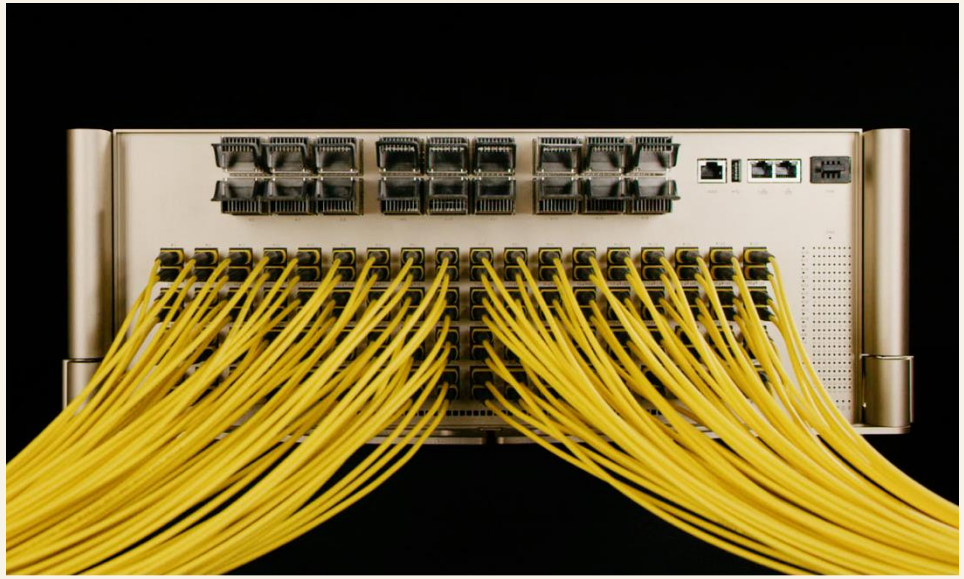
资料来源：NVIDIA Research 个人主页

1.1 From Circuit to Link

从信号、握手、缓冲与流控构建可靠链路

“Mr. Watson, come here, I want to see you.”

—— 亚历山大·格雷厄姆·贝尔



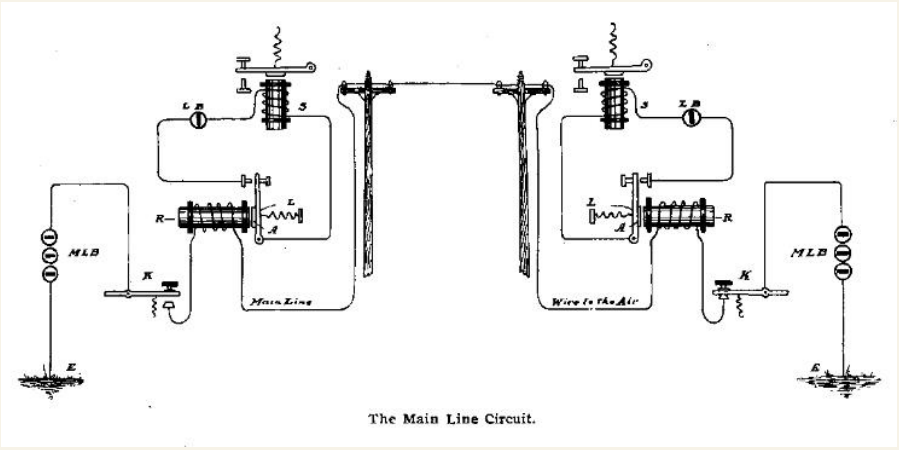
CPO Switch with 144x800 Gbps InfiniBand Port

NVIDIA Quantum-X InfiniBand Photonics Switch · NVIDIA

Wire and Communication

A ——— Y

A[7:0] ——— Y[7:0]



The Main Line Circuit.

Tickers for news and stock

1869 年起，股票报价不再靠人耳——ticker 自动把行情印上纸带。

工作原理（Western Union 3-A，Edison 系）：

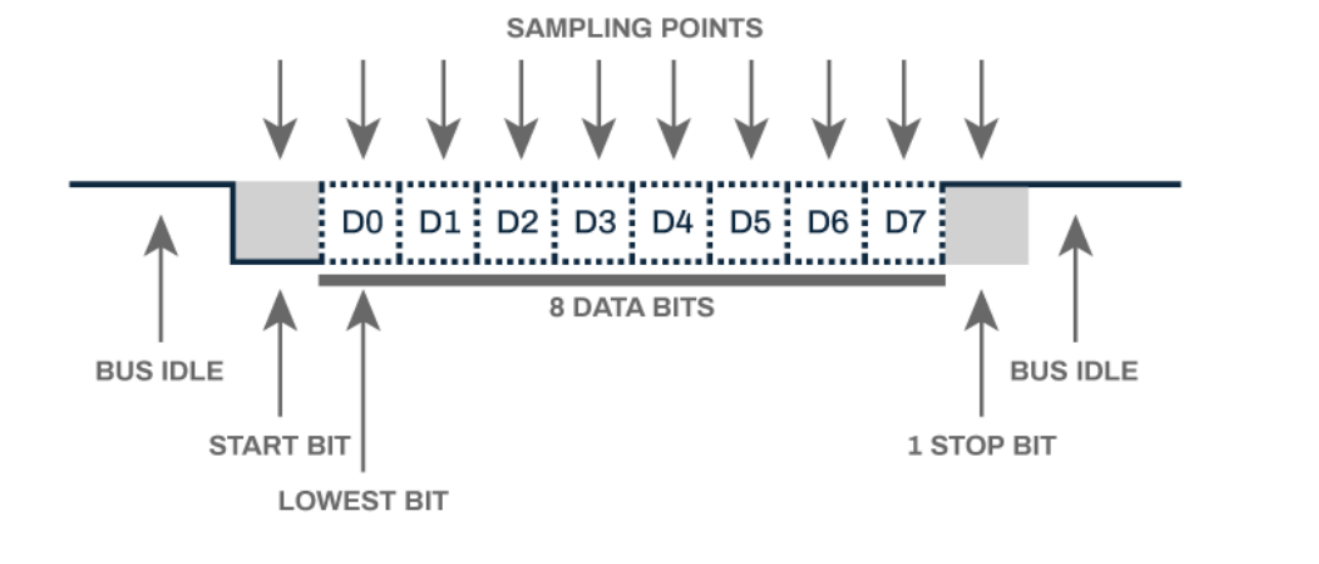
- 两条电报环路：一条传步进脉冲转动字轮，一条传打印脉冲落锤；
- 同一环路上所有收报机跟随发送端的定时轮——单向广播，没有反向信道；
- 字符码复用：LTRS/FIGS 换档，同一组码点既表示字母又表示数字（Baudot/Murray 码的先声）。

齐奏机制（unison）——失同步后的集体重新对表：

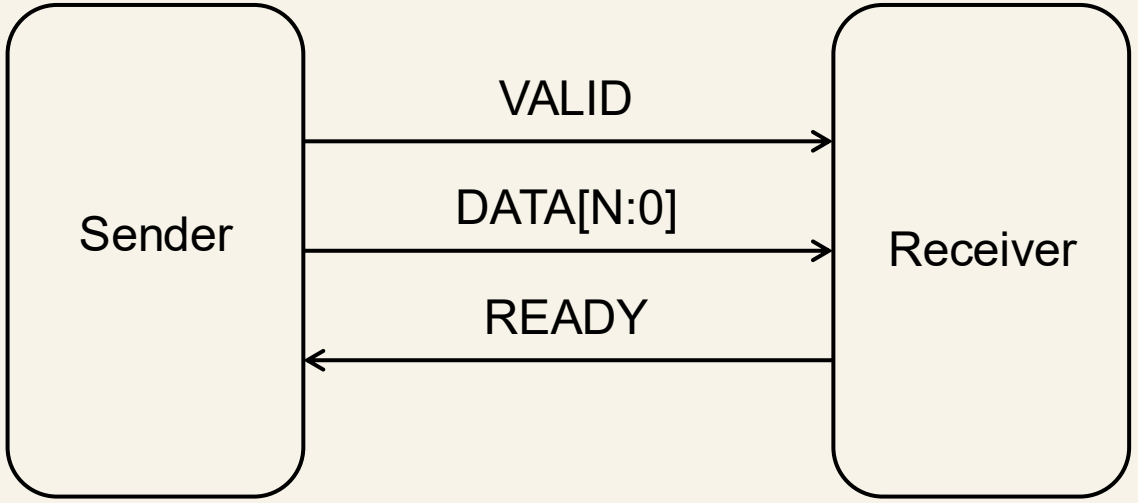
✓ 某台机器字轮跑偏了怎么办？发送端停发打印脉冲、只发步进脉冲，让所有机器转回起始位置；再发一个打印脉冲，全环恢复同步、自动回到字母档。

TeleTYpe

Case Study: UART



Flow Control



- 数据不总有效 → VALID：发送端声明"现在 DATA 上是真数据"
- 接收端不总就绪 → READY：接收端声明"我现在能接"

三条信号各自回答一个问题：DATA 回答"是什么"，VALID 回答"是不是真的"，READY 回答"接不接得住"。一次交接只在 VALID && READY 时发生。

Flow Control

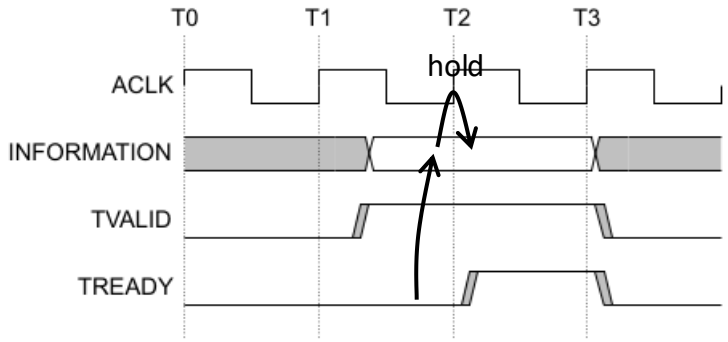


Figure 2-1 Handshake with TVALID asserted before TREADY

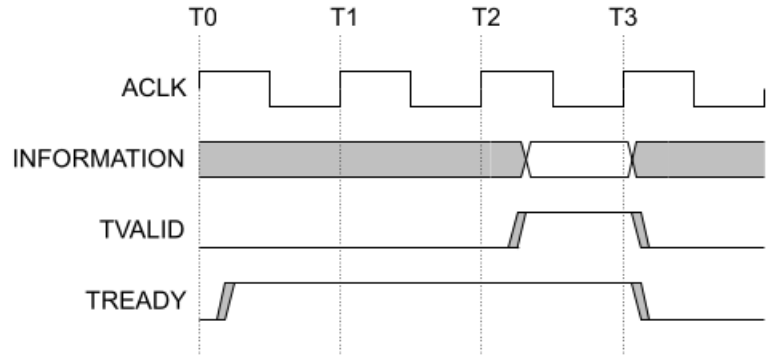


Figure 2-2 Handshake with TREADY asserted before TVALID

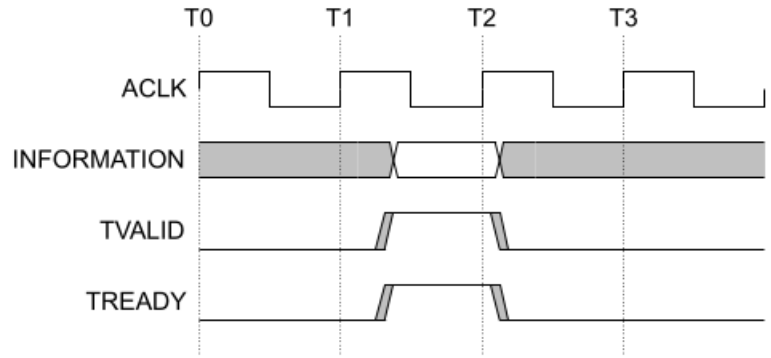
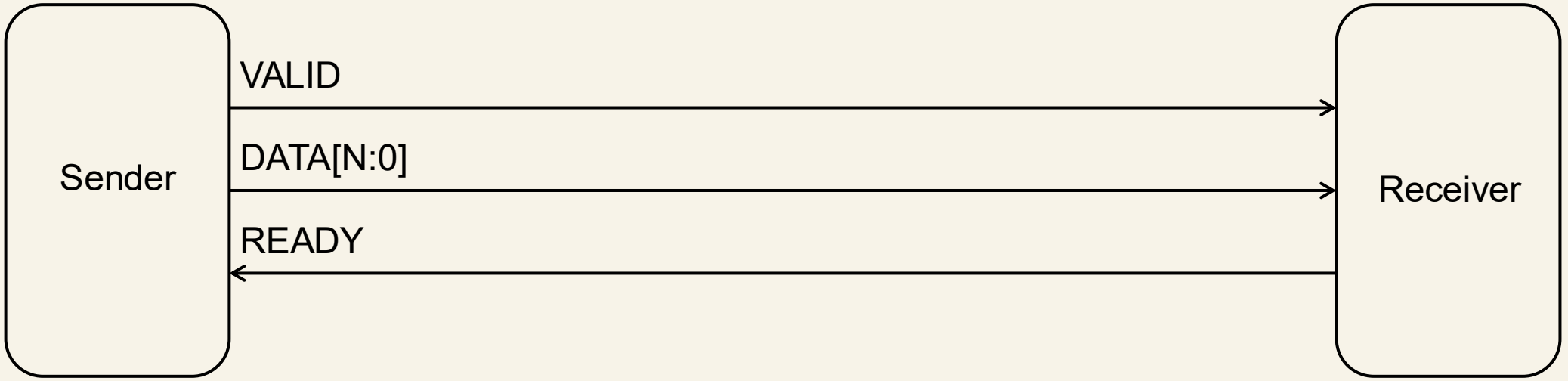


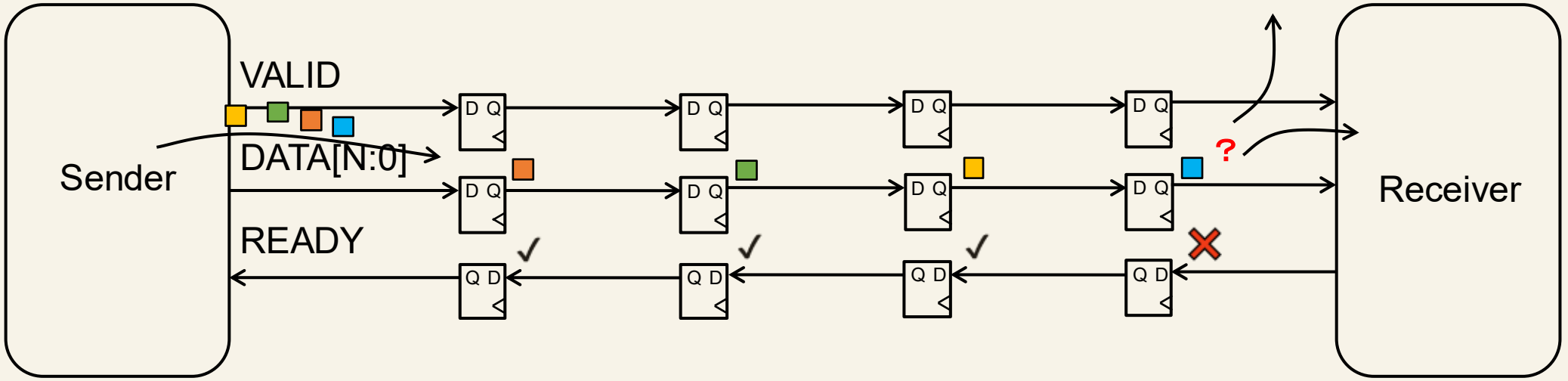
Figure 2-3 Handshake with TVALID and TREADY asserted simultaneously

Long Wires



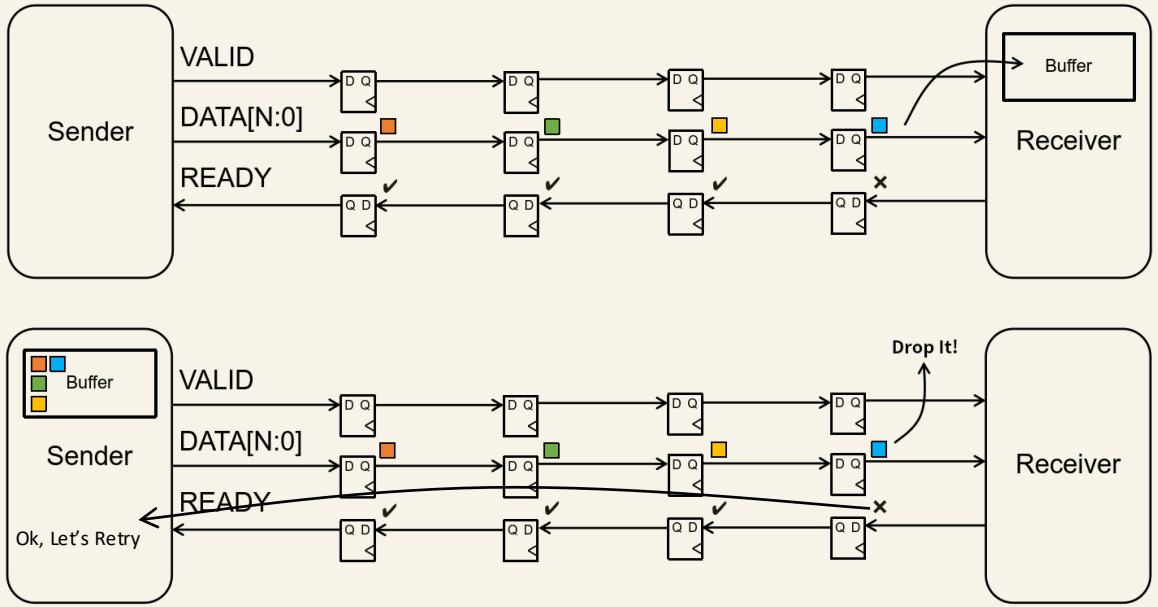
- 线路变长之后，ready 的回答要 N 拍才能传回来。

Long Wires



- 线路变长之后，ready 的回答要 N 拍才能传回来。
- 在飞的 N 个数据没有刹车。

Long Wires

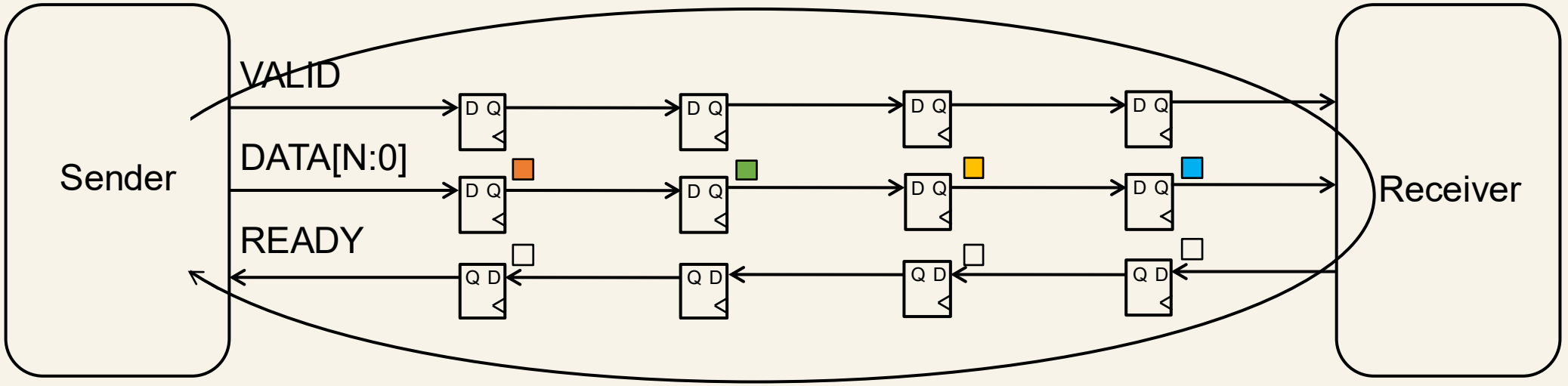


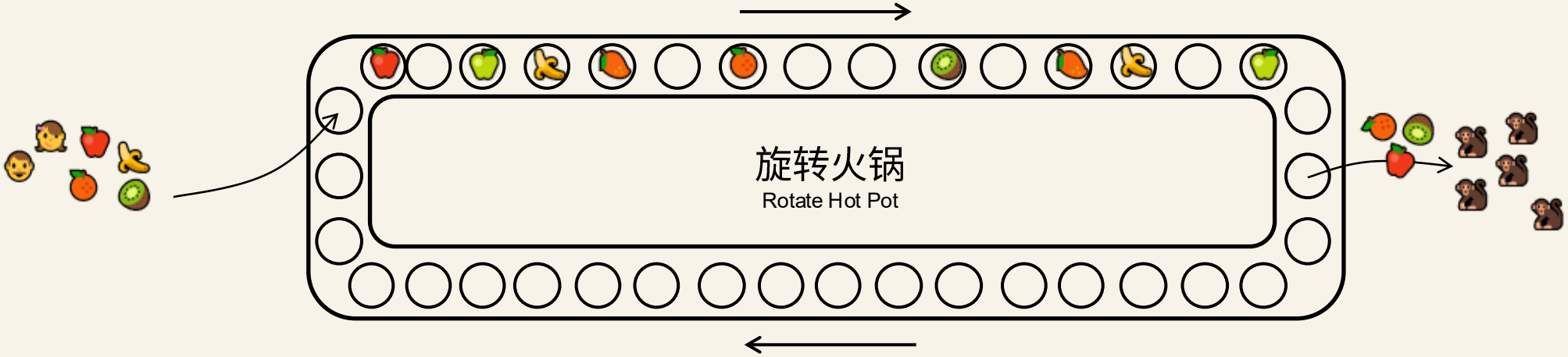
- 线路变长之后，ready 的回答要 N 拍才能传回来。
- 在飞的 N 个数据没有刹车。
- 每个"能不能做"的问题，只有两种回答方式：

✓ **先验** —— 先拿许可再发（ready / credit）：许可会迟到，接收端必须备足 RX FIFO 接住窗口内全部在飞数据。这种网络被称为**无损网络(Lossless)**。

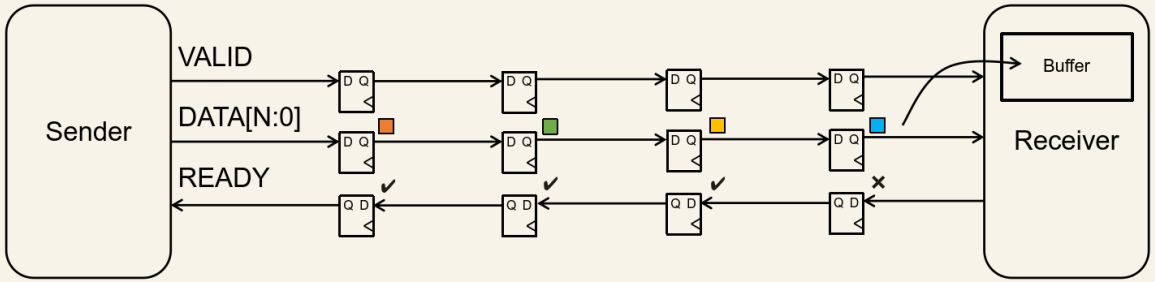
✓ **后验** —— 先发后判（ACK NAK / Retry）：不等许可直接发，丢了错了重来。发送端必须备足 TX Window，把未确认的数据存到 ACK 回来。这被称为**有损网络(Lossy)**。

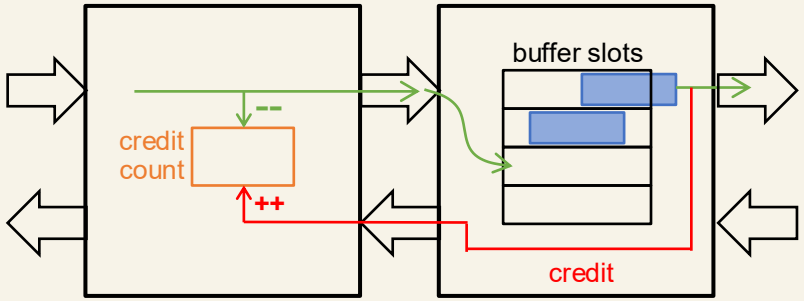
Credit

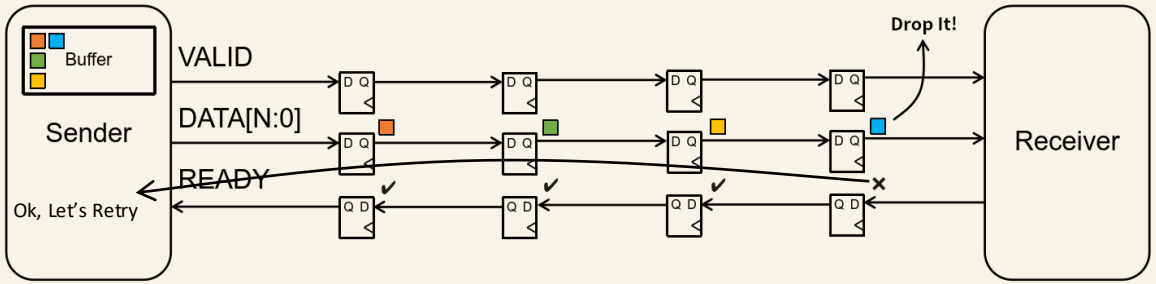

Sender
VALID
DATA[N:0]
READY
D Q
Q D
Receiver

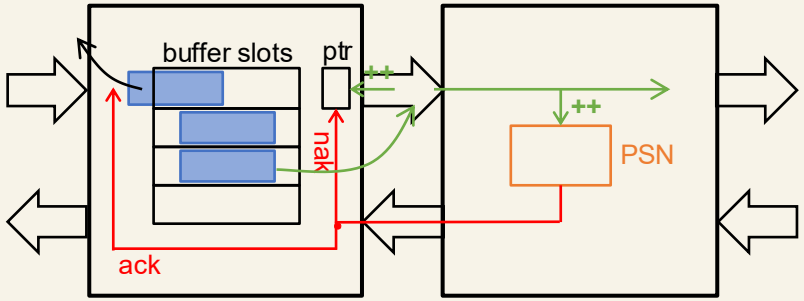

旋转火锅
Rotate Hot Pot

Credit & Replay Window

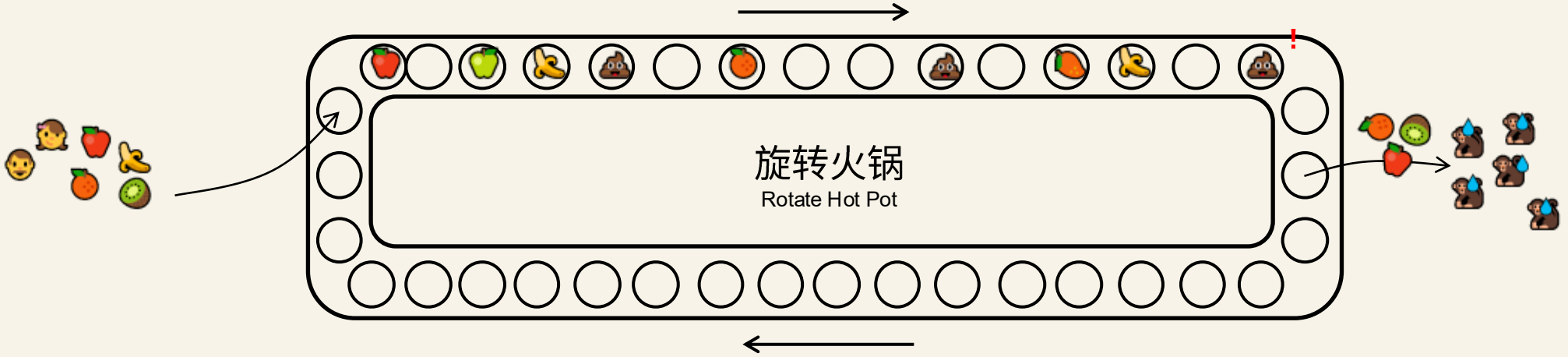








Failure Everywhere in the world



- ✓ 无损网络(Lossless)和有损网络(Lossy)的区别在于，**面对潜在发生的溢出，采用何种策略**(先验或后验)。
- ✓ 由于现实世界是充满**故障**(Failure)，接收到的数据可能发生**腐烂**(Rot)，需要**注意**其与无损/有损概念的区别。
- ✓ 解决办法有两种：一种是利用**冗余**(Redundant)实施修复，一种办法是**丢弃**并重传(Retry)。

两个维度的排列组合：可靠无损网络，不可靠无损网络，可靠有损网络，不可靠有损网络

Case Study: PCIe Link Layer

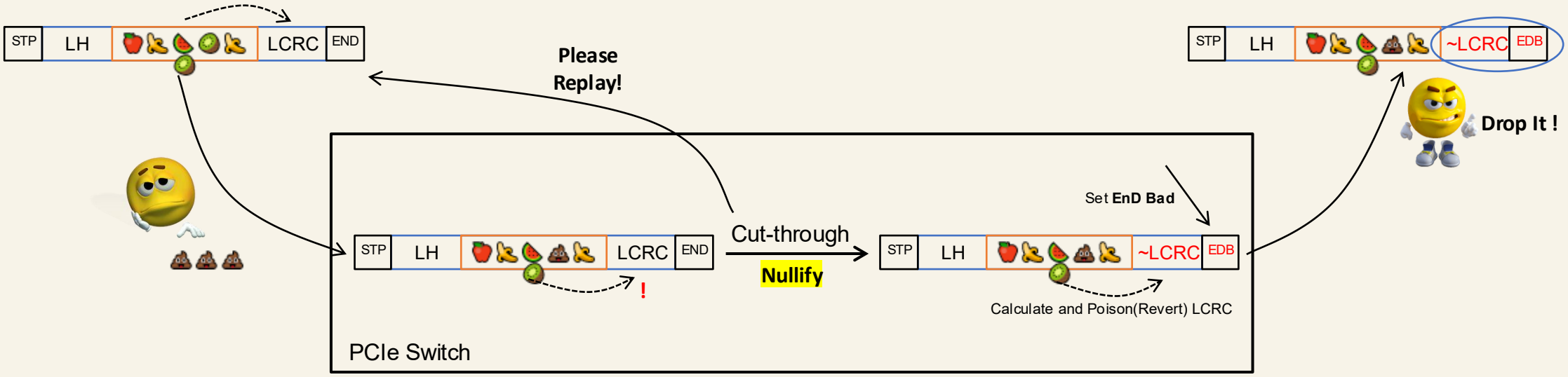
Flow Control:

- 三类流量（Posted / Non-Posted / Completion）的头与数据均独立记账；逐 VC 记账，VC0 强制。
- 信用面额：数据 16 B/个，头 1 个/个；
- 开局握手：InitFC1 申报（我方缓冲几何）→ InitFC2 确认（你方账本已抄收）；账本未收齐之前，链路上不存在数据。
- 运行期：UpdateFC DLLP 随时还盘，发送端无 credit 不发。
- 最低配申报 = 恰好接住一个最大包（0x40 = 1024 B ÷ 16 B）；性能配置必须盖住信用往返延迟，否则流水线断粮。

Retry:

- 挂账与退休：TLP 前加 12-bit 序号、后加 32-bit LCRC，副本存 Retry Buffer；
- ACK 推进 ACKD_SEQ 退休，NAK（CRC 错误）或超时 → 按原序重放（**Go-Back-N**）。
- 窗口纪律：(NEXT_TRANSMIT_SEQ − ACKD_SEQ) mod 4096 ≥ 2048 即停收新 TLP——序号空间必须 ≥ 2× 窗口，否则回绕后新旧不分。
- 超时纪律：REPLAY_TIMER 只在有进展时重置——超时器度量的是进展的缺席，不是时间的流逝。

Case Study: PCIe Link Layer



PCIe 是一种采用 **逐跳 Credit 无损流控和逐跳 Replay 可靠传输(Go-Back-N)** 的树状互连。

在 Non-FLIT 模式下，Switch 可在入口 TLP 尚未通过 LCRC 校验时进行 Cut-Through 转发。若到达包尾才发现入口 LCRC 错误，已经发往出口的字节无法收回。

Nullify：Switch 使用反相 LCRC 和 EDB 结束已开始发送的出口 TLP。下游将其识别为一次可靠送达的撤销，静默丢弃，不推进出口链路序号，也不触发下游 Replay；原始错误则由入口链路通过 NAK/Replay 恢复。后续链路上的包将被丢弃，直到重传恢复。

Appendix: Slicing and Dicing

切片（slicing）：一条宽通道切成几条窄的 → 端口多、并发多、单条慢

切块（dicing）： 几条窄通道绑成一条宽的 → 端口少、并发少、单条快

这块板上最贵的不动产不是硅，是 CPU 边缘的引脚。每一排 DIMM 槽和高速互联都是预算的一份。

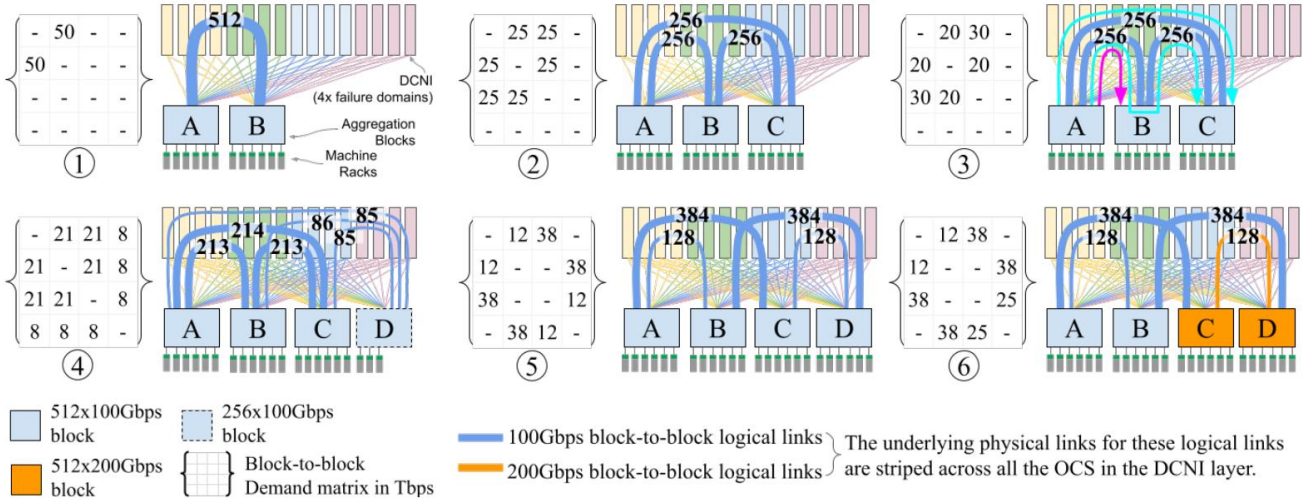
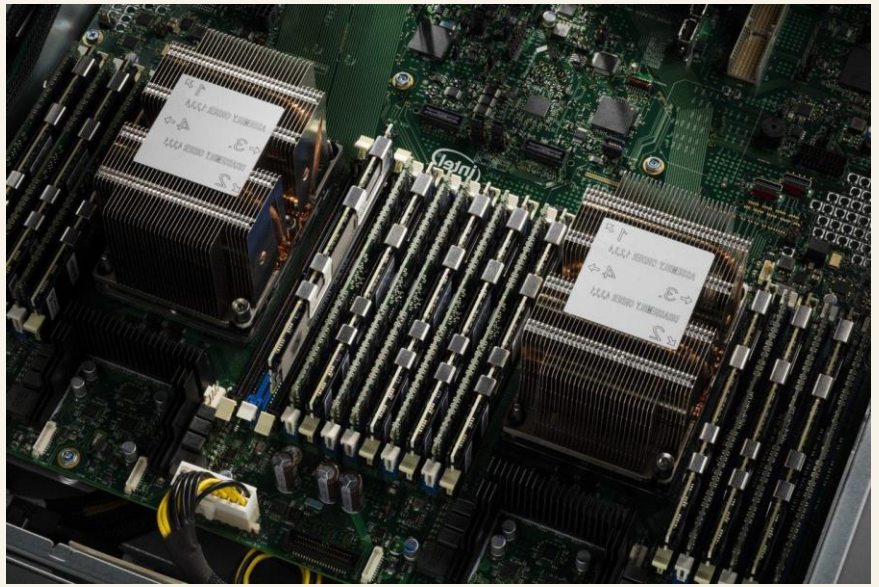


Figure 5: An incrementally deployable Jupiter fabric with a direct-connect topology enhanced by dynamic traffic and topology engineering. ①: Initially, Aggregation Blocks A, B are added with 512 uplinks each. ②: Block C is added with 512 uplinks. Each block has 50T outgoing demand, uniformly distributed across other blocks. Topology engineering forms a uniform mesh topology to match the uniform steady-state demand matrix. ③: Traffic Engineering (TE) adjusts Weighted Cost Multi Path (WCMP) weights based on finer-grained version of demand in ②: A sends all traffic (20T) to B directly (pink) and splits traffic to C (cyan) 5:1 (25T:5T) between direct and indirect paths (via B). ④: Block D is added with 256 uplinks (only a subset of machine racks in D are populated). ⑤: Block D is augmented to 512 uplinks. ⑥: Blocks C, D are refreshed to 200G link speed.

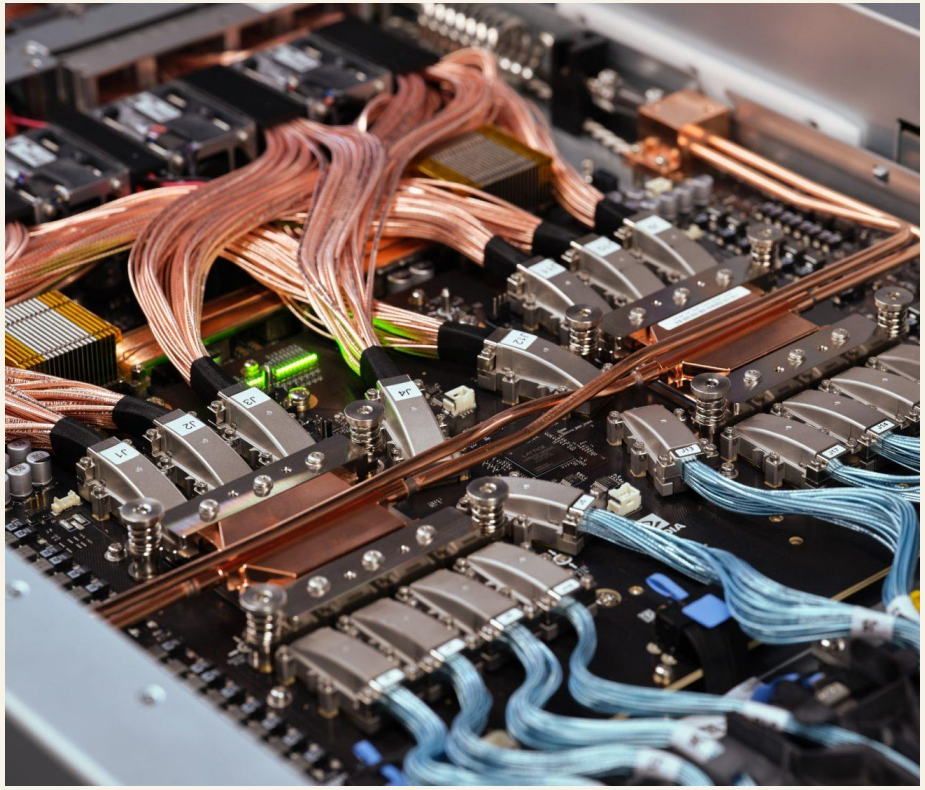


1.2 From Link to Network

交换、仲裁与虚通道

“UNIX is basically a simple operating system, but you have to be a genius to understand the simplicity.”

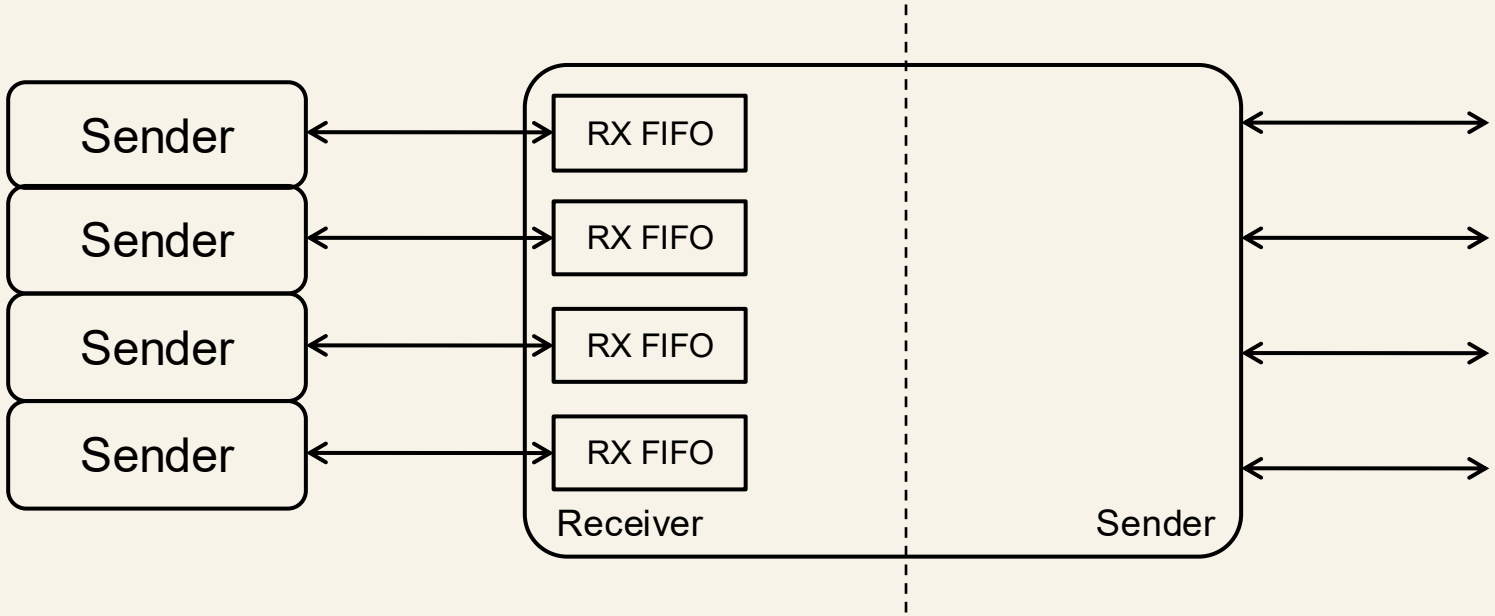
—— 丹尼斯·里奇



交换机托盘内部结构

Blackwell NVLINK Switch Tray · NVIDIA

Router

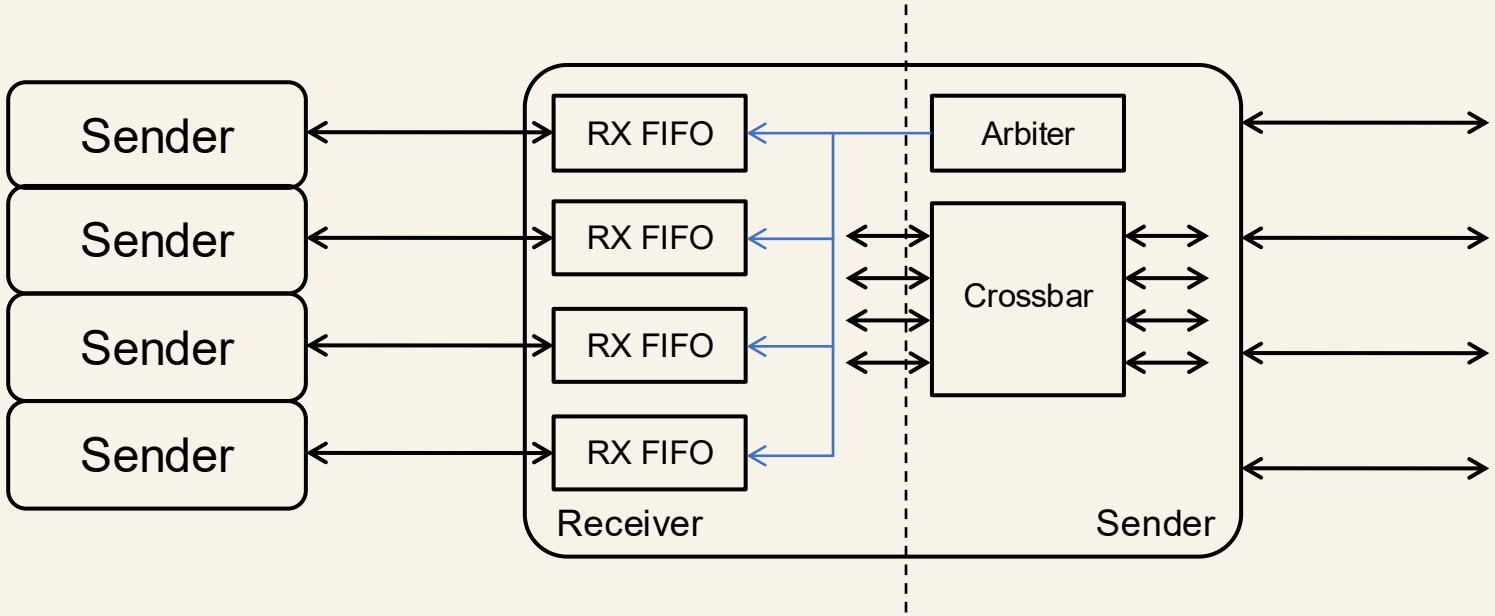


上一章我们推导了结构，这里复制复制 N 份的端到端 = 一台路由器

每条入线 = Receiver + RX FIFO（这条链路 credit 的记账端）

- 每条出线 = Sender（下一跳的 FIFO 就是它的接收端）

Router



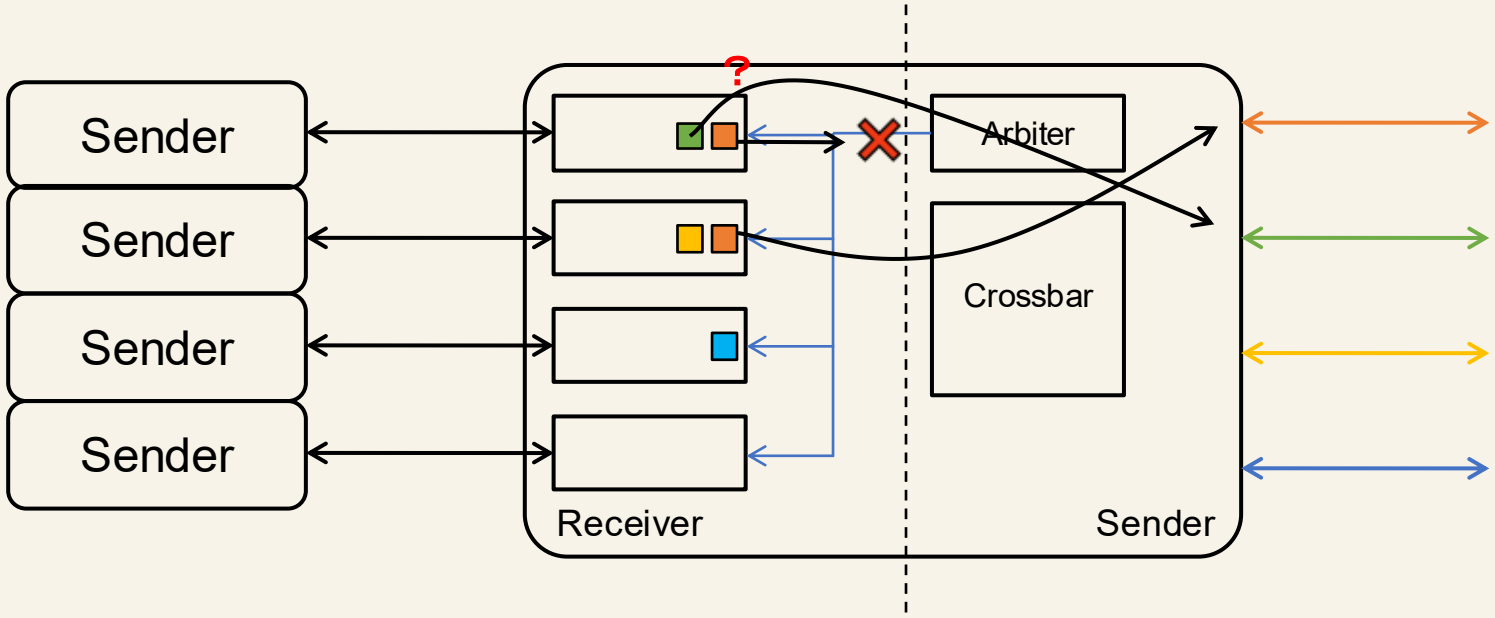
一台路由器 = 复制 N 份的链路端点

- 每条入线 = Receiver + RX FIFO（这条链路 credit 的记账端）
- 每条出线 = Sender（下一跳的 FIFO 就是它的接收端）

数据从入线到出线：仲裁 与 Crossbar

公平性：谁优先？会不会有人永远轮不到？

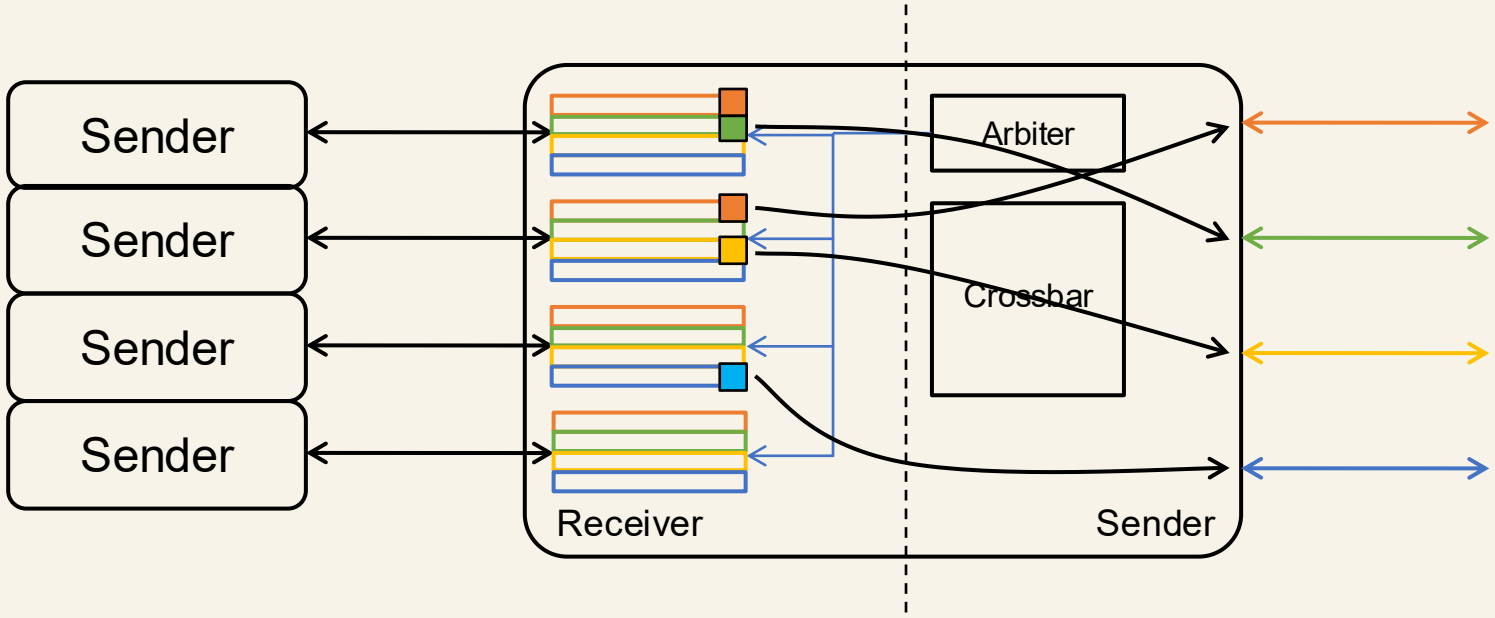
Head-of-Line Blocking



- FIFO 先进先出：队首动不了，身后能走的也全被堵死；
- 队首被阻塞，后续包的目标空闲：队头阻塞（Head-of-Line Blocking）

随机流量下，输入队列交换机的吞吐上限只有 58.6%（Karol, 1987）——四成交换能力被队头阻塞吃掉。

Virtual Channel



把一个输入口拆成几条并行队列，这里我们安排**不同去向分开排队**——VC0 堵着，VC1 照走。

- "虚"在共享：物理入线仍只有一条，是两份小缓冲 + 仲裁把它复用成互不阻塞的通道，每个 VC 一本自己的 credit 账，相互不影响
- 副产物比主产物更重要：流量类别隔离——我们也可以安排**请求与响应走不同 VC**，永不互等，避免协议级死锁

Full Router Diagram

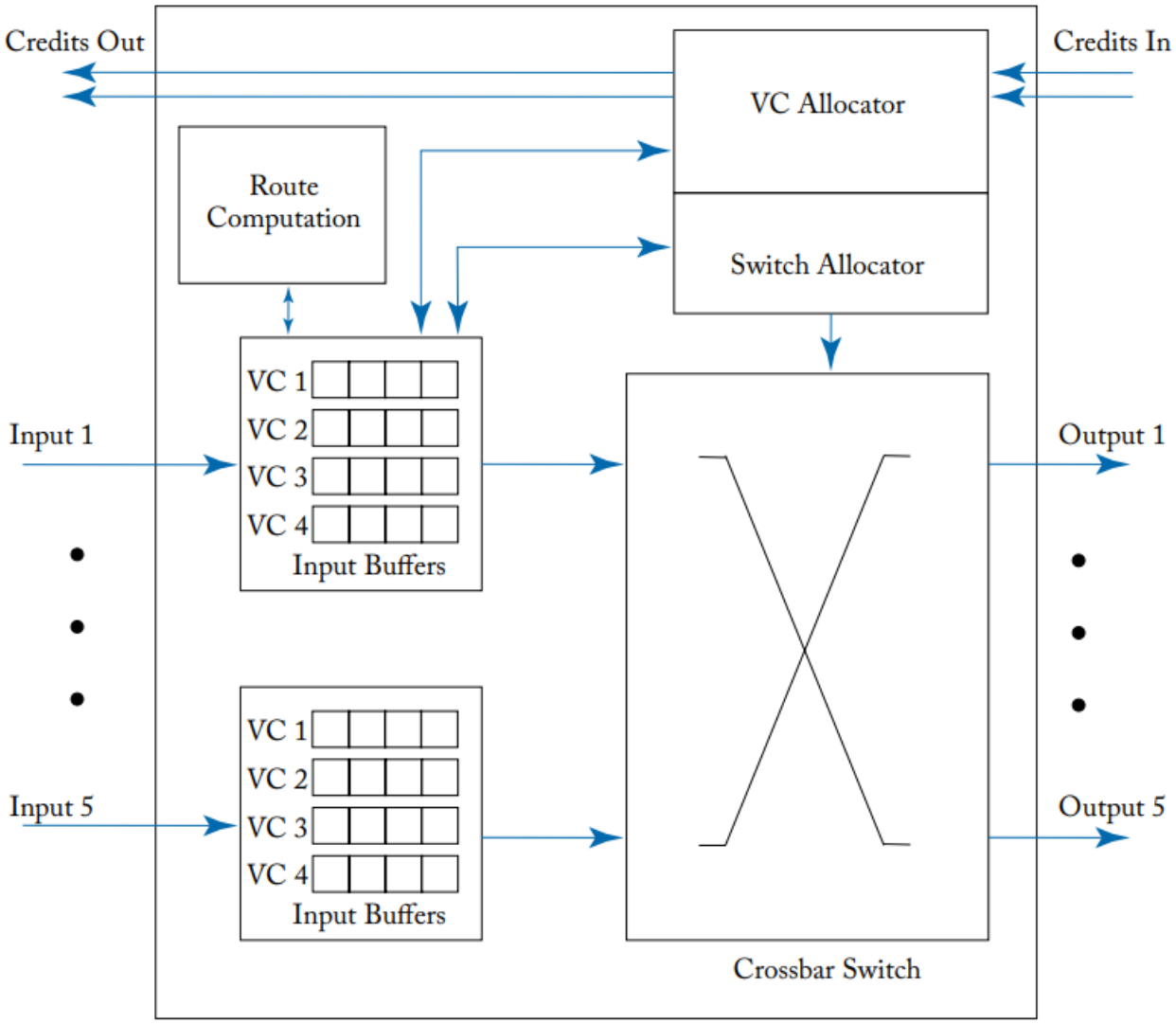


Figure 6.1: A credit-based virtual channel router microarchitecture.

Case Study: Intel TeraFLOPS

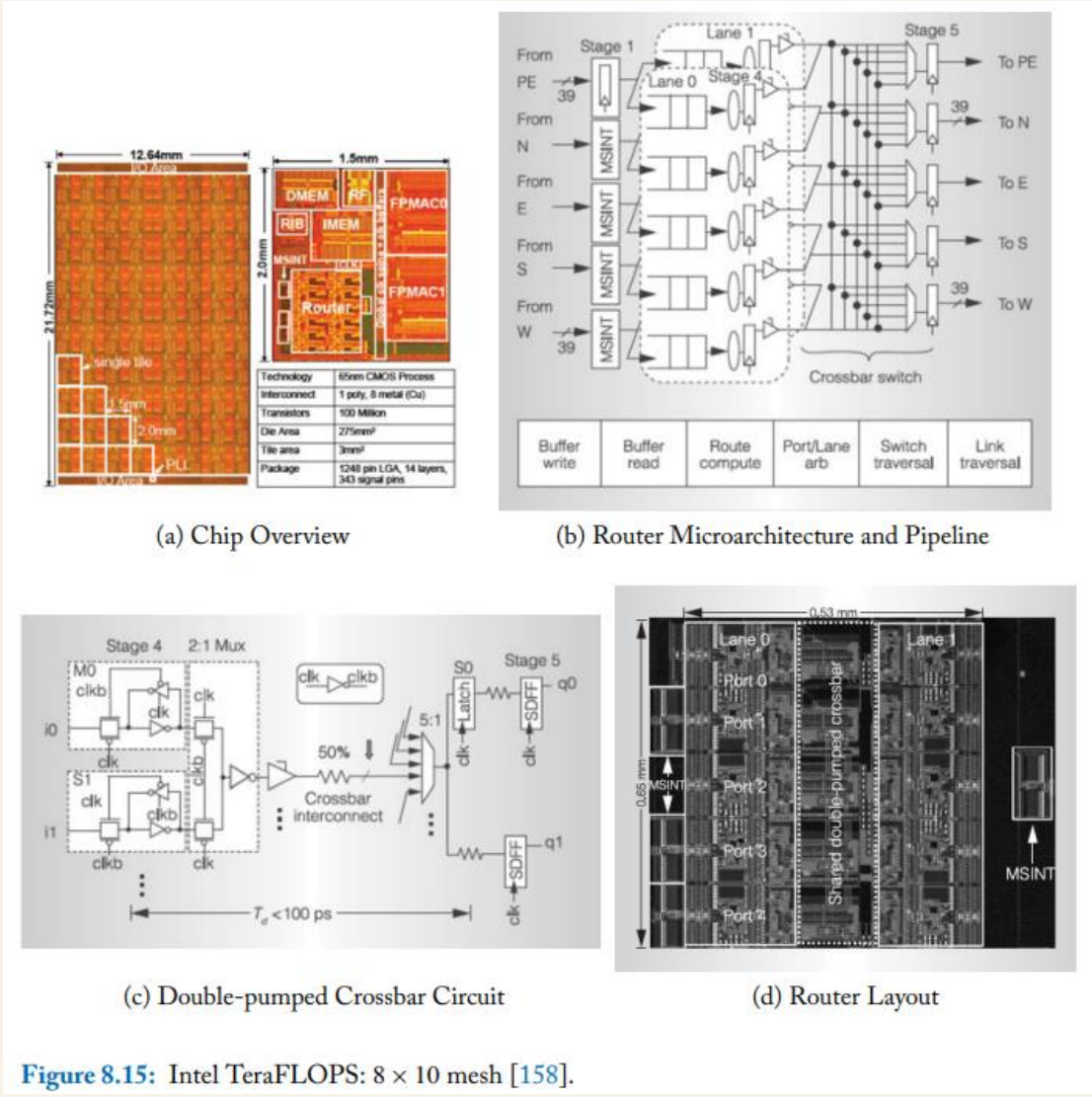


Figure 8.15: Intel TeraFLOPS: 8 × 10 mesh [158].

5 Stage Pipeline DAMQ Router

我们用一个共享队列代替每个 VC 独占一个队列。

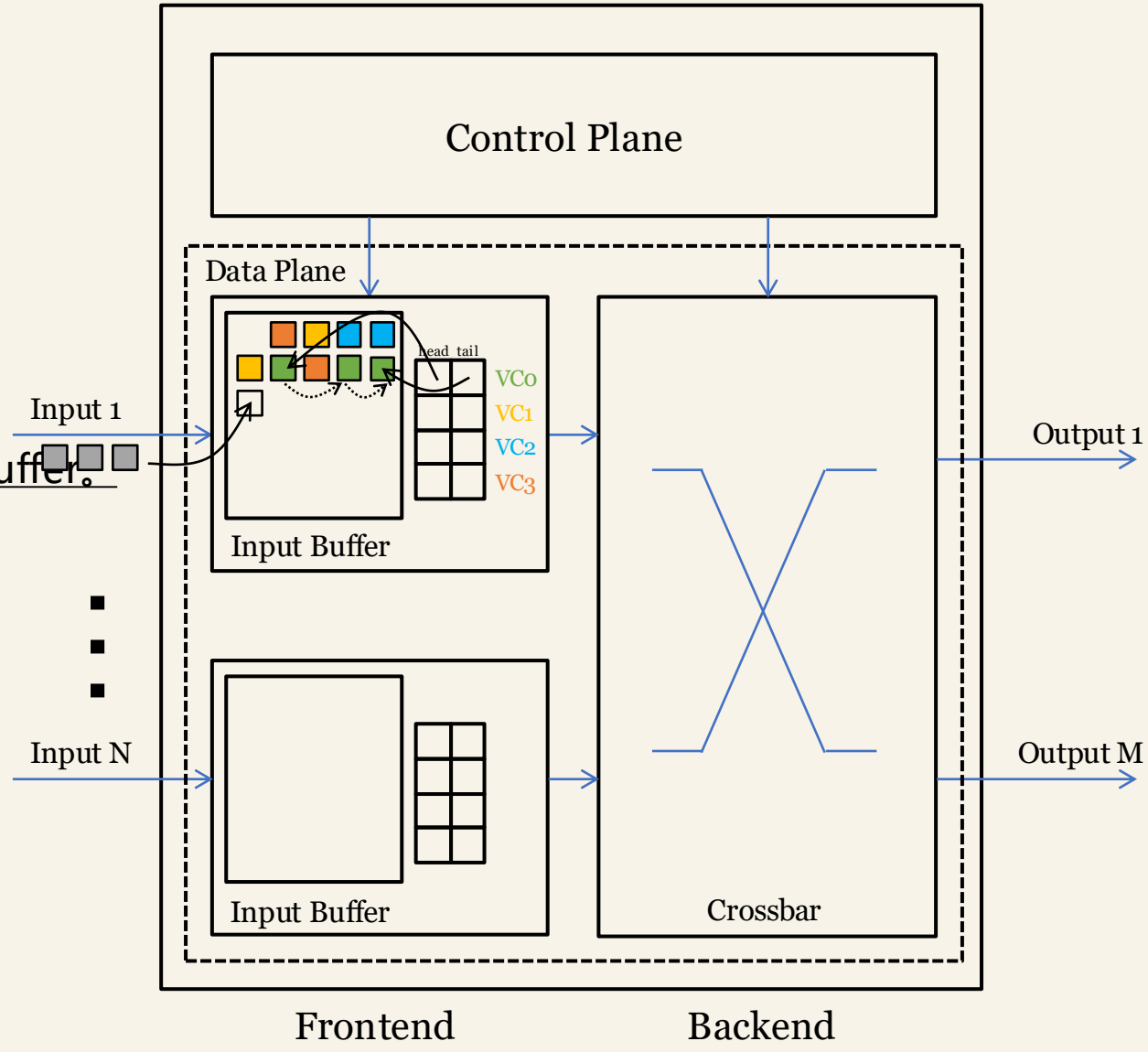
① BW: 将包写入 **缓冲结构**

② RC: 计算或获取 **目标 Output 端口**

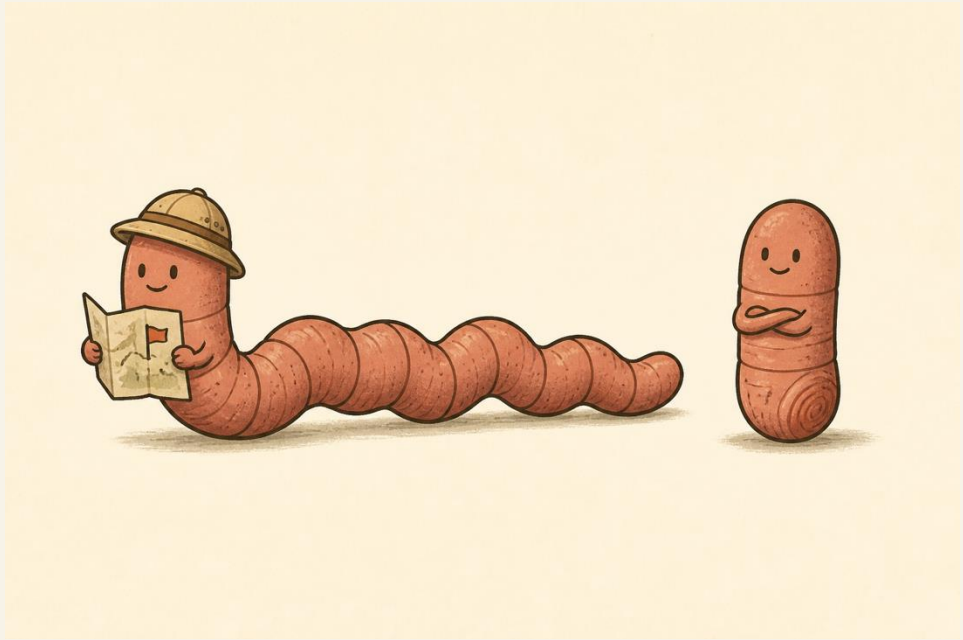
③ VA: 计算或获取 **VC 信息**(染色)，染色后挂链：旧尾槽的 next 和 VC 的 tail 指向新槽。数据入Buffer。

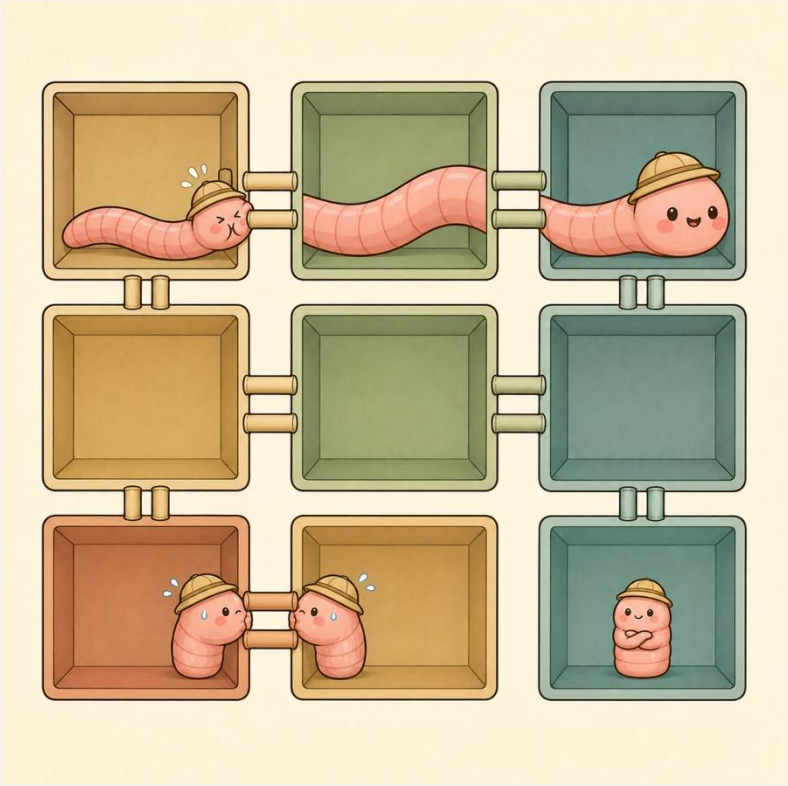
④ SA: 后端以 **所有 VC 的 head** 为候选，仲裁器为每个 Output 端口 尝试选一个(取出并更新 VC head)

⑤ ST: 交叉开关通过**多级复用器**，将输入的包送到对应的端口

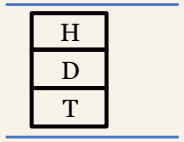


Packet and FLIT

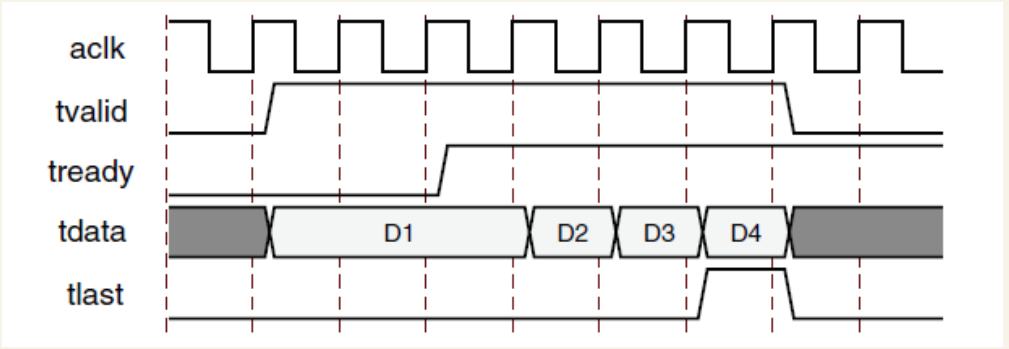




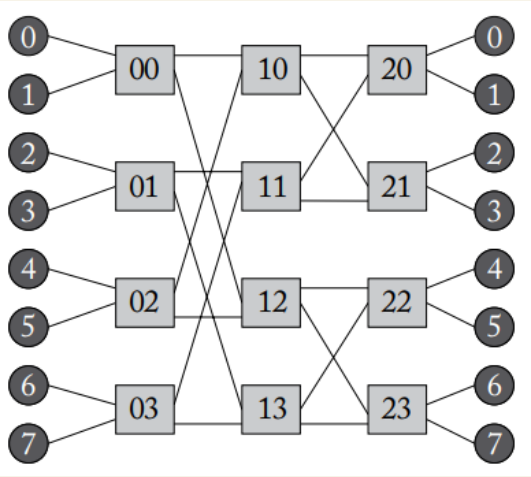
Packet

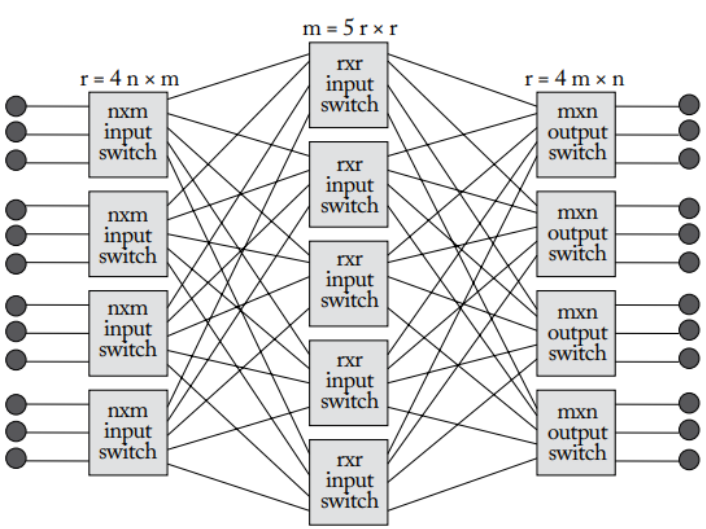


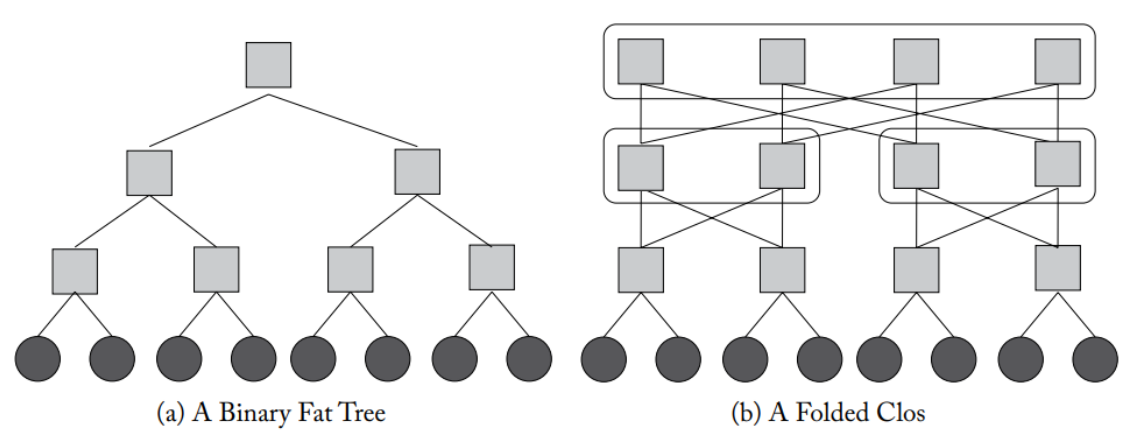
FLIT



Build Fabric with Router and Wire

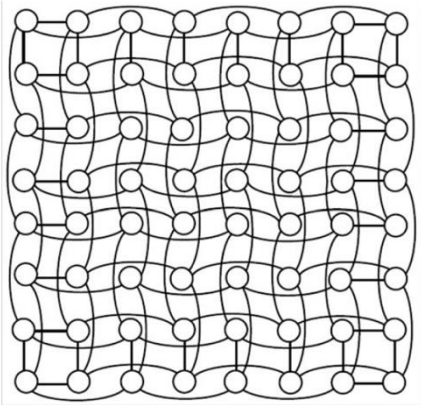


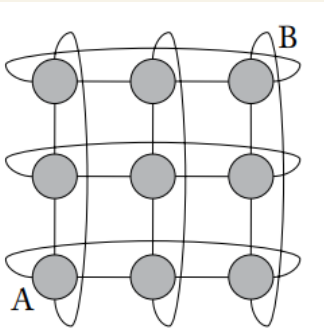


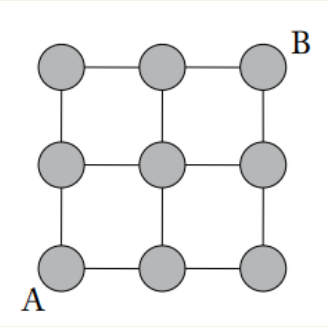


Indirect Network

Direct Network







Causation

- 硬件对事件顺序的担保只有**因果律**（happens-before）：果，一定发生在因的之后。**延迟从不被承诺**——"何时完成"无法被验证，"何者先于何者"才可以。因此两个事件之间没有因果律，其先后是**未定义的**(undefined)。

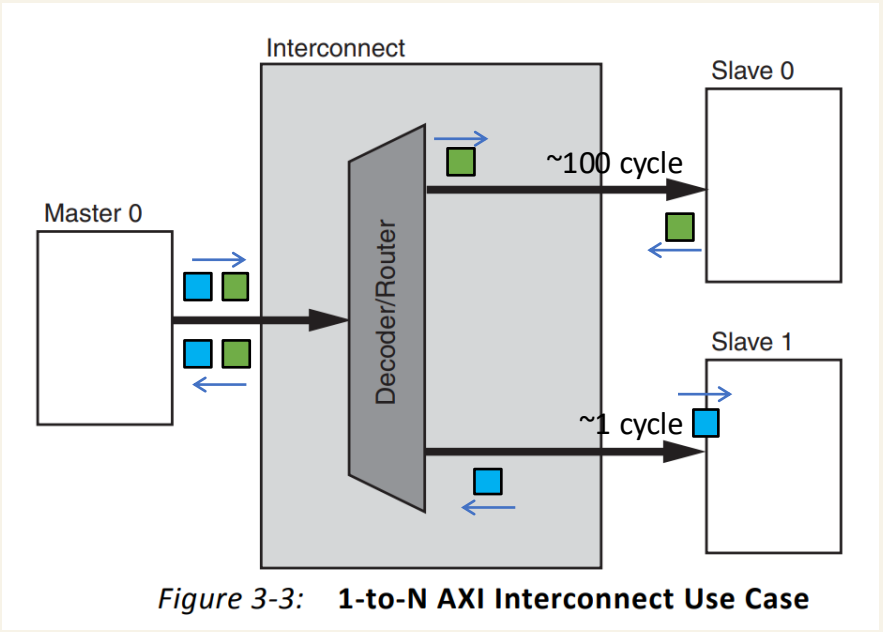


Figure 3-3: **1-to-N AXI Interconnect Use Case**

- ✓ 发往 Slave 0 与 Slave 1 的两笔事务之间没有因果链，后发送的 Slave 1 的包，也会比 Slave 0 的先被确认和返回。
- ✓ 发往 Slave 0 的两个先后事务顺序到达，顺序完成并返回：事务在单通道有时间的因果依赖，排队开火车不会乱序。
- ✓ 主机先发送 Slave 0 事务，等到返回后再发送 Slave 1：保序的代价即因果的代价，**串行化**(Serialize)消除了并行度。

Deadlock and Livelock

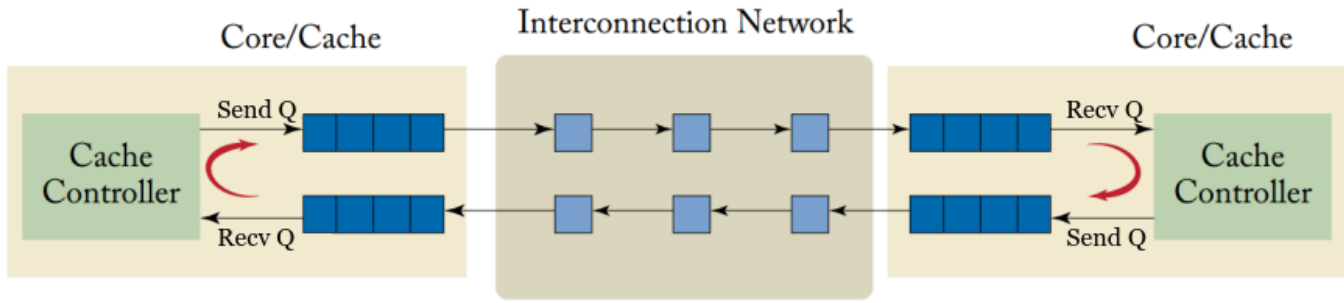


Figure 2.3: Protocol-level deadlock. Figure adapted from [294].

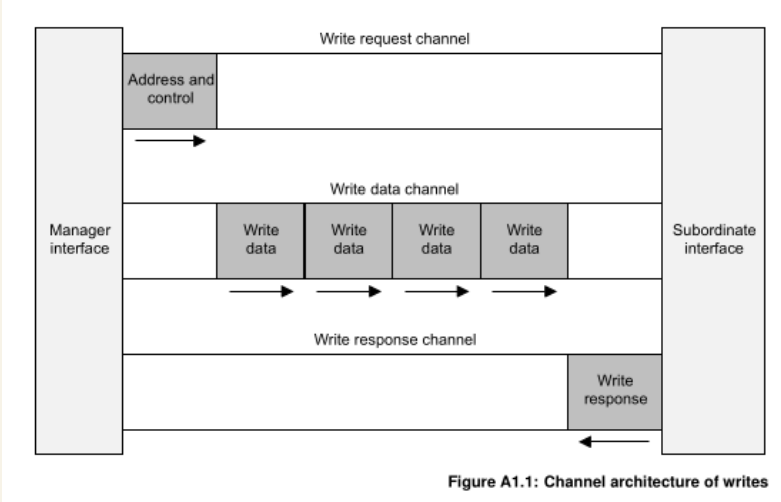


Figure A1.1: Channel architecture of writes

同一个对象的三种结构性病理：

因果成环 = 死锁(区别于**拥塞**)；因果振荡 = 活锁；因果自闭 = OrderLock。

- 死锁：如图双方同时发送包，最终网络死锁阻塞：——使用虚通道或单独通路解耦
  网络塞满待接收的包。
  设备要发响应，但是无处可放。
- 活锁：因果振荡，进展为零。

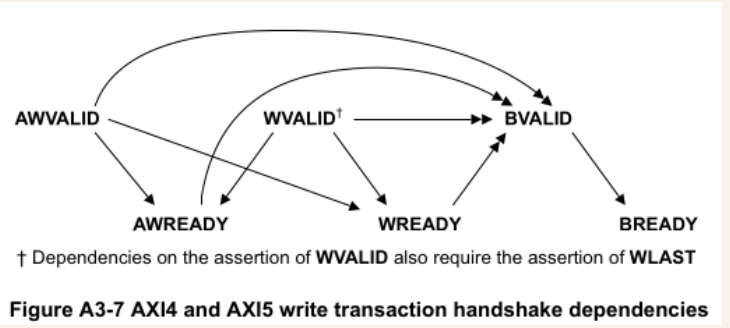


Figure A3-7 AXI4 and AXI5 write transaction handshake dependencies

单头箭头 ⟶ ：允许在对方之前断言（无强制依赖）
双头箭头 ⟶⟶ ：必须等对方完成之后才能断言（强制因果边）

SIGCOMM'25: Orderlock

𝕀 保序递交：交付给上层的顺序必须等于发送顺序。

𝕃 无损网络：每次传输在两个界面各自守恒。

𝕆 乱序持有：协议允许缺口的制造与跨缺口的持有，两半缺一不可。

——只能选两个。三者齐备 + 缓冲有界 ⇒ 死锁

- ✓ PCIe: 保序提交、无损传输、拒绝持有(Go-Back-N)
- ✓ TCP: 保序提交、有损传输、**乱序持有(Selective ACK)**
- ✓ Infiniband Adaptive Routing + Out-of-Order Placement:

  乱序提交、无损传输、乱序持有

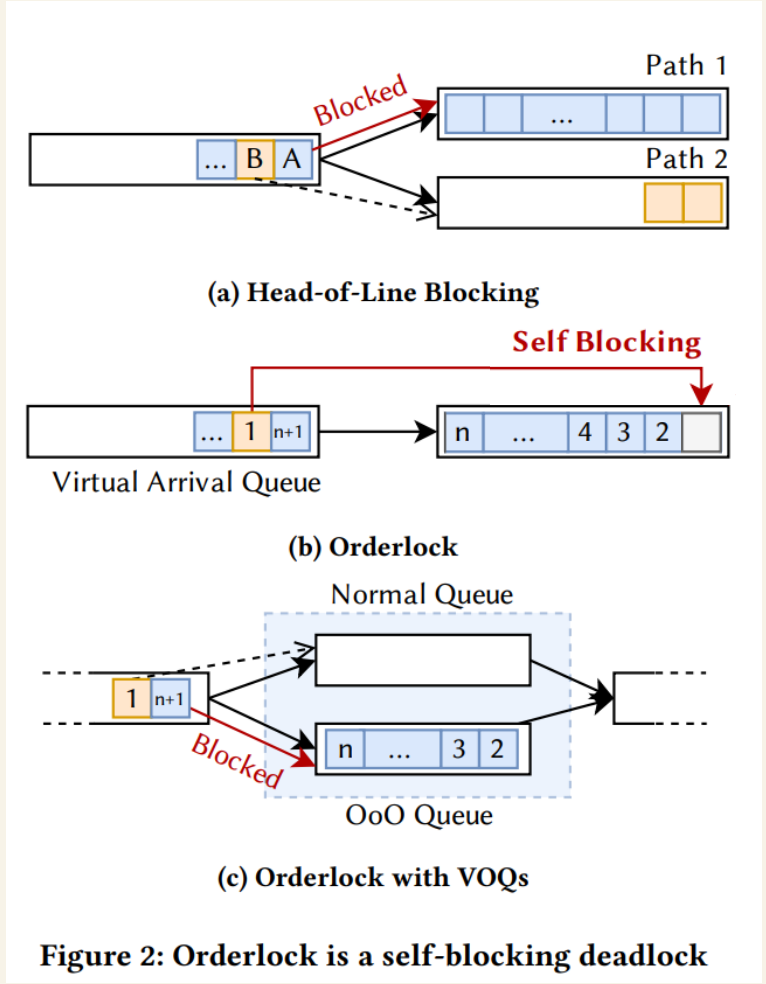
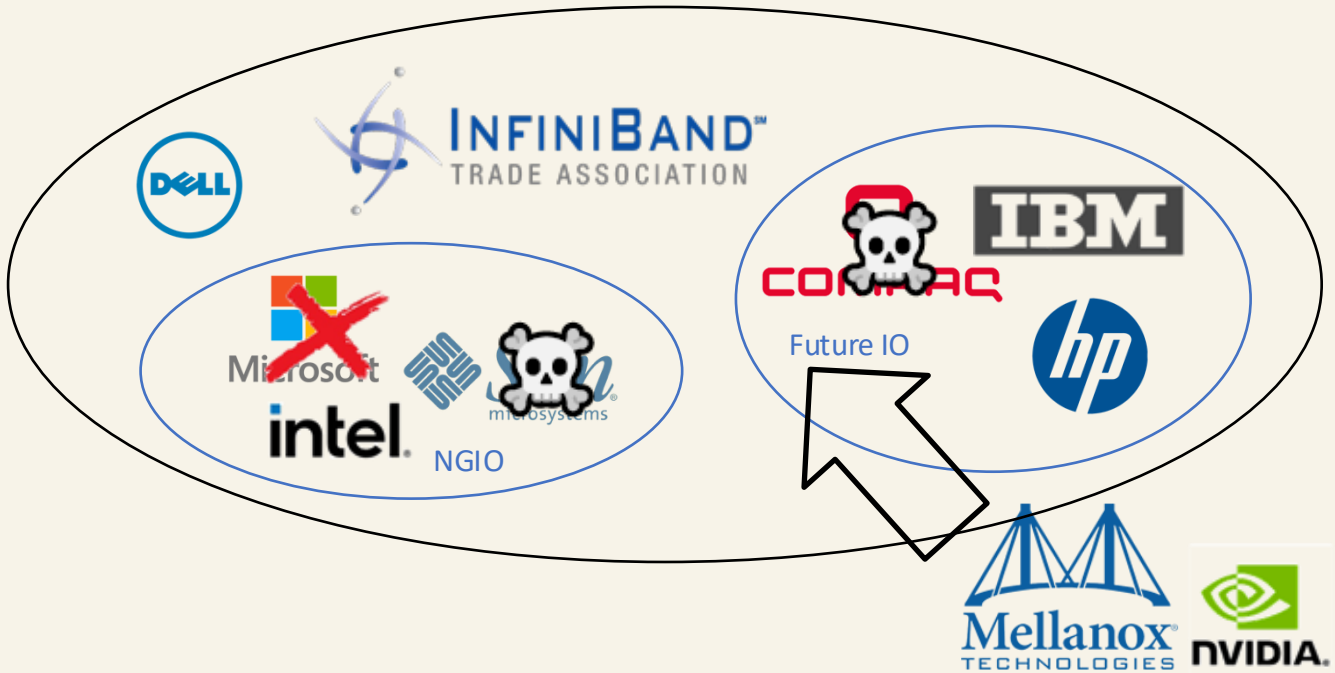
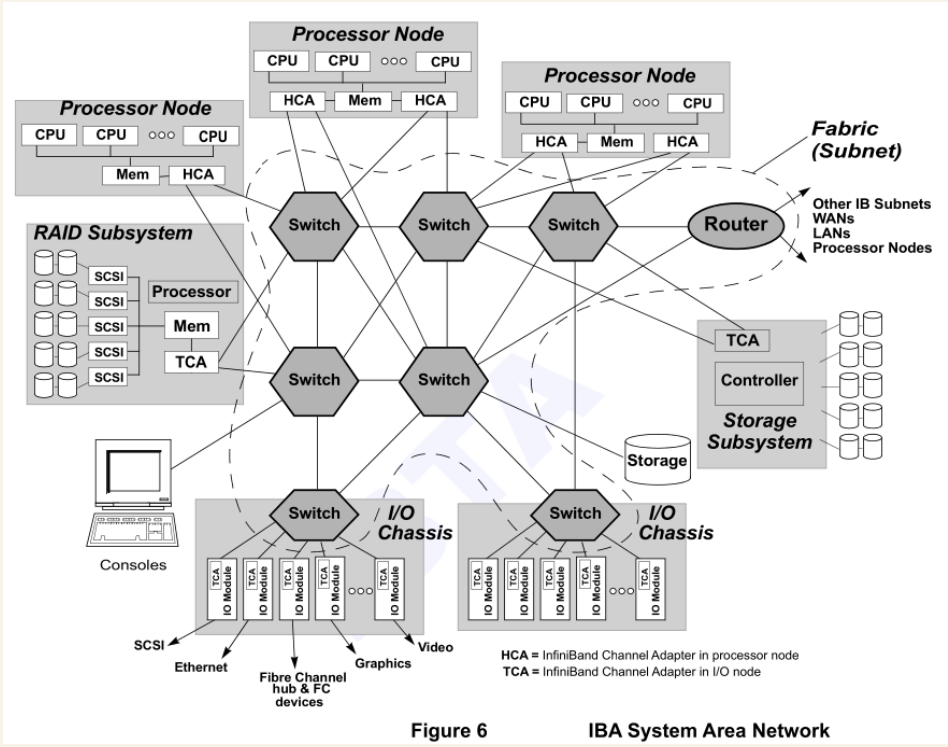


Figure 2: Orderlock is a self-blocking deadlock

Case Study: Infiniband





- 与PCIe不同(P.24)，Infiniband 是一种采用 逐跳 Credit 无损流控，**但端到端 Replay 可靠传输 (RC模式)** 的互联协议：**第一次产业级远征：把总线搬出机箱，一根 fabric 统管计算与 I/O。**
- 同期 HPC 互联之争——Myrinet、Quadrics、SCI、以太网——最后输给了 Infiniband。不过，随着 AI 热潮来临，以太网正借 RoCE/UEC 以及其他新的协议回归。
- IB 赢在工程收敛：标准协议固化进 ASIC，以极低时延和规模成本压过可编程网卡。

Case Study: Infiniband

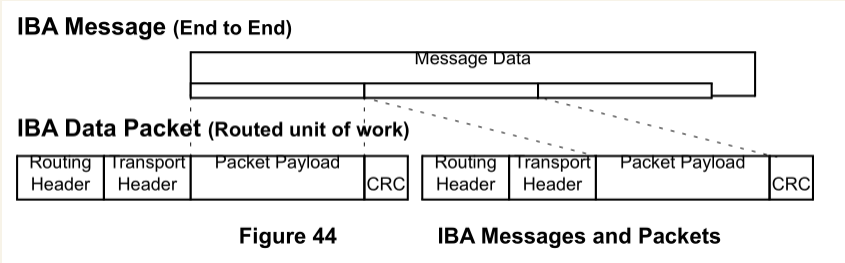


Figure 44 IBA Messages and Packets

bits / bytes	31-24		23-16			15-8	7-0
0-3	VL	LVer	SL	Rsv2	LNH	Destination Local Identifier	
4-7	Reserve 5	Packet Length (11 bits)				Source Local Identifier	

Figure 55 Local Route Header (LRH)

bits / bytes	31-24			23-16				15-8	7-0
0-3	OpCode			SE	M	Pad	TVer	Partition Key	
4-7	F/Res1 [a]	B/Res1 [a]	Reserved 6 [a]	Destination QP					
8-11	A	Reserved 7		PSN - Packet Sequence Number					

Figure 67 Base Transport Header (BTH)

bits / bytes	31-24	23-16	15-8	7-0
0-3	Virtual Address (63-32)			
4-7	Virtual Address (31-0)			
8-11	R_Key			
12-15	DMA Length			

Figure 70 RDMA Extended Transport Header (RETH)

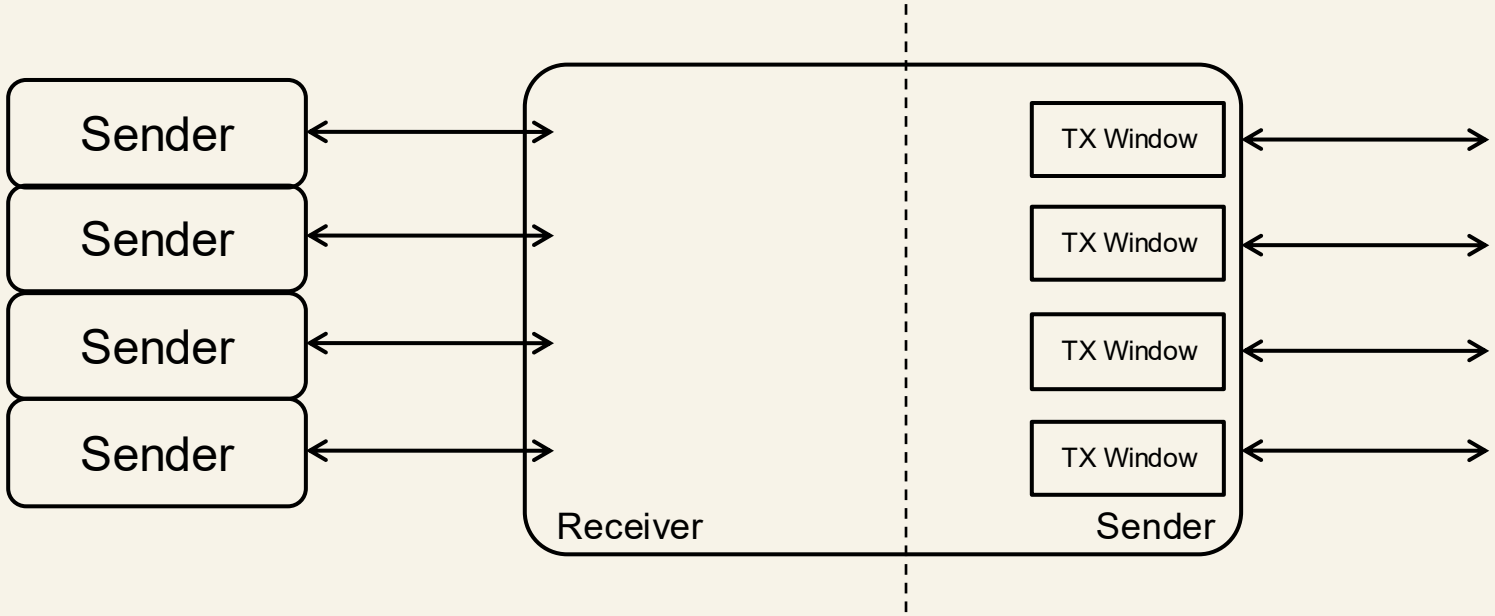
Table 38 OpCode field

Code[7-5]	Code[4-0]	Description	Packet Contents following the Base Transport header[a]
000 Reliable Connection (RC)	00000	SEND First	PayLd
	00001	SEND Middle	PayLd
	00010	SEND Last	PayLd
	00011	SEND Last with Immediate	ImmDt, PayLd
	00100	SEND Only	PayLd
	00101	SEND Only with Immediate	ImmDt, PayLd
	00110	RDMA WRITE First	RETH, PayLd
	00111	RDMA WRITE Middle	PayLd
	01000	RDMA WRITE Last	PayLd
	01001	RDMA WRITE Last with Immediate	ImmDt, PayLd
	01010	RDMA WRITE Only	RETH, PayLd
	01011	RDMA WRITE Only with Immediate	RETH, ImmDt, PayLd
	01100	RDMA READ Request	RETH
	01101	RDMA READ response First	AETH, PayLd
	01110	RDMA READ response Middle	PayLd
	01111	RDMA READ response Last	AETH, PayLd
	10000	RDMA READ response Only	AETH, PayLd
	10001	Acknowledge	AETH
	10010	ATOMIC Acknowledge	AETH, AtomicAckETH
	10011	CmpSwap	AtomicETH
	10100	FetchAdd	AtomicETH
	10101	Reserved	Undefined
	10110	SEND Last with Invalidate	IETH, PayLd
	10111	SEND Only with Invalidate	IETH, PayLd
	11000-11111	Reserved	undefined

Table 43 Schedule of Valid OpCode Sequences

Previous Packet OpCode	Valid OpCodes for Current Packet
None e.g., first packet following connection establishment	"First" packet "Only" packet
"First" packet	"Middle" packet (message is 3 or more packets) "Last" packet (message is exactly 2 packets) Type of operation must match the previous OpCode
"Middle" packet	"Middle" packet "Last" packet Type of operation must match the previous OpCode
"Last" packet	"First" packet (1st packet of a new message) "Only" packet (1st packet of a new single packet msg)
"Only" packet	"First" packet "Only" packet

Open Question: What about Lossy Fabric Router?



1.3 From Connectivity to Management

管理域、Scale-Up 与 Scale-Out

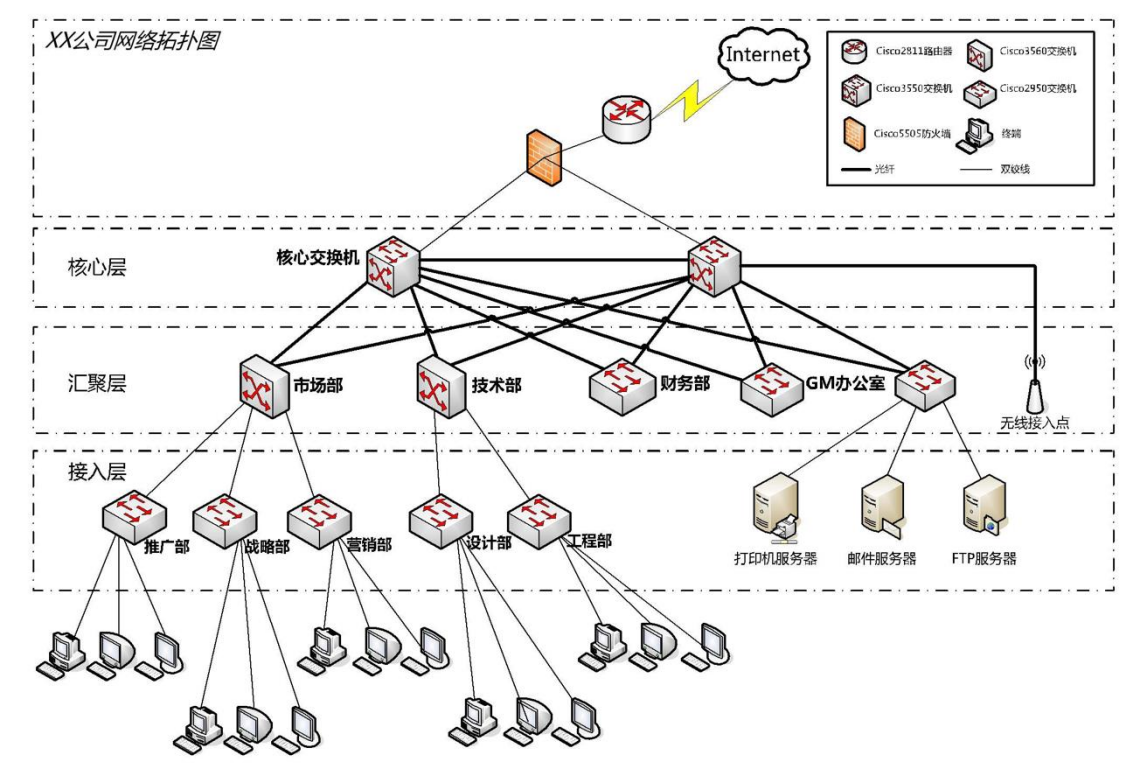
“A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable.”

—— 莱斯利·兰伯特

超节点集群

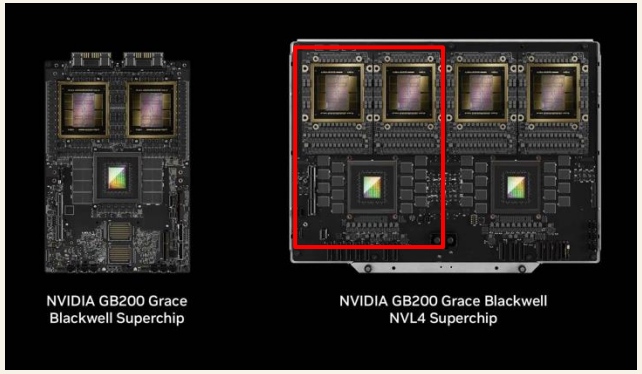
Supermicro NVIDIA GB200 NVL72 · Supermicro

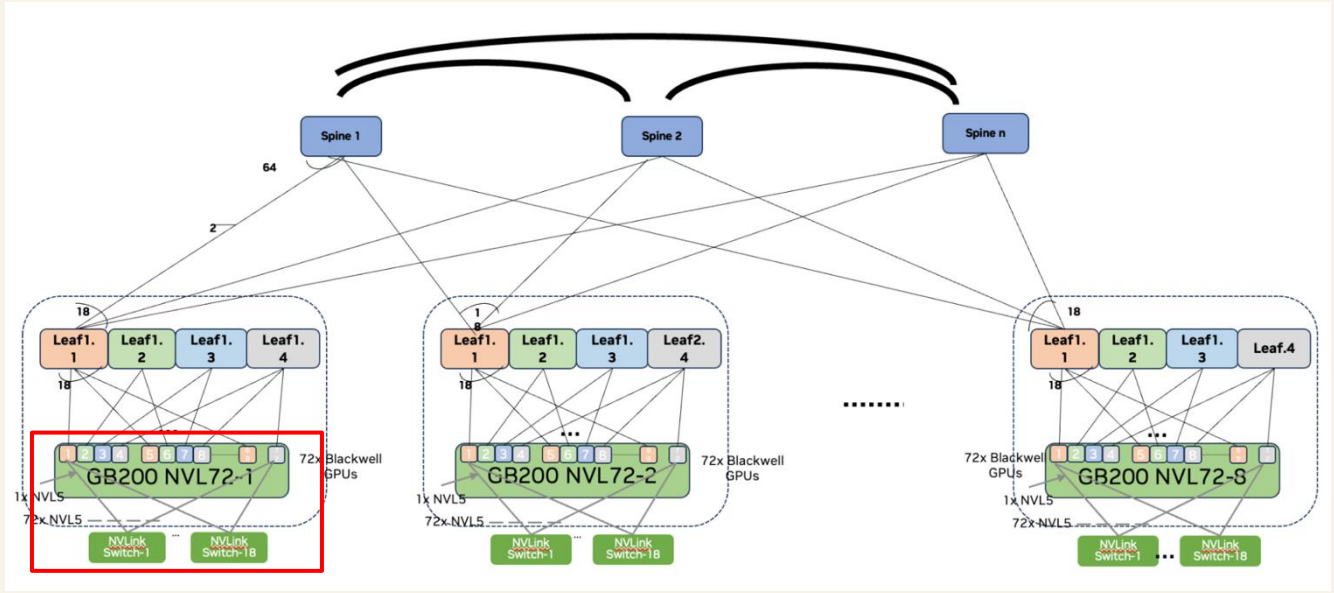
Domain and Connection

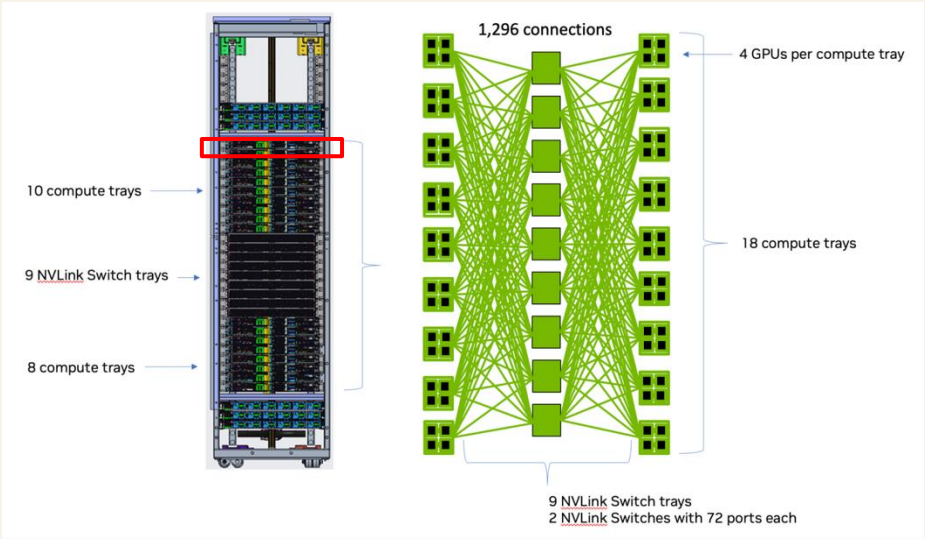


- ✓ **域**(Domain)：存在既定机制(强或弱)访问同一份内存状态的范围。两个域之间彼此不直接共享内存状态。
- ✓ **连接**(Connection)：两个域之间需要经由接入端口建立的**虚拟**传输关系。连接具有明确的生命周期和访问权限。
- ✓ **嵌套**(Nest): 在 Domain 内可以构造 Sub-Domain 以及虚拟的连接体系。也可以将 通过连接组成的多个 Domain 合并为一个 Mega-Domain。
- ✓ **操作系统**(OS)：**内存管理的权威**，一台计算机可以由一个 OS 管理，也可以是分布式的多个 OS 分别管理，并利用某种机制完成协作管理。

Case Study: NVIDIA Grace Blackwell NVL72 SuperPod





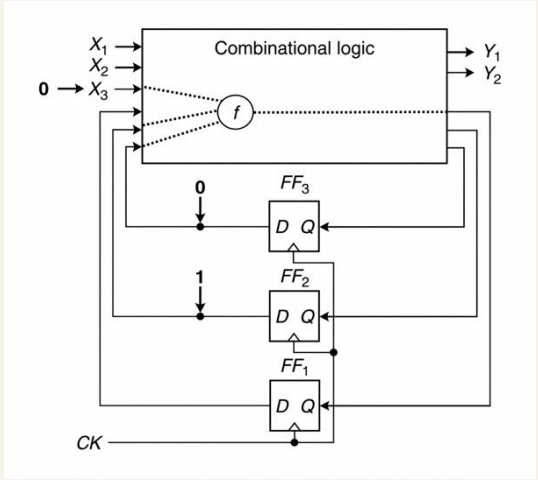


- 一个 NVL72 机柜是一个 Domain：18 个计算**托盘(Tray)**每个包含两个节点，由 1 颗 Grace CPU 与 2 颗 Blackwell GPU组成，他们之间采用 NVLINK-C2C **一致性(Coherence)**总线通讯。节点之间通过 NVLINK 互联系统(**Scale-Up**)直达。不同的 NVL72 系统通过 Infiniband 网络连接(**Scale-Out**)，在发起通讯前需要显式发起连接。
- 每个节点有独立的 OS 对本节点的**一致性内存**范围进行管理(为什么？)，节点通过 CUDA 完成协商，实现内存映射以支持全域的内存访问。

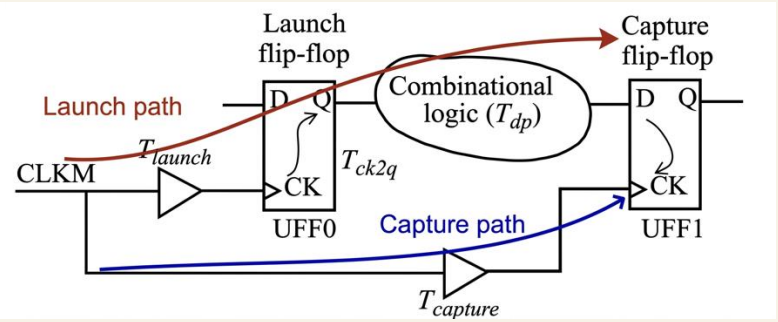
Computer System: State and Computing

- 计算机是一门研究**状态**与计算(**状态转移方程**)的学科。
- **状态是被记住的东西，计算让状态改变的过程。**状态放在哪里，如何命名，谁能访问，何时可见，跨过一道边界要花多少时间、付出多大代价——这些问题的答案，决定系统最后长成什么样。

$$Q^* = f(Q, Input)$$
$$Output = g(Q, Input)$$



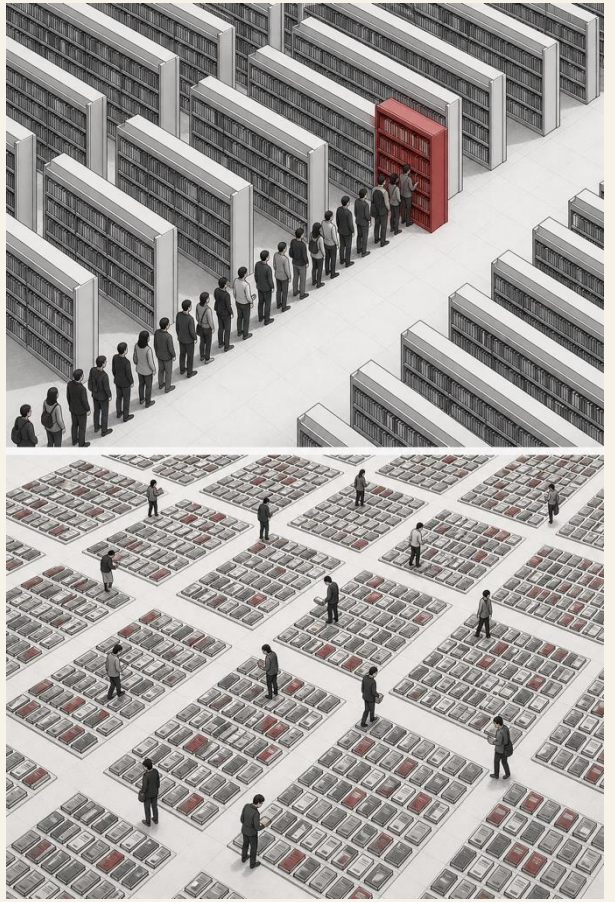
- 今天的计算机系统要容纳更大的模型、更多的数据和更密集的协同。**没有一颗孤立的芯片装得下这些野心。**有限，所以外溢：**层次化存储**兜状态，**分时复用**兜方程；而状态与方程在空间上的外溢，都叫**分布式**——它的出生证明，就是互联。
- 沿这条线看，这门学问在很大程度上由**取舍(Trade-Off)**写成：寄存器、缓存、主存、远端内存，是状态在容量、延迟、一致性与共享范围之间的取舍；ISA 与 DSA，是方程在通用与专用之间的取舍。链路与网络也不是理想中的传输介质，受到时延和带宽的约束。



Library Bisearch Problem

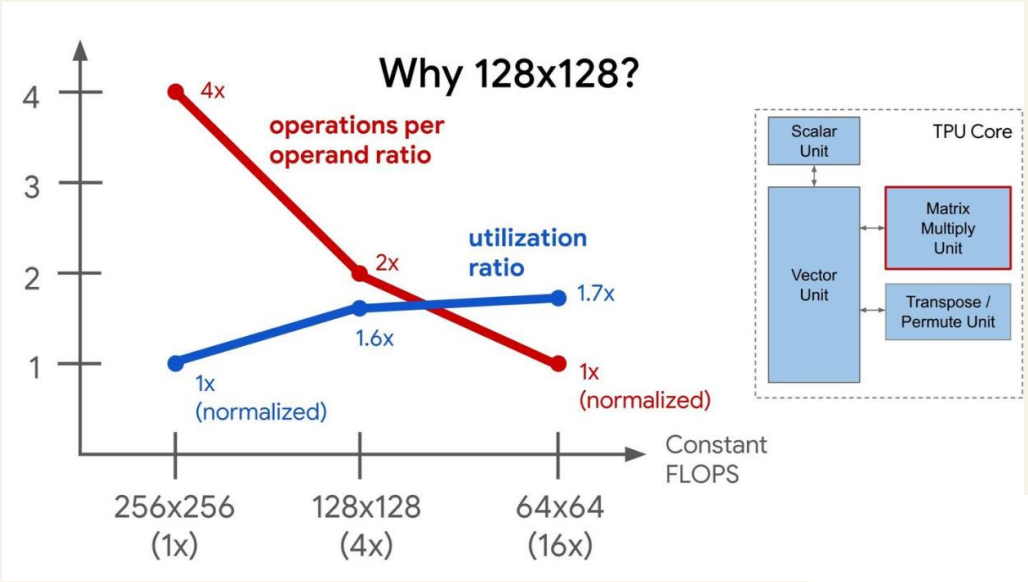
图书管理员问程序员怎么找书最快。程序员说：二分查找——先走到图书馆正中，看书号，决定去左半边还是右半边，再取中点，重复 log n 次就到了。

- 书架不是排好序的——算法的前提是有人维护了"有序"这个状态，而维护它是真金白银的工作；
- "走到正中"不是 O(1)——随机访问在物理上不存在，腿是顺序访问设备。理论上 log n 次比较，实际上 n 次走路。



"内存墙"是不可能解决的，它不是工程缺陷，是物理和经济的联合约束。速度、容量、成本构成一个三角形，你只能选两个角占优，永远买不到三个。

The Fundamental Fact



Operation	Energy [pJ]	Relative Cost
32 bit int ADD	0.1	1
32 bit float ADD	0.9	9
32 bit Register File	1	10
32 bit int MULT	3.1	31
32 bit float MULT	3.7	37
32 bit SRAM Cache	5	50
32 bit DRAM Memory	**640**	**6400**

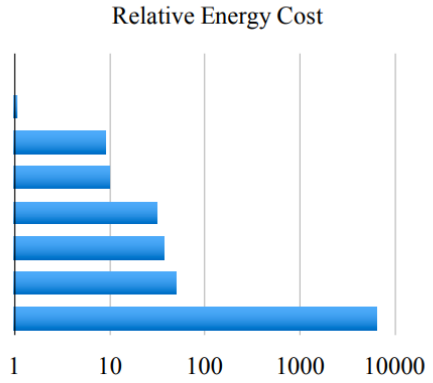


Figure 1: Energy table for 45nm CMOS process [7]. Memory access is 3 orders of magnitude more energy expensive than simple arithmetic.

Synchronization

同步是系统里最贵的操作——它本质是跨距离达成共识，无法被掩盖，只能被推迟或批处理。

成本下限由物理距离与可扩展范围决定，因此衍生了不同的机制：

层次	机制	成本(数量级)	例子
核内	乱序执行顺序退休（或同步点）/ 顺序执行乱序写回	cycles	OoO CPU, SIMT
核间（同 SoC）	一致性协议 + 屏障	~10 ns–1µs	MESI, DMB
核 ↔ 设备（提交）	Programmed I/O / Doorbell	~100 ns–µs	BlueFlame, SQ, cp.async
核 ↔ 设备（完成）	轮询 / 中断(用户,内核)	~100 ns–µs	evq, UINTR, IRQ, mbarrier
分布式	Replicate, Fence	>1 RTT	Paxos, Raft

The Same Story: Prior? Posterior?

如 P.19，

每个"能不能做"的问题，只有两种回答方式：

- 先验：动作之前拿到许可
- 后验：先动作，再等裁定

请求：推，直到反压（先验）——Programmed I/O

拉，放单敲门（后验）——Doorbell

完成：我早知道（先验）——计时器 / **轮询**(Polling)

对方来叫（后验）——**中断**(Interrupt)

Case Study: Audio Device Timer Model

一个反直觉的案例：音频设备不用中断

音频有铁节拍：采样率恒定、缓冲周期恒定——数据何时被需要，完全可预测。

- ✓ **中断模型**(Poettering 2008)：

  缓冲：大防断音 / 小求延迟　　碎片：大省电 / 小防断音

- ✓ **计时器轮询**：

  处理对齐到周期边界——无 IRQ 抖动、无中断处理程序、主循环完全确定。

- 结果：省掉每秒成百上千次特权级往返，**处理器睡眠不被打断**

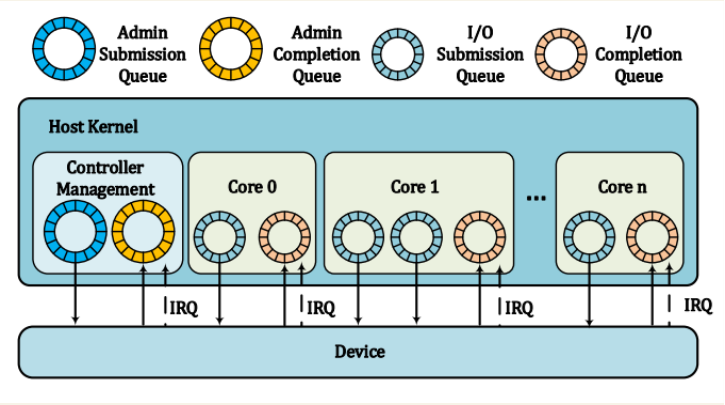
PulseAudio

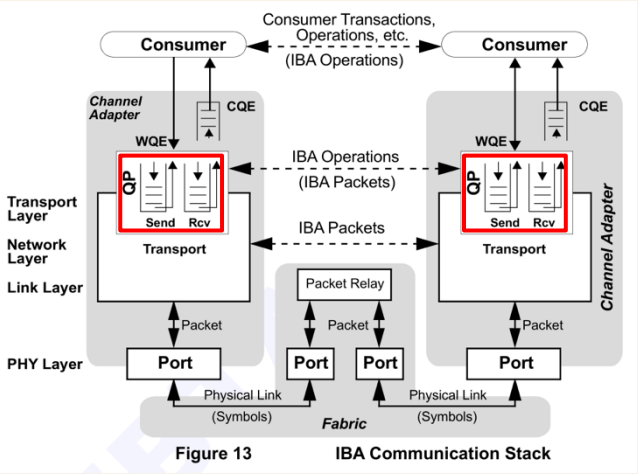
[● ◀] systemd

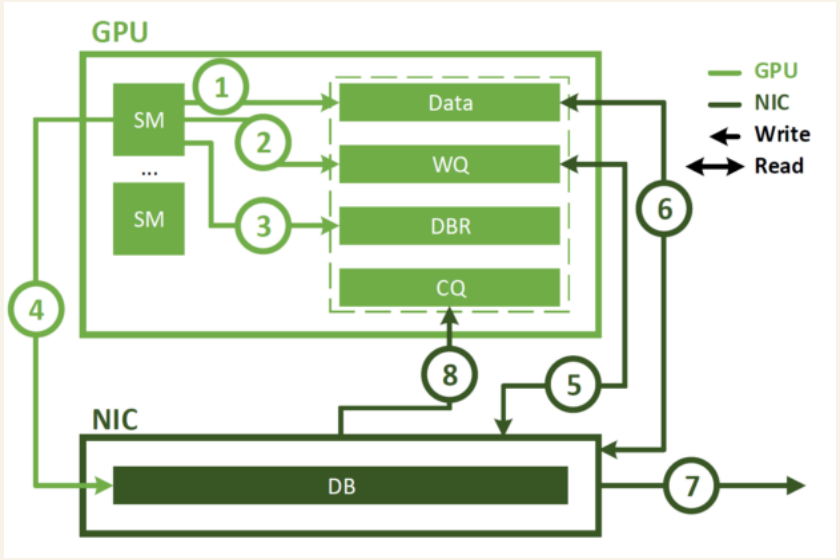
同样的故事：

- 游戏（Valve Source 引擎）：服务器固定 tick，到点结算——计时器模型（先验）；客户端预测先动，服务器滞后补偿事后裁定——后验。手感是后验买的，公平是先验保的。
- 协程（Coroutine）：让渡点预先约定（计时器）；抢占式线程：随时被打断（中断）。

SQ/CQ Model







[一条命令的一生]():

① Host 写入位于 DRAM 中的 SQ(WQ, Work Queue)。

② 向一个属于 Device 的地址空间执行一个写入操作（MMIO）。这个操作被解码为一个 Doorbell 事件。

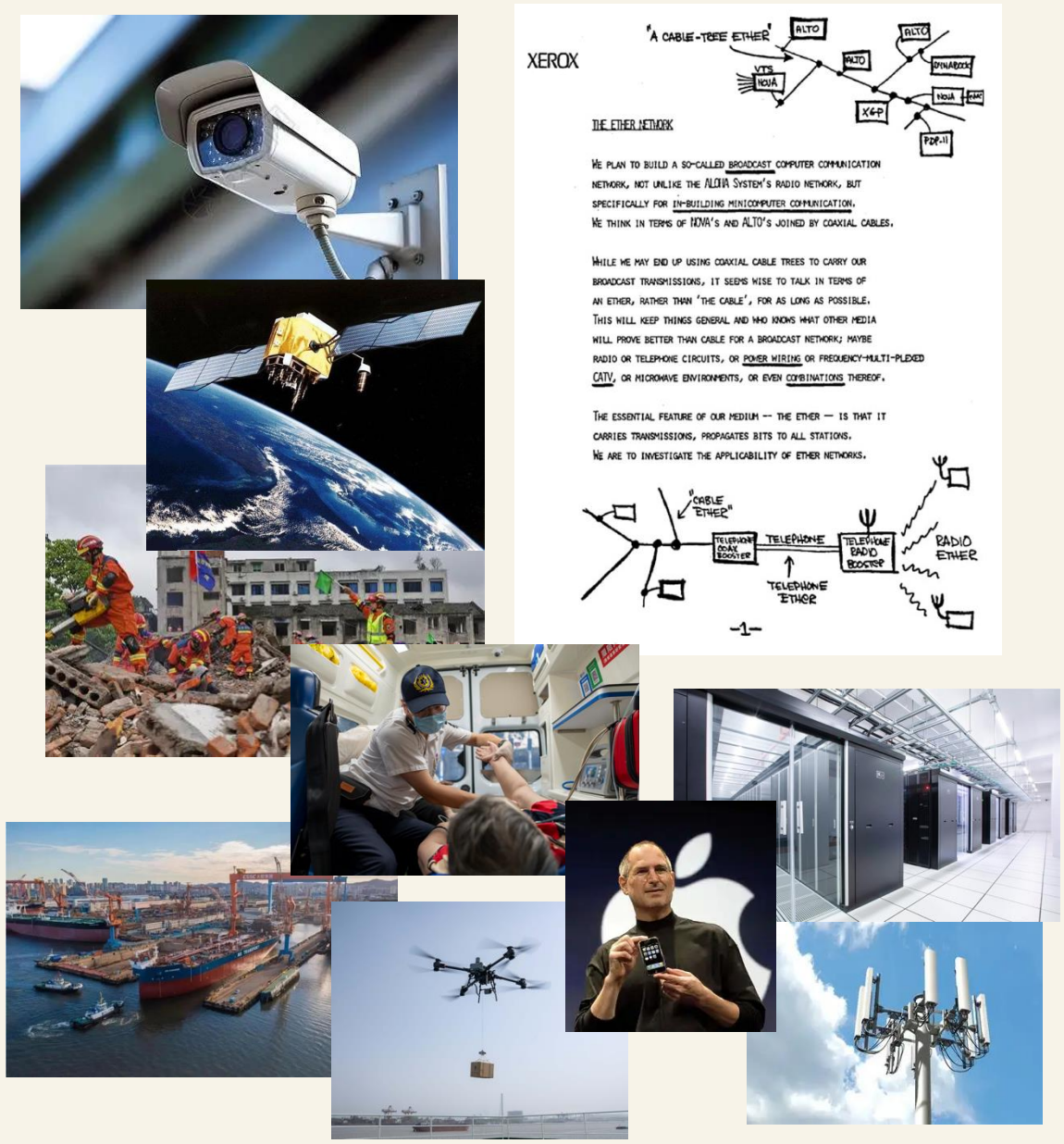
③ 设备收到了这个 Doorbell (相当于 Host 向 Device 发出一个中断)，从 SQ 里面取出任务。

④ 执行，并写回产物到 CQ。

⑤ Device 发起中断(Doorbell 的反向)，拉起信号线。Host 跳转至中断处理程序，将产物取出。

Case Study: Ethernet

- ✓ **以太网无处不在**：现代社会的通信基础设施。
- ✓ ALOHA 无线网络（夏威夷大学，~1970）——最早的分组无线电网之一。ALOHA 本是夏威夷语的问候语；随后思想被以太网原样继承。
- ✓ 施乐公司 PARC 研究中心 Bob Metcalfe 与 David Boggs 在 1973, May 22 的备忘录中提出了 "The **ETHER** Network" 的构想。
- ✓ 以太，即与介质无关：Radio Ether, Cable Ether, Telephone Ether……无实体，所以无处不在。



Case Study: Ethernet

- 网卡也叫网络适配器，全称**网络接口卡 NIC**（Network Interface Card）。主机通过 NIC 连接和访问以太网。

✓ 对内（内存世界）提供队列：描述符、DMA、doorbell、中断

✓ 对外（网络世界）完成成帧、符号、介质。网络包是**原子(atomic)**的。

- **MAC** 提供接口接入地址(称为 MAC 地址)与寻址过滤、介质访问控制(CSMA/CD半双工碰撞检测，千兆起弃用或删除)、CRC 校验等功能，通过 MII (**介质无关接口**)与PHY连接。
- **PHY** 提供编码与调制等功能，将数据编码输出到 MDI (**介质有关接口**)上，经由网络变压器或光模块等输出到介质上。

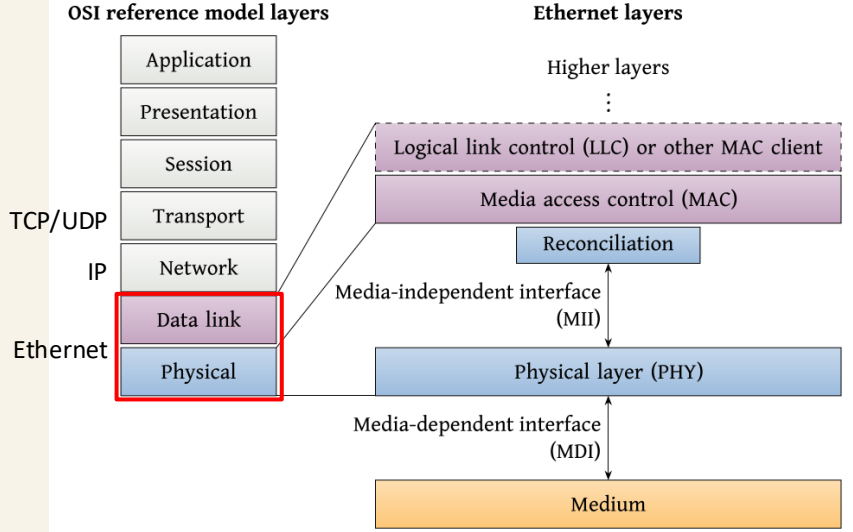


Figure 2.1: IEEE 802.3 simplified layer architecture in relation to the OSI reference model. Adopted from [5, fig. 1-1].

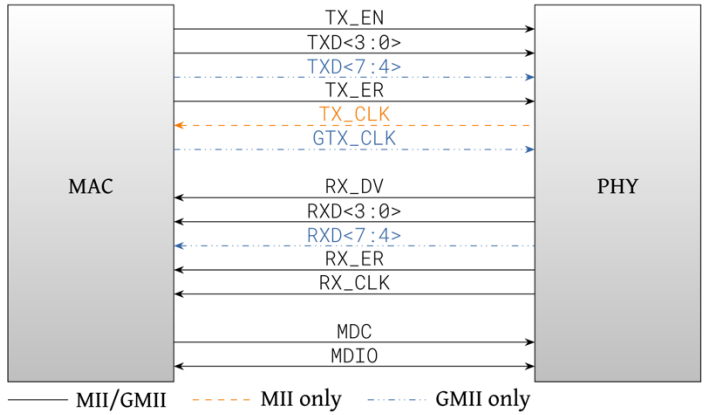
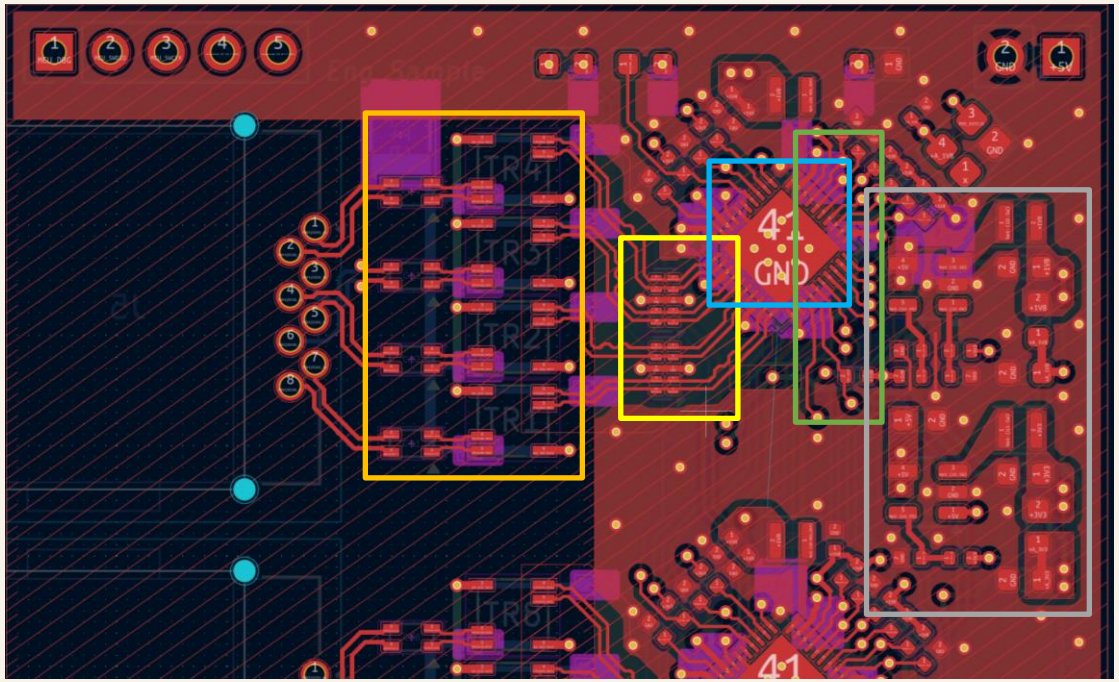


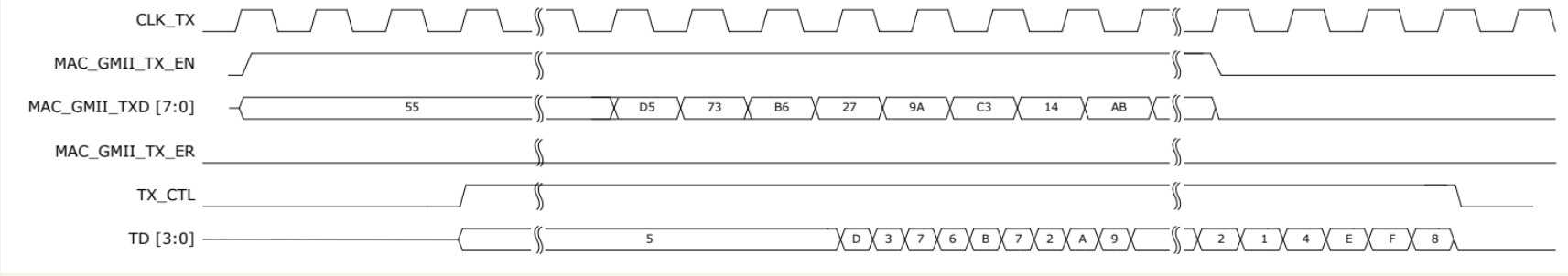
Figure 2.2: Signals used by MII and GMII in full-duplex operation. Adopted from [5, fig. 22-3] and [5, fig. 35-2].

Case Study: Ethernet



- 蓝色：PHY 芯片
- 灰色：电源轨
- 黄色：MDI 串行信号，至橙色区域
- 绿色：RGMII 并行信号，至主机
- 橙色：网络变压器与共模滤波器

Figure 4-1. CoreRGMII Timing Diagram of Packet Transmitted



Case Study: Ethernet (L2)

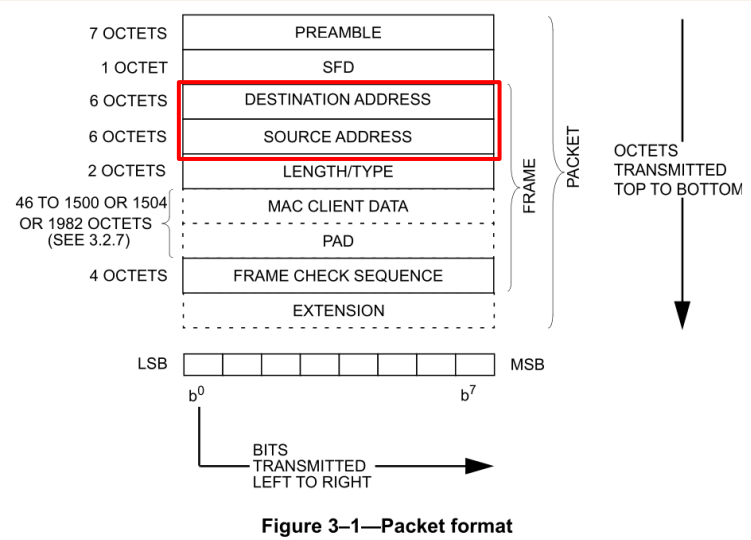
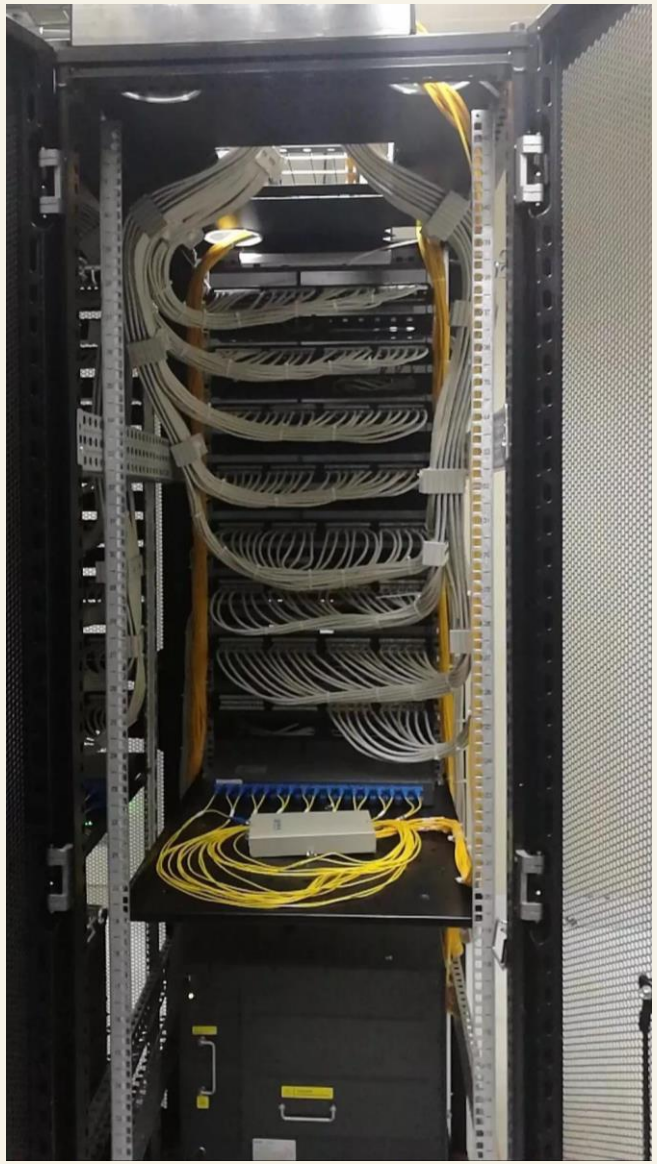
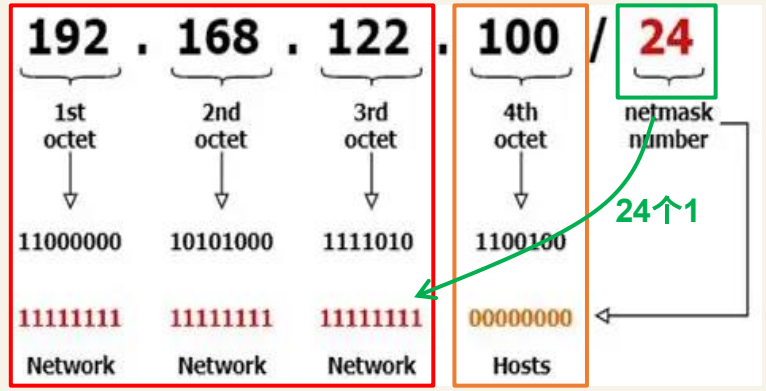


Figure 3–1—Packet format



- ✓ MAC 地址：48 bit = OUI（厂商编号）+ 流水号，出厂自带
- ✓ 交换机自学习：看源地址记账——“这个源地址从这个口进来”；表项会老化（默认 300s）。没有控制协议，没有配置：转发表全部由到达的帧养成——即插即用。
- ✓ 尽力而为转发：认识目标地址 → 单口投递；不认识或广播（FF:FF:...） → 洪泛。

Case Study: IP (L3)



- ✓ 交换机缓冲区满了怎么办？丢包
- ✓ 交换机转发表满了怎么办？层次化 => **IP 协议**，**可聚合**的表项

所以 L3 走得出园区，L2 走不出。命名决定容量，容量决定边界。

CIDR: 一个 IP 地址分为**前缀**和**后缀**，用**掩码**区分。

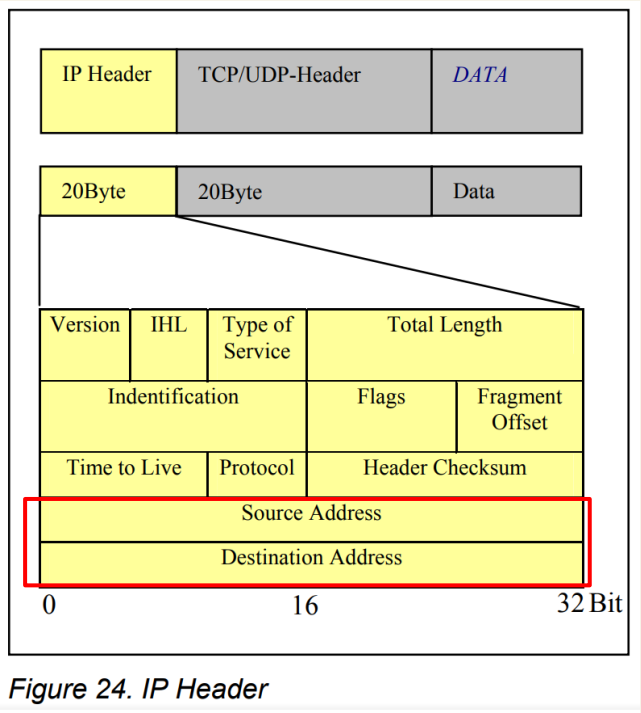
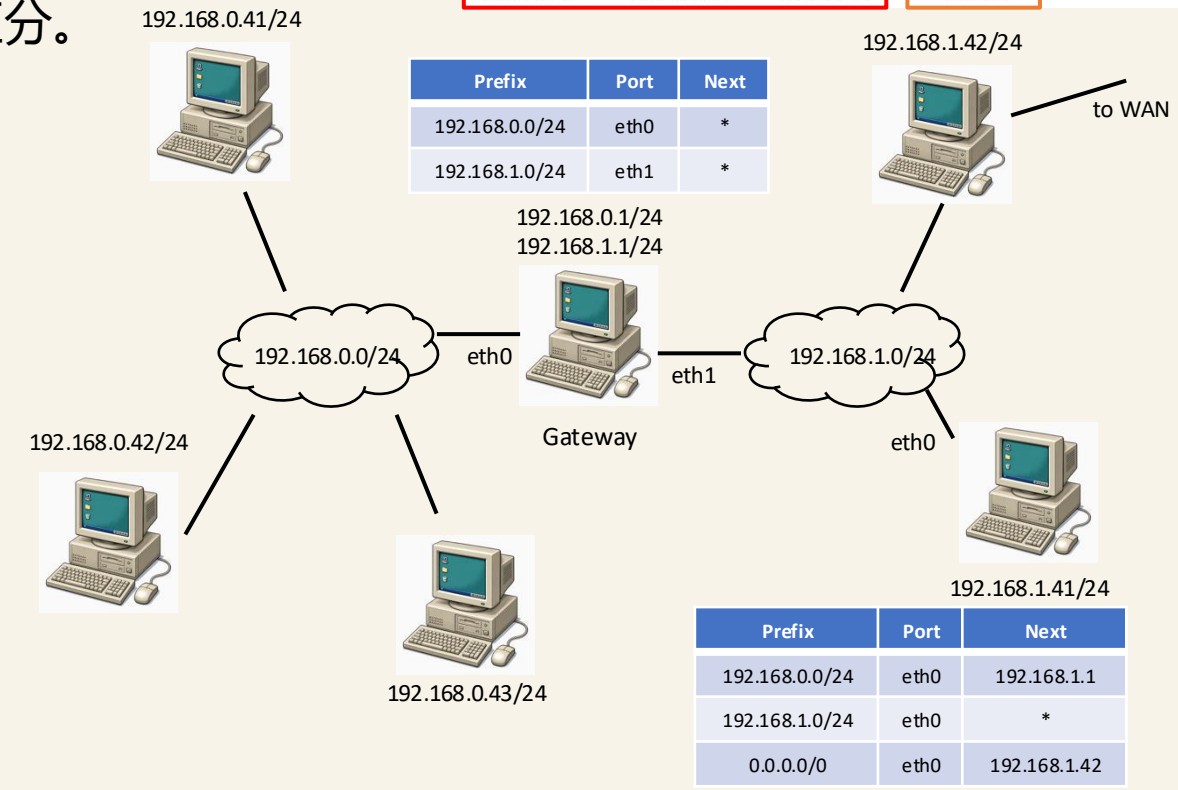
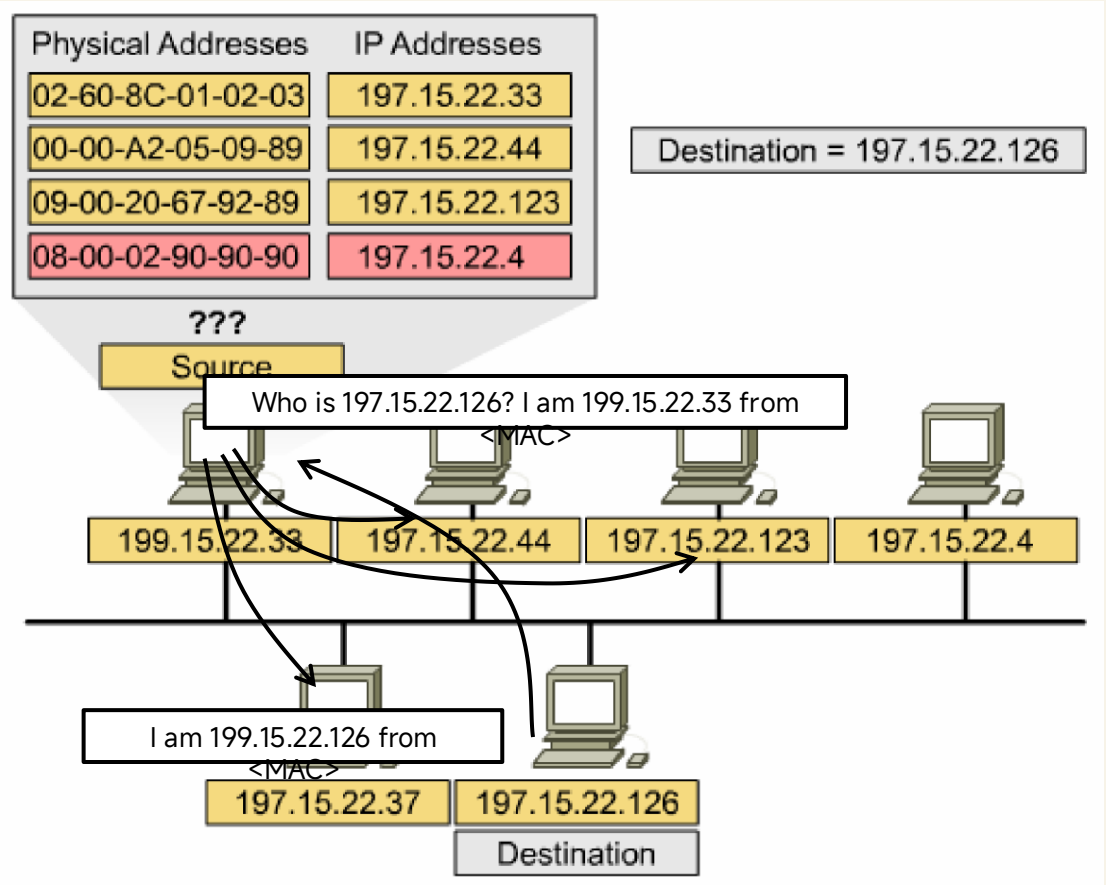


Figure 24. IP Header



Case Study: IP



ARP：知道目的地的 IP，但线上只认 MAC。

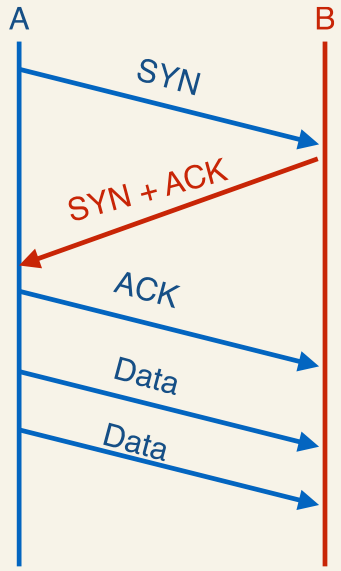
① 广播：谁是 197.15.22.126?

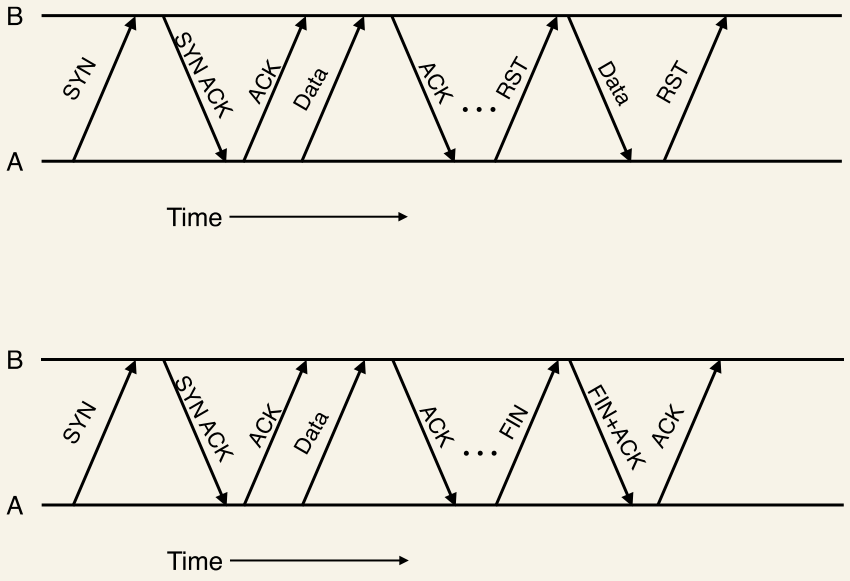
② 单播：我是 197.15.22.126，MAC地址……

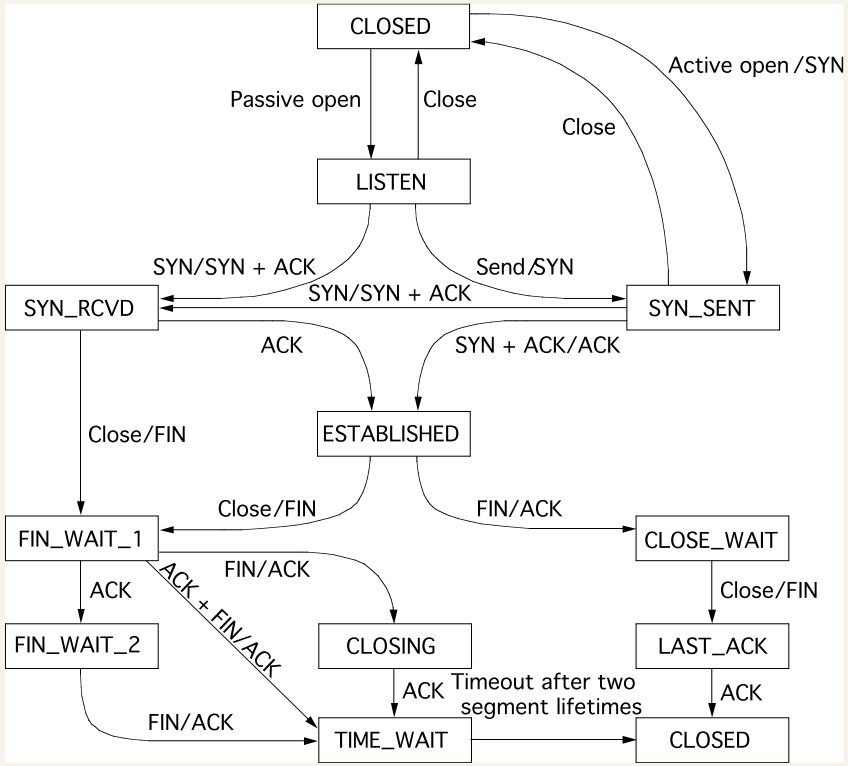
③ 答案缓存进 ARP 表

身份：EtherType 0x0806，与 IP 平级——ARP 不是 IP 的一部分，是 L2/L3 之间的胶

Case Study: TCP/IP







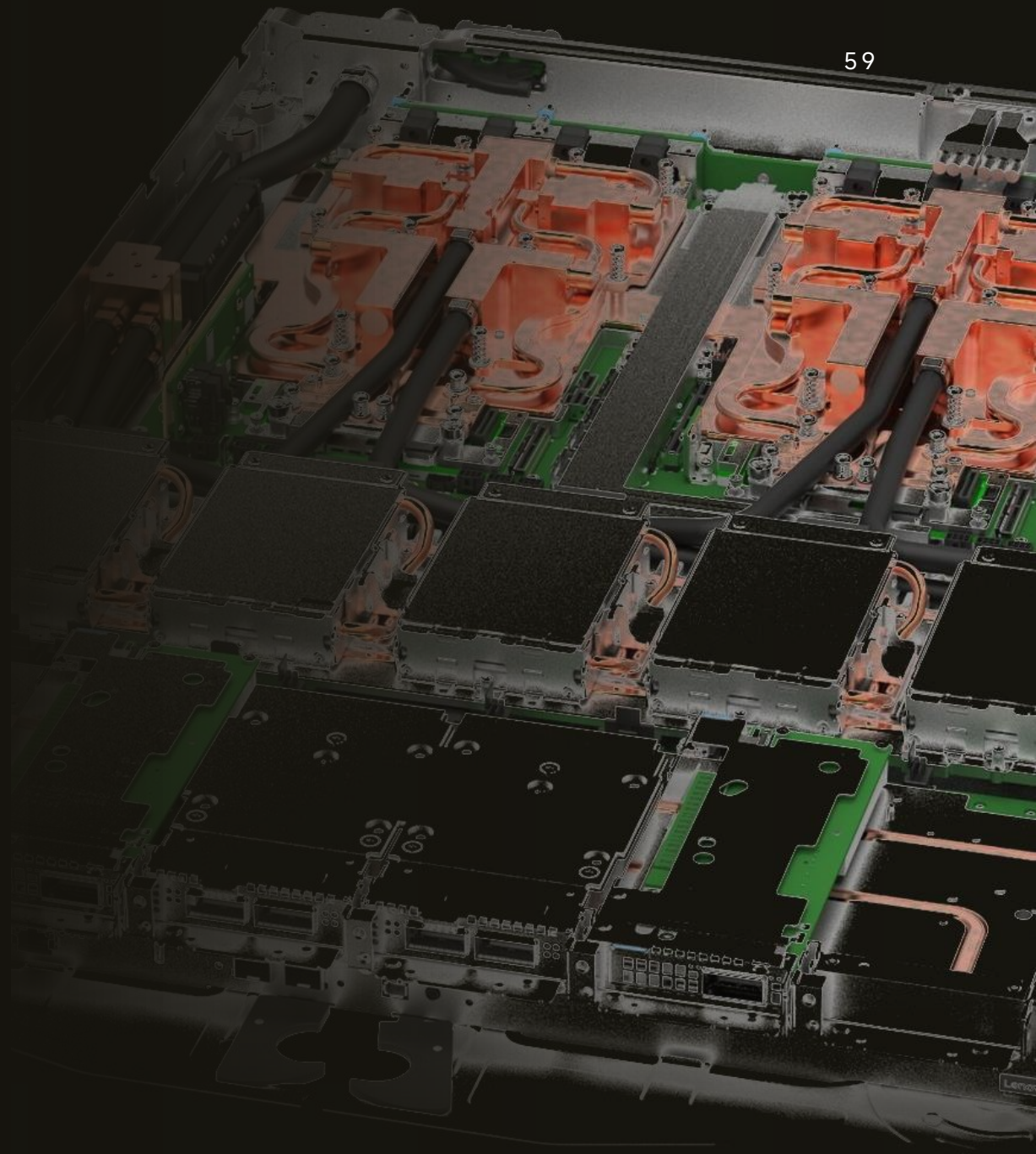
以太网丢包不道歉，IP 尽力而为——可靠、顺序、去重，底下两层都不管。IP 提供的互联抽象没有流控、也不可靠，TCP 在这层抽象上把它们补出来（机制本讲不展开）。

- 出生——三次握手：互报初始序号，防古代 SYN 复活
- 死亡——两次单向告别：FIN 占一个序号、必须 ACK；"我说完了" 不等于 "你说完了"。
- 守灵——TIME_WAIT 2MSL：等迟到包死透 + 给丢失的最后 ACK 兜底（Linux 60s）
- 暴毙——RST：不告别、不确认、在飞数据丢弃

PART II

II

Inside the Monolith: Network Micro Architecture

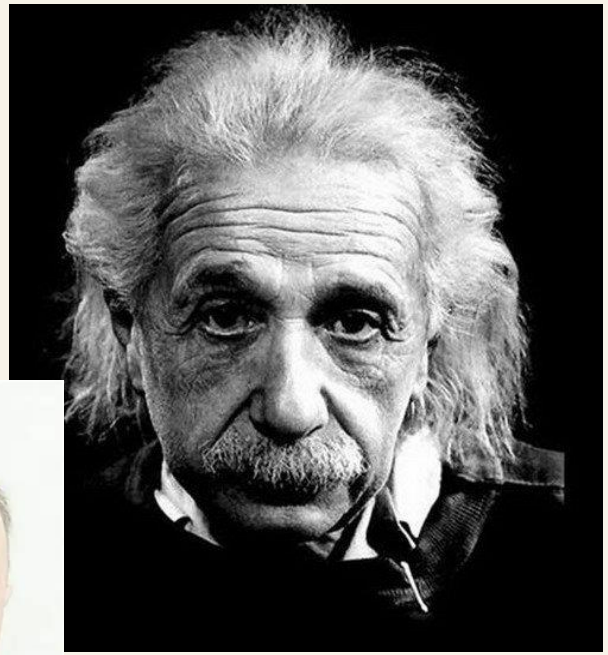


Principles of Research

在开始这一章前，请允许我先做一次寄予。

你们中，会有人写通信库、训练 AI、做 Infra、造芯片、定协议。这些工作表面是工程，深处是在为人类共同的计算世界立法。山河等待你们建造，人间需要前行的灯火；你们是探索家与建筑师，是接过灯火的人。

但这图景不是没有重量的。它更像山野：要读天气、认窗口、定下撤线，任何决定都有后果。冒险追逐危险，极限管理风险；冒险把后果交给运气，极限把后果收回纪律。真正让人心潮澎湃的，不是危险，而是明知畏惧仍敢负责的那一刻。愿你们把回程交到彼此手里，用一生去挑战足够真、足够难的问题。



Principles of Research

- 在科学的殿堂中，有许多楼阁，居住其中的人真是各式各样，而引导他们走向那里的动机也各不相同。有许多人热衷于科学，是因为科学赋予他们**超乎常人的智力快感**；科学是他们自己的一种特殊娱乐，他们在这种娱乐中寻求**生动的经验**和对**自身雄心的满足**。在这座殿堂里，另有许多人是出于纯粹功利的目的，将他们大脑的产物献给祭坛。如果有一个天神来驱逐这两种阶层的人，那么聚集在那里的数量会大大减少，但仍会留下一些古人，也有今人，我们的普朗克就在其中，这正是我们敬仰他的原因。

- 他们大多是沉默寡言的、相当奇特和孤独的人。然而，尽管拥有这些共同的特征，他们之间却不像那些被排挤的人那样彼此相似。究竟是什么力量将他们引向这座殿堂呢？这是一个难题，无法用一句话概括。首先，我同意叔本华的观点：促使人们向艺术和科学迈进的最强大动机之一，是**逃避日常生活中令人厌恶的粗俗和使人绝望的沉闷**，是**摆脱自由变幻不定的欲望的枷锁**。一个修养有素的人总是渴望逃离个人生活，进入**客观知觉和思想的世界**——这种愿望就像城市里的人渴望逃离熙来攘往的环境，去高山享受幽寂的生活。在那里，**透过清净纯洁的空气，可以自由地眺望、沉醉地欣赏那似乎是为永恒而设计的宁静景色。**

Principles of Research

- 除了这种消极的动机外，还存在一种积极的动机。**人们总想以最适合于他们自身的方式，为自己画出一幅简化且易于理解的世界图景；然后，他们又试图用这个由他们构建的宇宙体系来取代经验世界，并以此征服后者。**画家、诗人、思辨哲学家和自然科学家都在各自的方式下做着这一切。他们将世界体系及其构成视为自己情感生活的中心，从而在个人经验的狭小范围内寻找无法企及的宁静与安定。

- 物理学家的最高使命是找到那些可以**由纯粹演绎法建立世界体系的普适性基本定律。**要通向这些定律，**没有逻辑推理的捷径**，只有**建立在经验的同感理解之上的直觉。**由于这种方法论上的不确定性，人们可能会推测，理论物理学存在多种可能同样适用的体系，这个看法在理论上无疑是正确的。但物理学的发展表明，**在某一时期，总有一个比其他体系都更为精妙的。**凡是真正深入研究过这一问题的人，都不会否认，唯一决定理论体系的，实际上是现象世界，尽管在现象与理论原理之间并没有逻辑的桥梁——这正是莱布尼茨非常准确地表述过的“**先天的和谐**”。
- 物理学家常常责备研究认识论的人没有对此做出足够的关注。我认为，几年前**马赫和普朗克的争论**的根源就在于此。

Principles of Research

- 在所有可能的图景中，理论物理学家的世界图景占据何种地位呢？在描述各种关系时，它要求对精确性的标准达到只有数学语言才能达到的最高境界。另一方面，物理学家必须极其严格地限制自己的研究主题：他必须满足于描述我们经验领域中最简单的事件。对于任何试图以理论物理学家所要求的精密性和逻辑完备性来重现更复杂事件的企图，都超出了人类理智所能及的范围。**这以完整性为代价，换取了至高无上的纯粹性、清晰性和确定性。**但是，当人们胆小谨慎地将所有更复杂而难以捉摸的东西撇开不管时，究竟是什么吸引我们去认识自然界的这一微小部分呢？这种谨小慎微的努力的成果，是否就足以配得上“宇宙理论”这样光荣的称号？依我之见，是足够的。**因为作为理论物理学结构的基石，它所建立的普适性基本定律，理应对任何自然现象都具有有效性。有了这些定律，如果演绎过程并不远远超出人类心智的容量，我们就有可能借助于纯粹的演绎，得出描述一切自然过程（包括生命过程）的理论。因此，物理学家放弃其宇宙体系的完整性，并非一个根本原则上的问题。**

Infiniband and TCP/iWARP worldviews

两幅自洽的世界图景；约束搬家后，中心也要搬家。

Infiniband:

- 世界 = fabric：把片上 NoC 延伸到机箱之外。
- 正确性住在 QP/硬件：credit、顺序、完成语义。
- 靠信任单一域来扩展：无损、固定、故障稀少。
- 有分区，没有租户句柄；多用户靠调度分时，不靠 fabric 并发隔离。
- 历史位置：单域、无损、故障稀少的世界里高度自洽；现代 DC 的多租户与常态丢包让它过期。

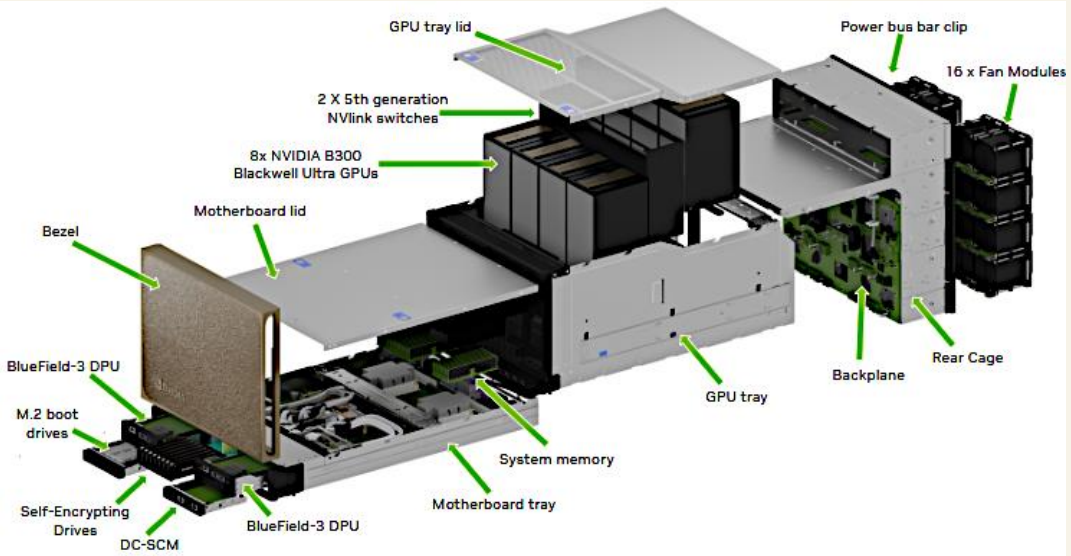
TCP/iWARP:

- 世界 = 端点：软件拥有协议。
- 正确性住在端点：重传、顺序、拥塞控制。
- 靠不信任网络来扩展：超时、DRAM、异构环境。
- 字节流没有消息边界，顺序、重传、拥塞是软件时代的答案。
- 历史位置：互联网尺度与 DRAM/软件状态的世界里自洽；加速器要求硬件契约时开始吃力。

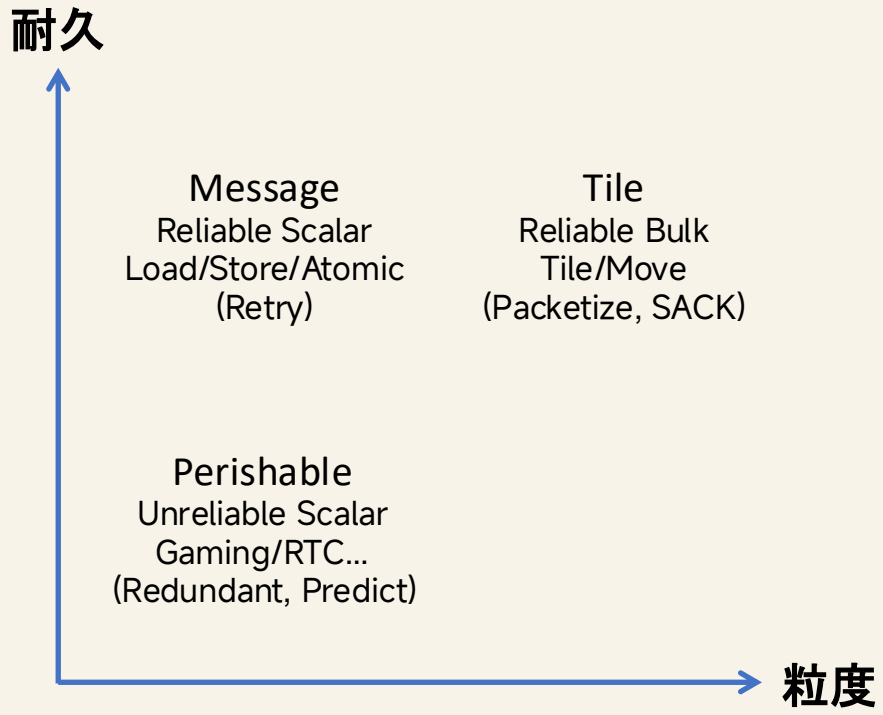
Today Large Distributed Network System Blueprint

- 端点变了：xPU/DMA/AGU 产生地址流，CPU 退到控制面；
- 负载变了：大块聚合、分散与副本成为主流，三网合一(计算、存储、传统服务)。
- 规模变了：十万级节点、百万级租户、海量在途事务，乱序是默认；多管理域并发共享同一张 fabric。
- 常态变了：误码、乱序、丢包、拥塞、迁移都是正常输入；一次超时可能是没到、没完成，或结果没回来。

Figure 1: B4 global topology. Each marker indicates a site or multiple sites located in close geographical proximity. B4 consists of 33 sites as of January, 2018.



Traffic Dimension



- 没有全局最优：每一格都在为这份流量选择结账方式——时间（RTT/窗口）、带宽（冗余）、状态（连接/授权/调度表项），或者不付（尽力而为）。算力只是收银员，不是商品。
- **域**(P.43)即半径，决定这单采购在哪结账：小半径可在域内把丢失压在链路(CBFC / LLR); 大半径跨域不直接共享内存状态，必须经 **连接** 显式建立生命周期与权限，可靠性回到事务端点。

Old cosmos is crashing

- 传统 RC QP 把五件生命周期完全不同的事焊在一个对象上：

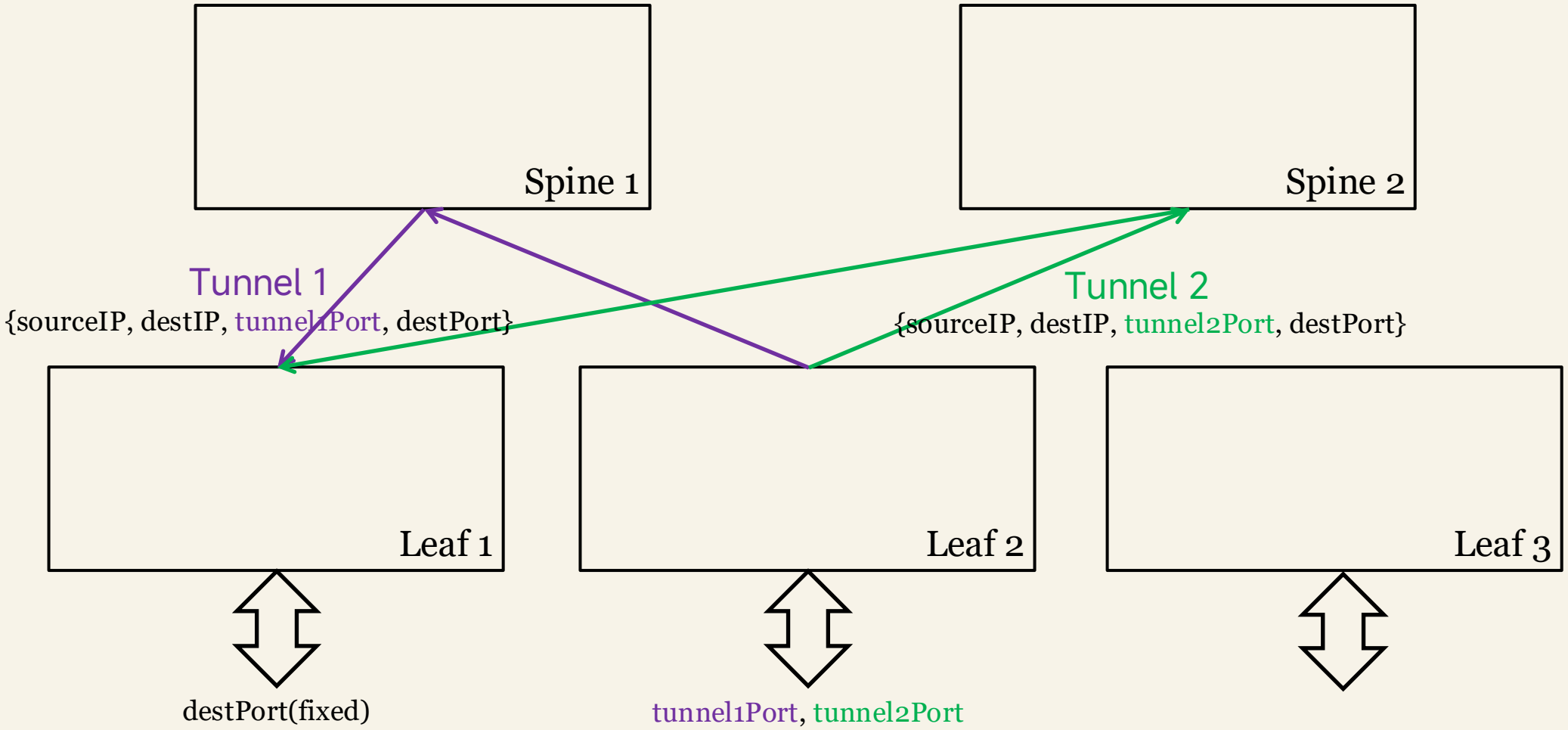
焊进去的东西	它自己的物理需求
身份（QPN 寻址）	长寿命：比一次传输、一条路径都活得久
顺序（PSN 空间）	单一序号：一旦全序，就排斥多路径与乱序
可靠性（窗口/重传）	逐连接驻留 NIC：连接数被 SRAM 上限锁住
路径（五元组）	动态可变：故障绕路、带宽聚合，不该重建身份
异步编程（SQ/RQ/CQ）	CPU↔NIC 共享的内存队列，位置即语义

- 焊死的代价：单路径、Go-Back-N、连接数受 NIC SRAM 限制、PFC 无损地基、出错杀 QP。
- UEC 解开了路径（包喷洒/乱序/选择重传），但单体没散：连接状态仍驻 NIC，接收端也仍有逐连接/逐消息状态的模式；RUDI 证明还能再退，窗口边缘却仍靠契约守。
- 焦油坑：每修补一个轴，其余轴焊得更紧。要多路径，就得多建 QP，或把喷洒丢给传输层；要扩连接数，就得加 NIC SRAM。挣扎越用力，陷得越深。
- 出路：不是继续修补，而是重新分层——五个生命周期，五种物理需求；**让每层只提供它能廉价提供的东西。**

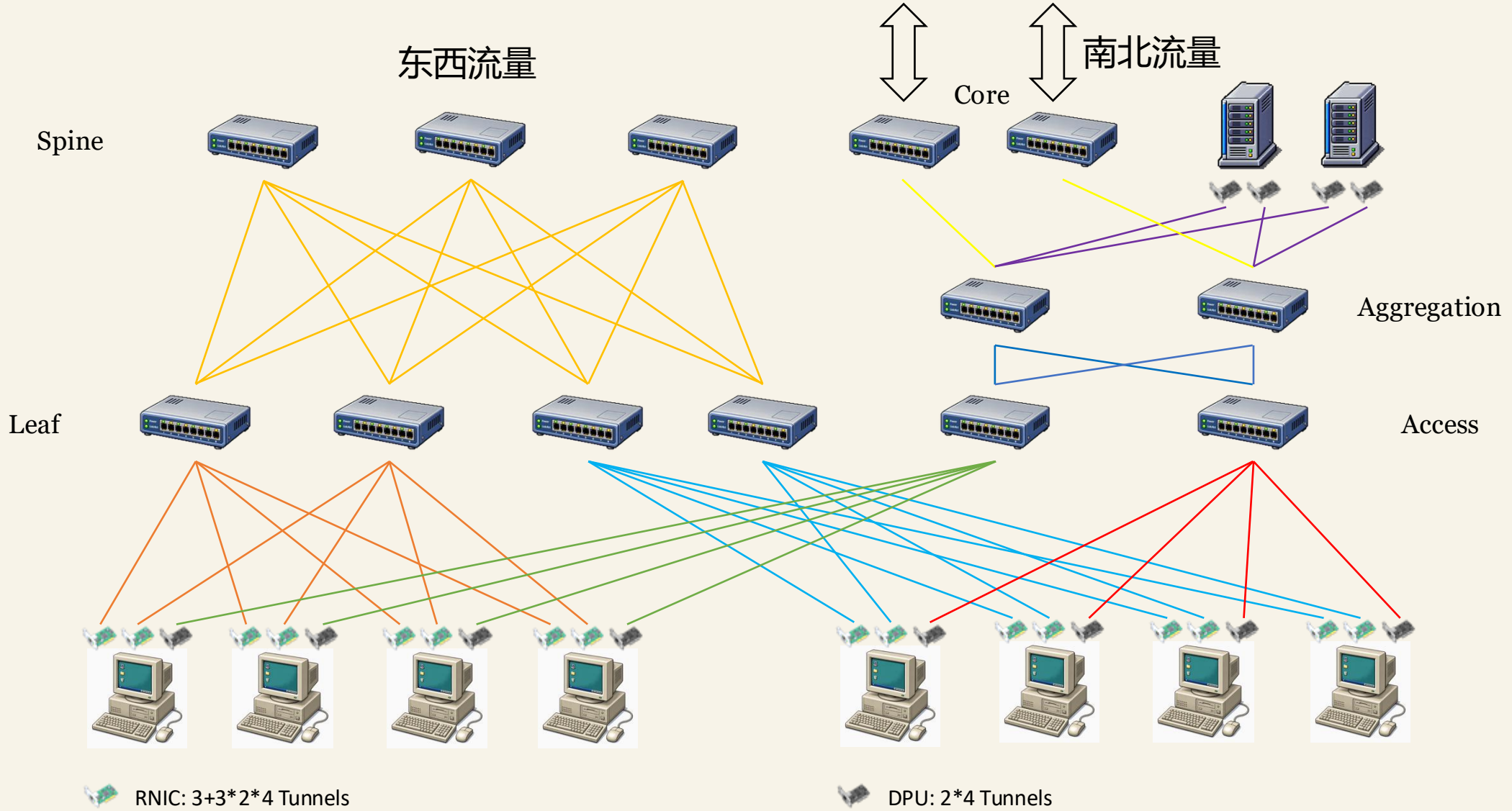
Network Hierarchy Design

层次	功能
语义层	任务语义兼容和实现的数据面
连接层	管理任务、生命周期、资源和授权的控制面
执行层	为 Message、Tile 以及聚合分散等任务产生有界事务和地址流
事务层	事务的准入、映射、执行(多路径/重路由等)和退休
隧道层	单 Tunnel 的可靠传递与拥塞控制
路径层	分组的转发与路由

Network Architecture



Network Architecture



PART III

III

Beyond the Monolith: Modern Network with Tile-based Computing

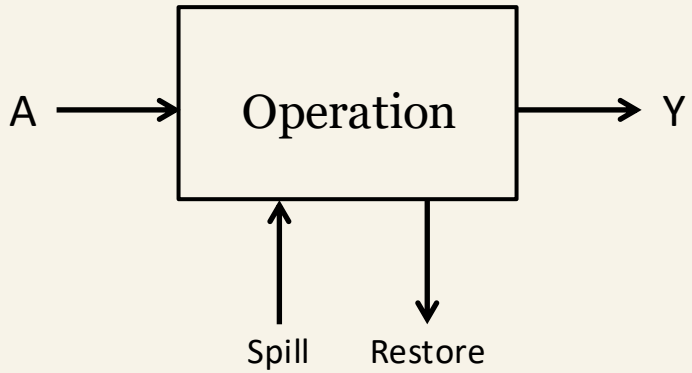
Stateful Operation is inevitable

过去，算子是无状态的：RISC 与 乱序用时间换性能

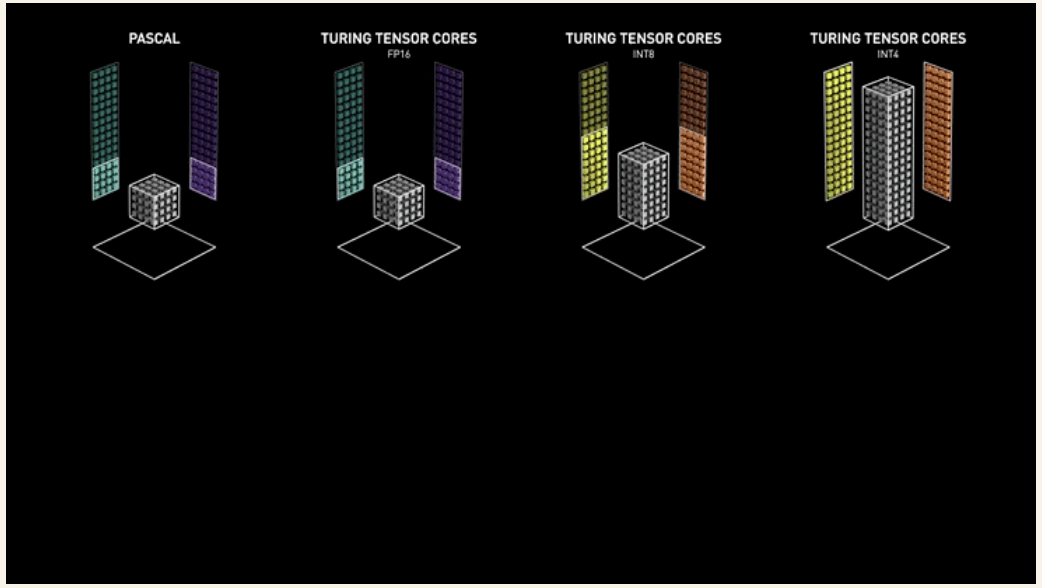
现在，算子是有状态的：能量账逼迫更多的数据复用

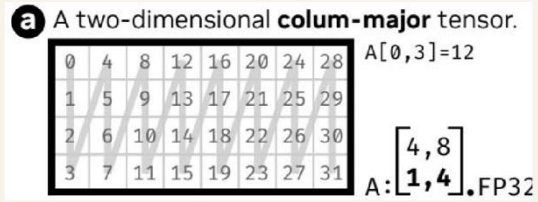
P.46/47：专用胜过通用，但自动化不会自动带来设计/调度方法学。

- ✓ 算子：一个有输入，输出，以及状态的 Spill/Restore 四个数据通路的硬件计算单元。
- ✓ 处理器：调度该种算子的系统，包括调度控制通路，片上存储单元与内存网络接口。

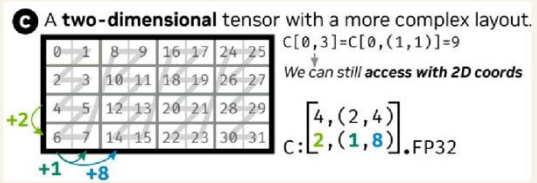


Tile Descriptor



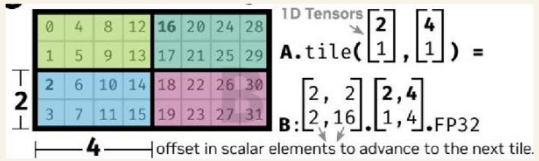


（a）4×8 列主序张量：stride $(1, 4)$

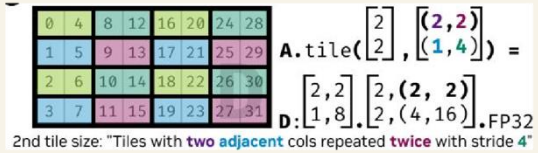


（b）第二维为 hierarchical dim $(2, 4)$ 的复杂布局

图 14: Graphene 中的张量内存布局抽象（与 CuTe `Layout<Shape, Stride>`在数学上同构）。(a) 展示简单的列主序布局；(b) 展示通过多级 stride 描述的层次化布局，对应 CuTe 中 `Layout<(4,(2,4)), (8,(1,4))>`的表达能力。(来源：Hagedorn et al., PLDI 2023, Fig. 3)



（a）切分为 2×2 个 2×4 连续 tile



（b）两维均非连续的层次化 tile

图 15: Graphene 中的 Tiling 抽象（与 CuTe `Layout::tile()` / `TiledMMA` / `TiledCopy` 同构）。(a) 展示常规的连续 tile 划分；(b) 展示通过非 unit stride 实现的复杂 tile 映射，这正是 CuTe 处理 `ldmatrix` 等底层 PTX 数据移动时所依赖的核心能力。(来源：Hagedorn et al., PLDI 2023, Fig. 4)

The Future Belongs to Tile

计算世界的粒度正在变化：

- 硬件按 Tile 算：CPU 矩阵扩展（Intel AMX / ARM SME / ACE）、GPU Tensor Core
- 软件按 Tile 写：Tile 编程范式（TileLang / Tilus / CUDA Tile……）。Shape / Stride 本质是给 AGU 的描述符。

对象变了: Tile 的传输会成为网络流量的主力：

- Tile 是算子输入/输出/状态搬迁的单位；它不该被迫先写 DRAM、再被读回。DRAM 中转付掉的是带宽、能量和一次 RTT。

网络必须原生承载"有界的大块事务"：Tile 是一等传输对象，不是被切碎后就消失的大消息。

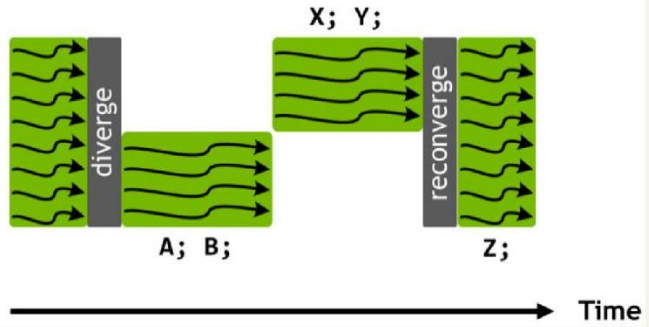
CPU owns setup. xPU owns issue.

接口变了:

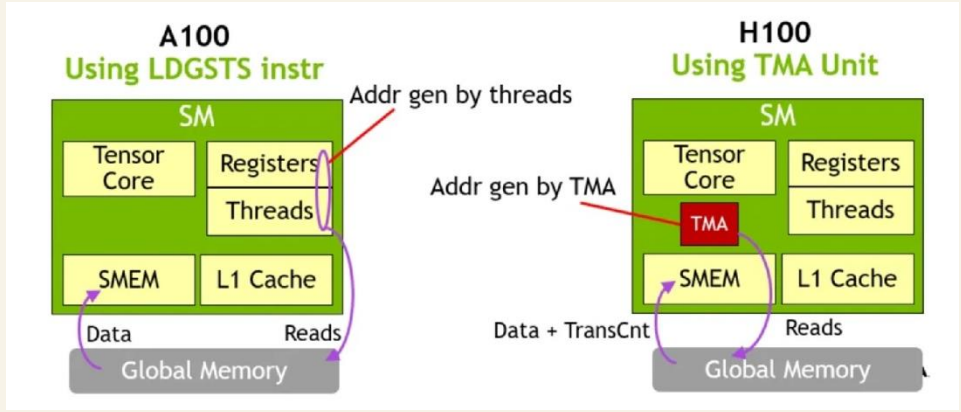
- 计算是 Tile，数据是 Tile，同步是标量控制与 Atomic；通信只想说：Tile Load / Tile Store。
- 连接/映射/权限由 CPU 一次装好；之后数据面不再过 CPU。DMA 是动词，不是名词。
- SQ/CQ 把 DMA 名词化成队列设备：SQE / doorbell / CQE / 水位 / 完成序，让每次通信先经内存办控制面手续；把软件队列 ABI 当成所有 xPU 的公共契约，成本太昂贵。
- 带宽把这笔税放大：队列管理设计于"目标很慢"的时代——目标越快，按次簿记的税越明显。NVMe 之于 μs 级闪存，正如 verbs 之于 μs 级网络；io_uring 优化了队列，却消灭不了队列编程。
- 新交互很小：Tile / Descriptor / Load-Store / Atomic / Barrier-Commit-Wait；GPU 已证明这是程序员日常。**地址空**间是终极奥义。

GPU Already Voted: Explicit Sync + Async Engines

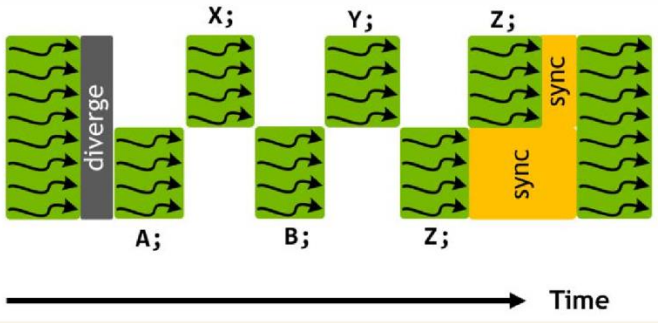
```
if (threadIdx.x < 4) {
    A;
    B;
} else {
    X;
    Y;
}
Z;
```



(a) Pascal 及早期 GPU 的 SIMT Warp 执行模型：分支发散时不同路径被串行化执行（A,B → X,Y），限制了线程间细粒度协作。

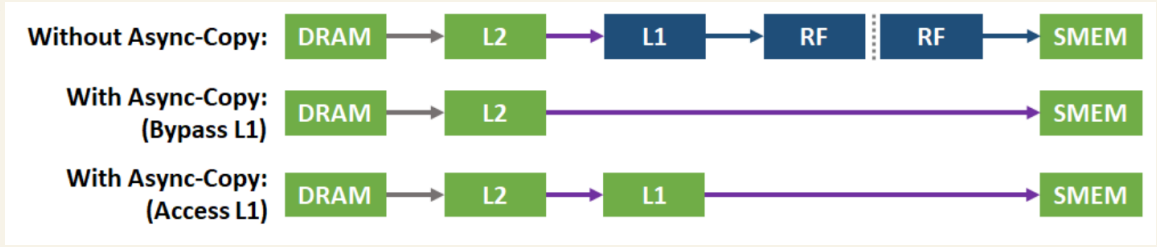



```
if (threadIdx.x < 4) {
    A;
    B;
} else {
    X;
    Y;
}
Z;
__syncwarp()
```



(b) Volta 独立线程调度与显式同步：为每个线程维护独立 PC 和调用栈，支持 `__syncwarp()` 显式同步，允许子 warp 粒度的线程同步与数据交换。

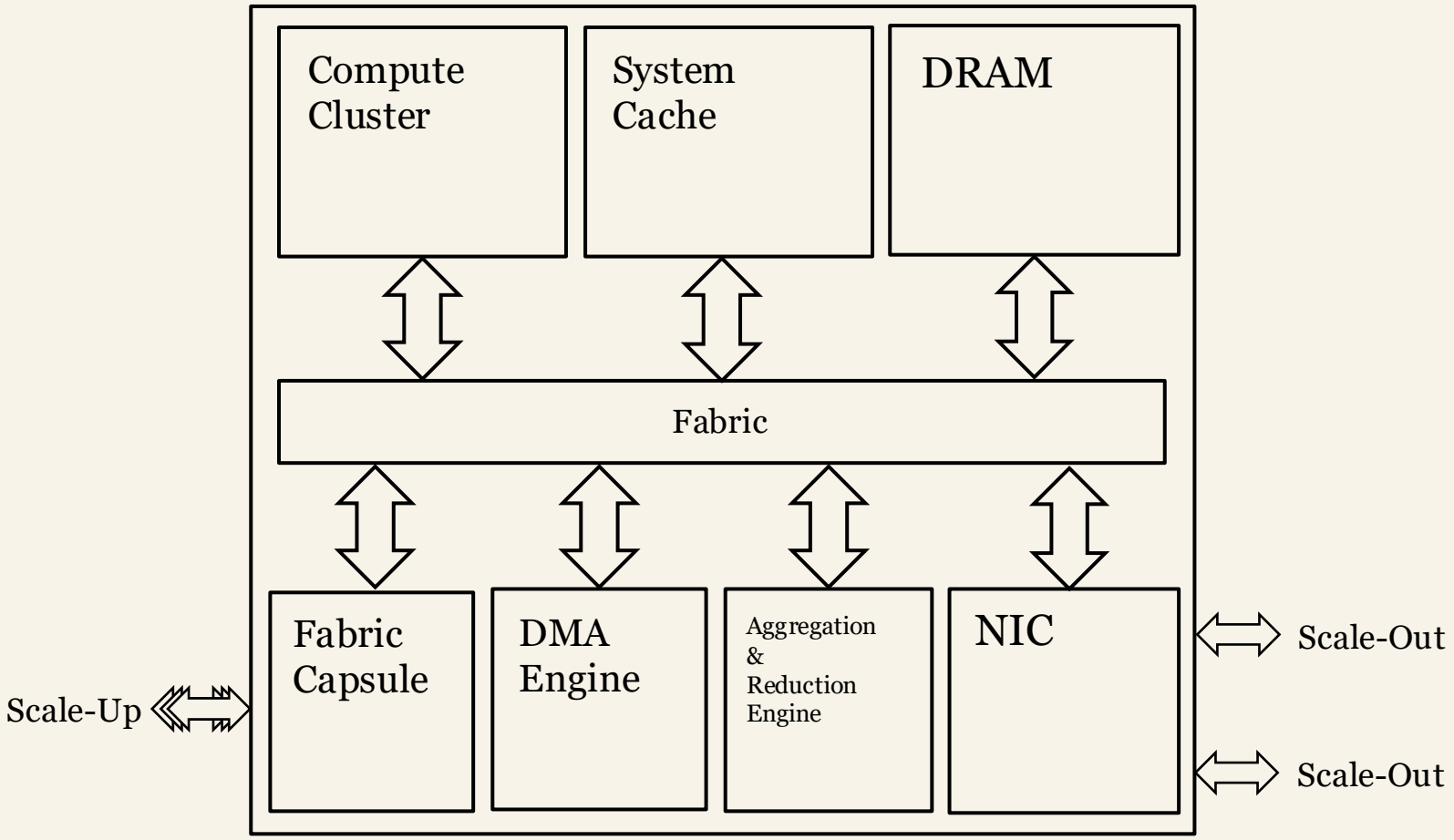
图 7: SIMT 分支调度机制对比。(a) Pre-Volta 架构使用 SIMT Stack 串行化执行发散路径；(b) Post-Volta 架构支持独立线程调度和显式汇合。这种演进使 Volta 能够支持细粒度锁和无饥饿算法。(来源：NVIDIA V100 Whitepaper, Figs. 20 & 23)



Asynchrony of MMA Instructions			
Architecture	**MMA Type**	**MMA Instruction**	**SASS**
Volta	Warp-scoped	`mma`	`HMMA`
Ampere	Warp-level synchronous	`mma`	`HMMA`
Hopper	Warpgroup-level asynchronous	`mwgmma.mma_async`	`HGMMA` `QGMMA`

Blackwell	Asynchronous	`tcgen05.mma`	`UTCHMMA`
			`UTCQMMA`
			`UTCOMMA`

Unified System



Questions & Discussion