Falcon: A Reliable and Low Latency Hardware Transport

Nandita Dukkipati, Neelesh Bansod, Chen Zhao (Google)
Yadong Li, Jay Bhat, Shiraz Saleem, Anjali Singhai Jain (Intel)
Netdev Conference, Santa Clara
15th July, 2024

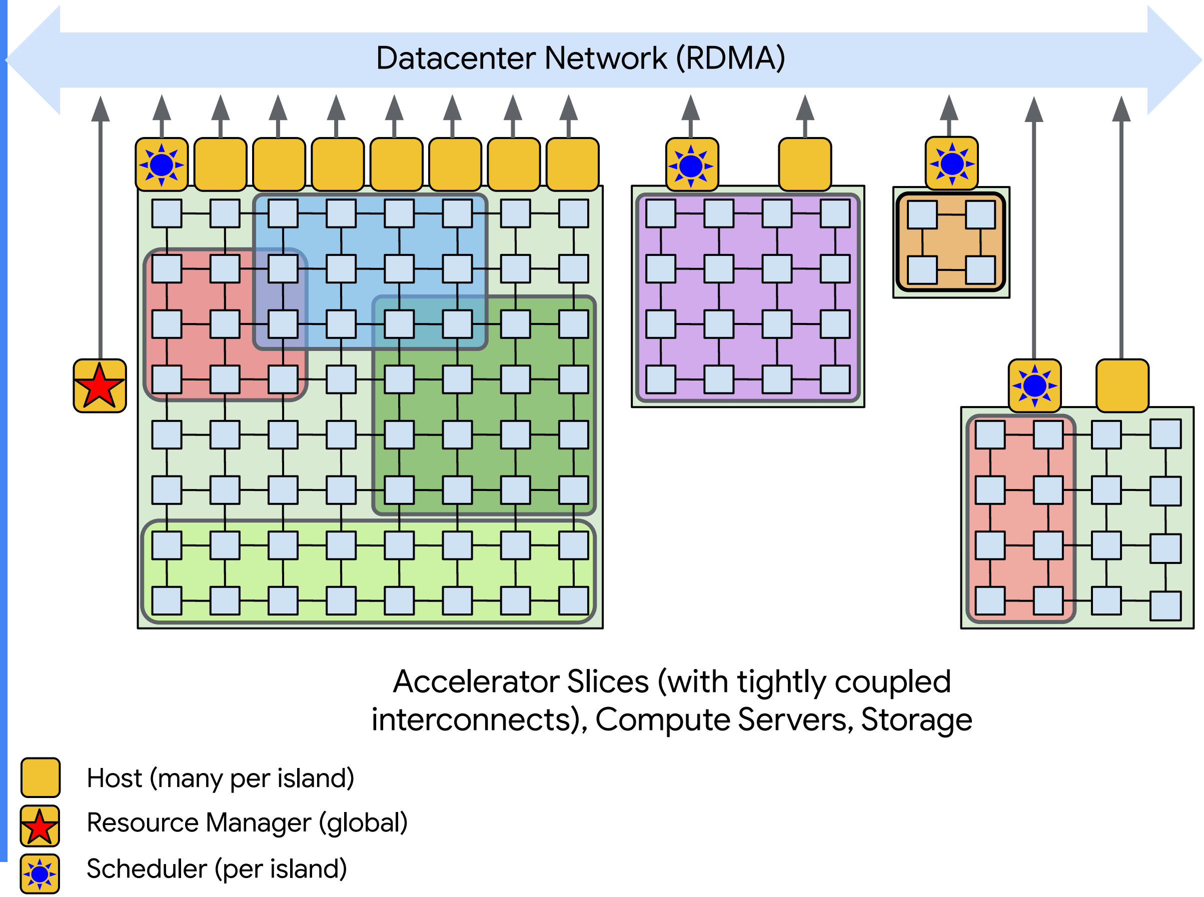
FALCON

This Talk:

- **Falcon: Motivation, Overview and Performance**
- RDMA/Falcon SW Components
- Relationship to netdev

Falcon: Motivation, Overview and Performance

Rethinking Transport in NICs - Why?



New and demanding workloads - high burst bandwidth, high Op rate, low latency.

Satisfy demands of new workloads (massive scale AI/ML training, High Performance Computing, Real-time Analytics) and existing ones (Storage, RPCs).

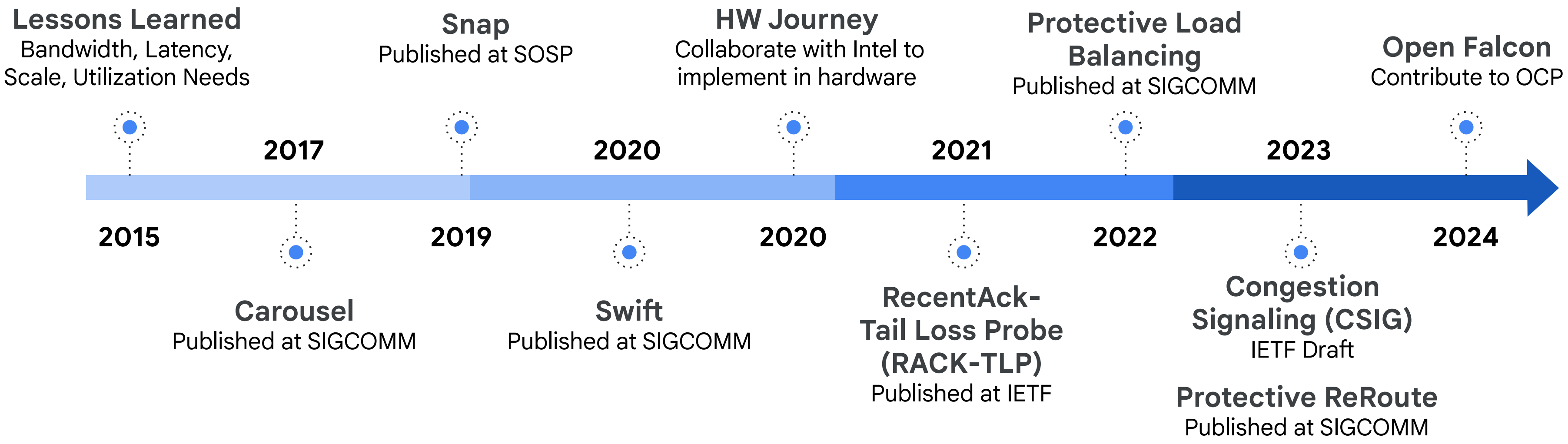
Incremental gains of software stack optimizations.

Need order of magnitude improvements over highly optimized Software Transports.

Modernising Ethernet for low latency and high bandwidth.

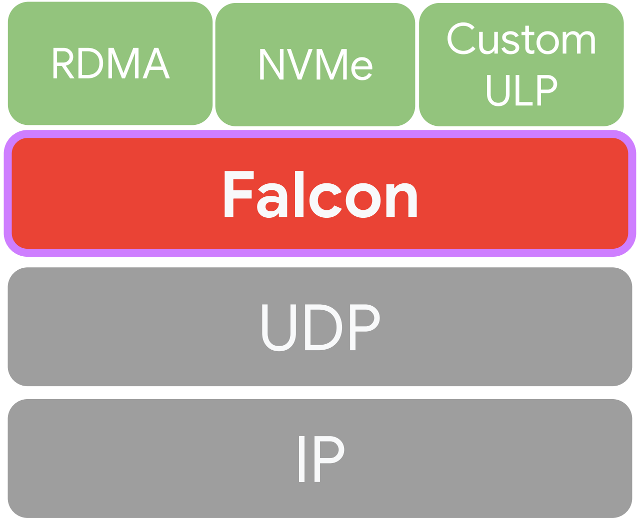
Deployment Experiences at scale give us a glimpse of the possibilities.

Falcon technology: A reliable, low latency hardware transport for the ecosystem to advance modern hyperscaler infrastructure



Bringing 10 years of advances in low latency, isolation and efficiency to hardware

Falcon: Multi-protocol Reliable Transport



Tail Latency in Ethernet networks

- ➔ HW assisted **delay-based Congestion Control**
- ➔ **Selective ACKs** for fast loss recovery
- ➔ **Multipath** capable connections
- ➔ Bundled under **Programmable Engine**

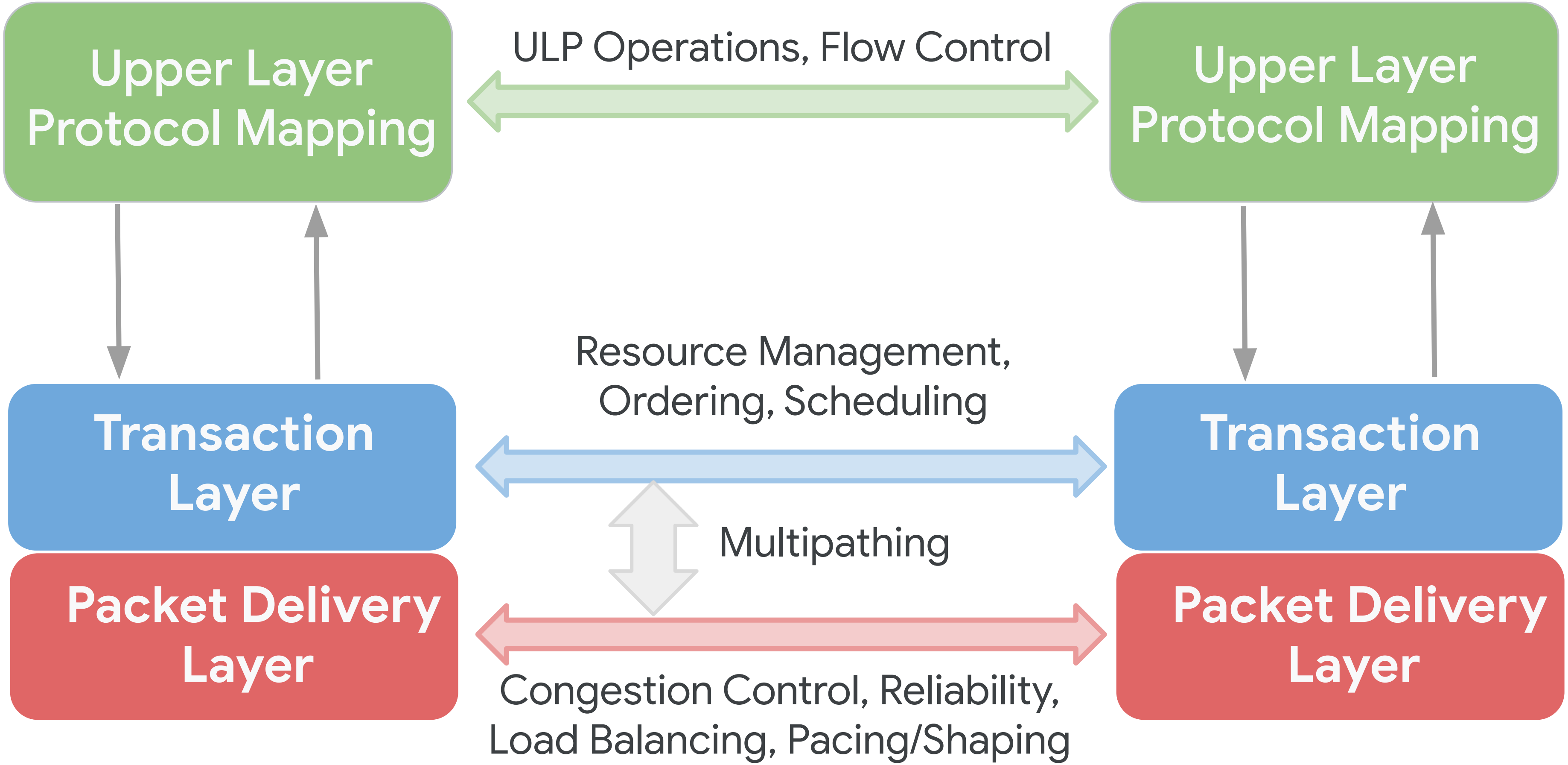
Isolation and Visibility at scale

- ➔ µs-granularity per-flow **Traffic Shaping**
- ➔ Fine-grained Stats for **Debuggability, Software Defined Network control**

Efficiency and Security

- ➔ **Implemented in HW** for Low Latency, High Op Rate using Industry-standard Interfaces, and [PSP](PSP) encryption

Falcon Layers



Mapping Upper Layer Protocols (ULP) to Falcon

Industry Standard Interface

- RDMA InfiniBand Verbs Compatible ULP
- Supports Reliable Connected (RC) and Unreliable Datagram (UD) Queue Pair types.
- Strictly Ordered: in-order data placement, in-order completions.
- On-NIC reorder buffer to support OOO delivery from the network.

Enhanced Interface

- Introduces Relaxed Ordering Modes: weakly and unordered.
- Graceful Error Handling with Complete-in-Error and continue (CIE): signals errors to applications without tearing down the connection.

Industry standard interfaces are extended in support of warehouse-scale applications.

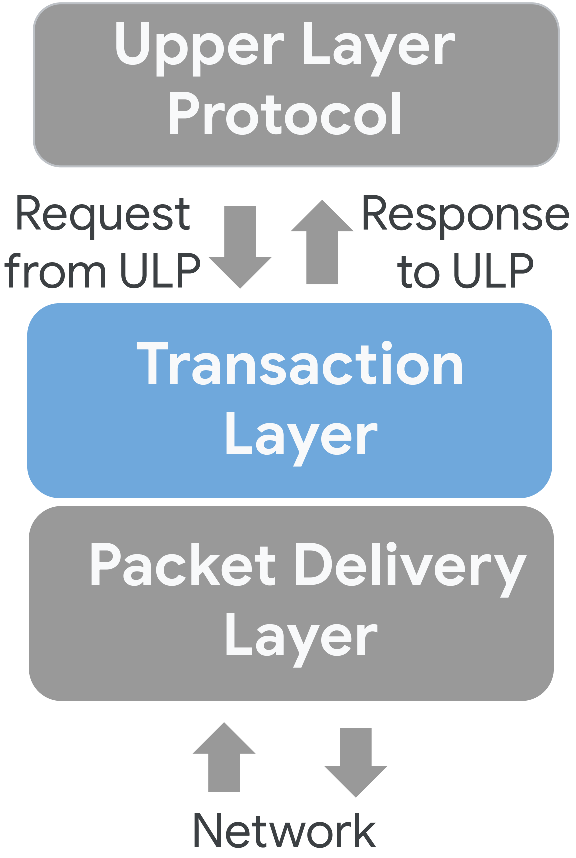
Falcon Transaction Layer

Exposes request/response interface to ULPs.
IB Verbs Reads, Atomics, Writes, Sends are mapped to request-response transactions.

Orders Transactions due to out-of-order network arrivals.
Ensures ordering semantics expected by the ULP.

Schedules transactions on the wire per QoS-policies.

Manages finite Falcon resources for isolation and deadlock prevention.

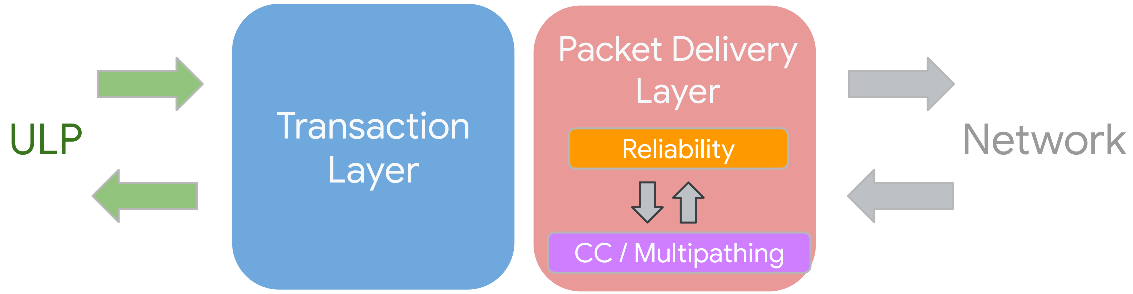


Falcon Packet Delivery Layer

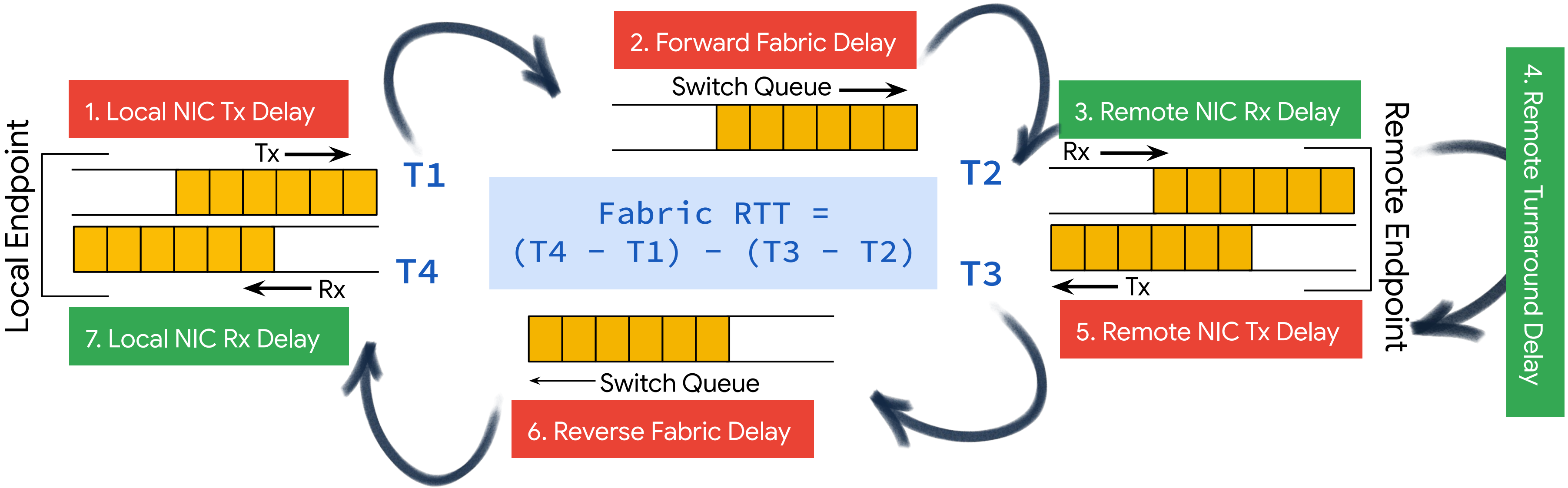
Falcon Packet Delivery Layer between Transaction Layer and the Network.

Performs the more canonical responsibilities of a typical transport -

- Ensures end-to-end reliable delivery from transmitter to receiver.
- Does congestion control and multipath network load balancing to ensure low-latency and high utilization.

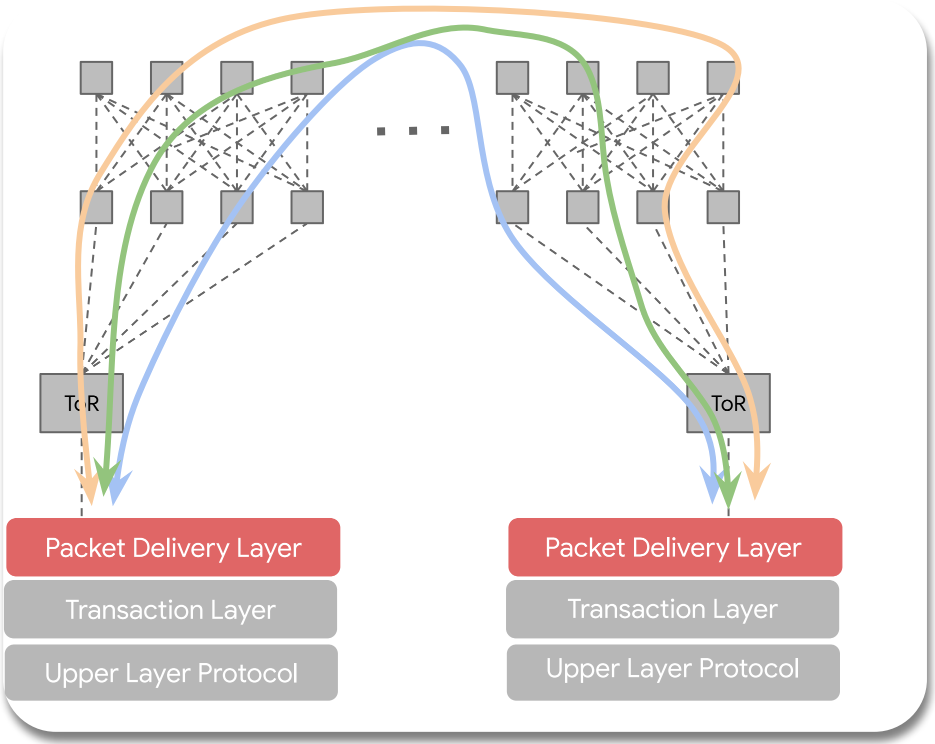


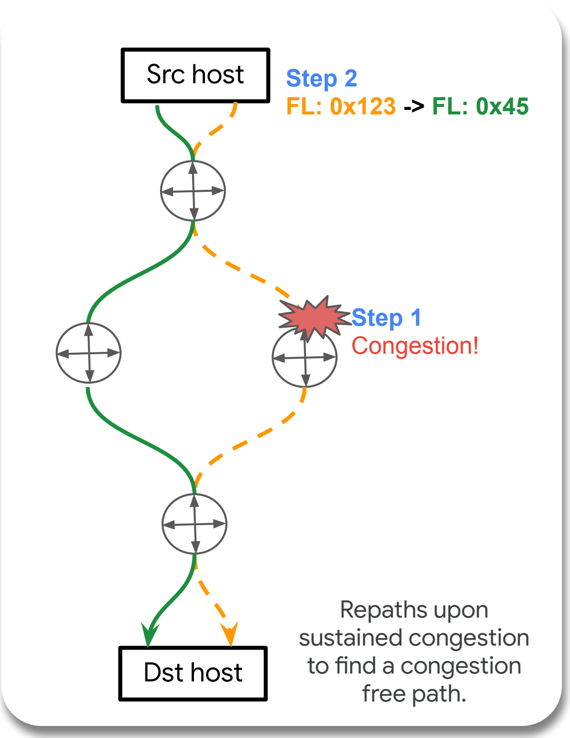
Swift Congestion Control as Baseline



Swift is a delay based congestion-control for Datacenters that achieves low-latency, high-utilization, near-zero loss implemented completely at end-hosts and NICs supporting diverse workloads like large-scale incast across latency-sensitive, bursty and IOPS-intensive applications working seamlessly in heterogeneous datacenters.

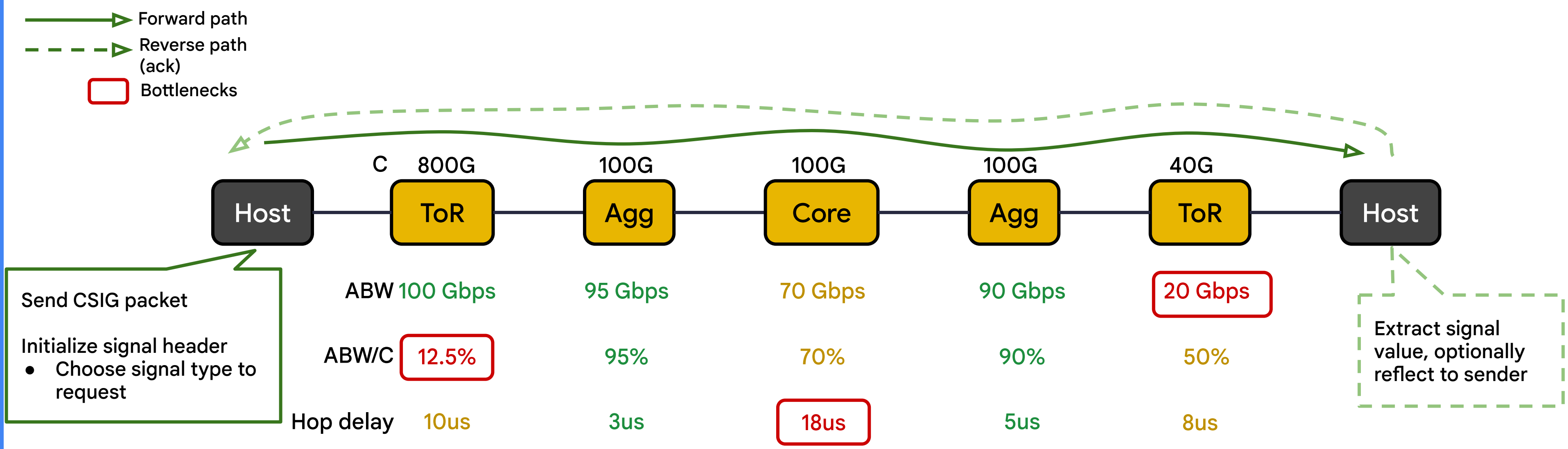
Congestion Aware Multipath Network Load Balancing





RDMA/Falcon applications leverage multiple paths for load balancing in the network fabric transparently.

Congestion Signaling (CSIG): Practical & Effective In-band Signaling Protocol



Minimum Available Bandwidth: min(ABW)	Minimum Available Bandwidth in bps across all links on the packet path.
Maximum Link Utilization: max(U/C) or min(ABW/C)	Maximum Link Utilization in percentage of link speed across all links on the path.
Maximum Per-Hop Delay: max(PD)	Maximum per-hop delay across all hops in the path.

Timely and Precise Loss Recovery via Selective Retransmissions

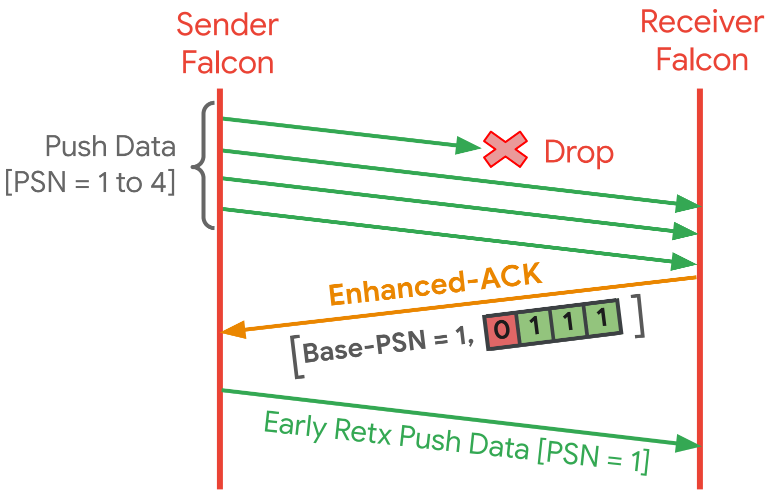
Goal: Ideally retransmit -- only once -- the lost packets in a timely manner.

Falcon Receiver

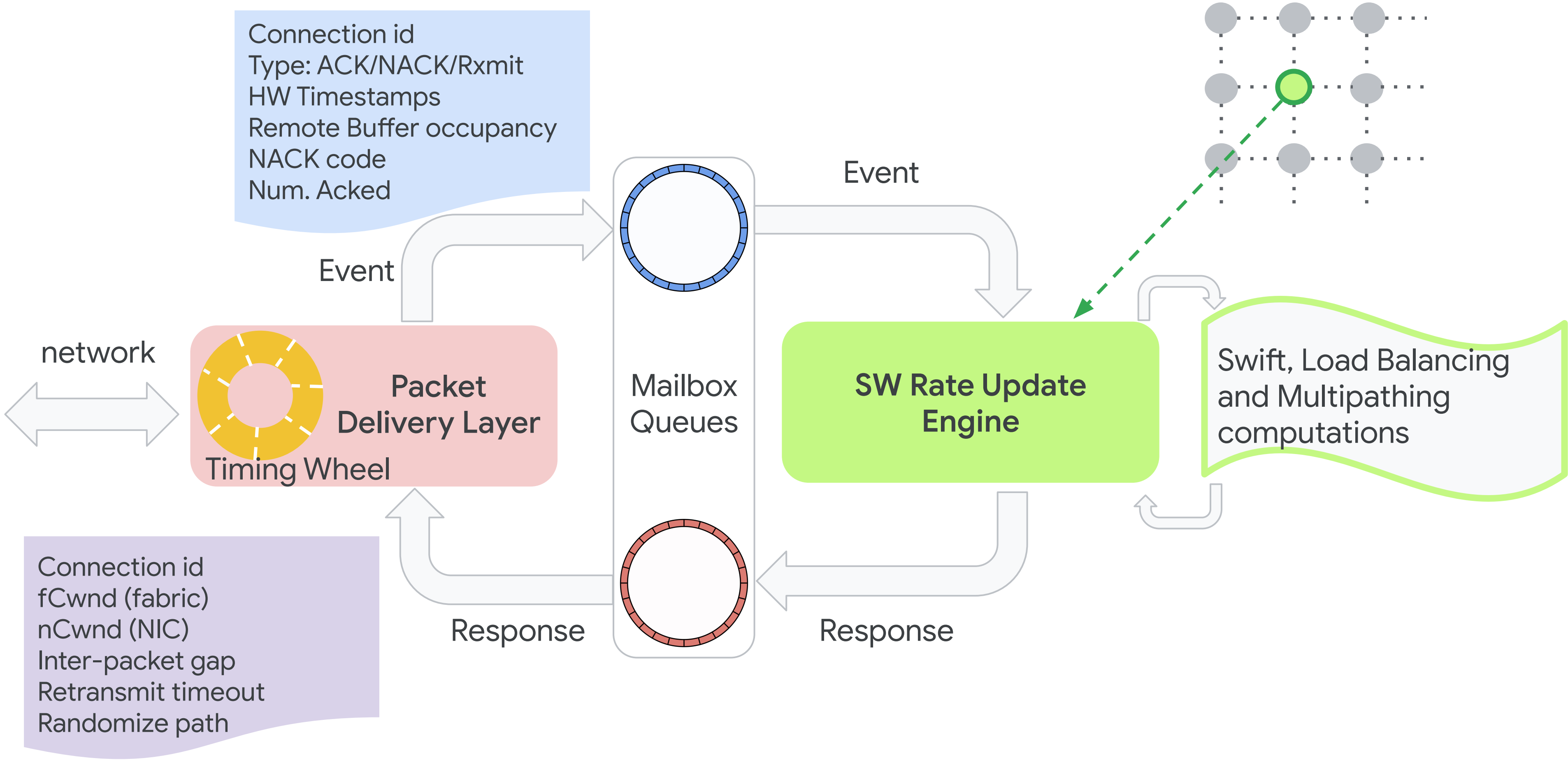
- Indicates to the sender which packets are received.
- Acknowledgement coalescing and piggybacking for high Op rate.

Falcon Sender

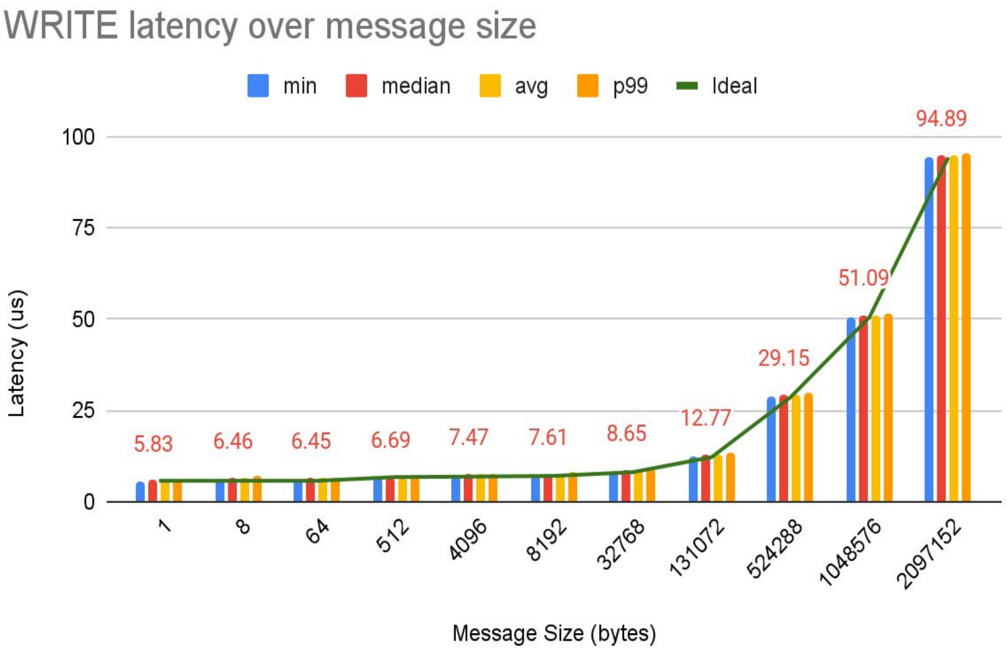
- Leverages relayed information to retransmit lost packets in a timely manner.
- Hardware-based retransmission - no firmware.
- Recent Acknowledgement (RACK) and Tail Loss Probe (TLP) can further enhance loss recovery.



Programmability in Rate Update Engine (RUE)



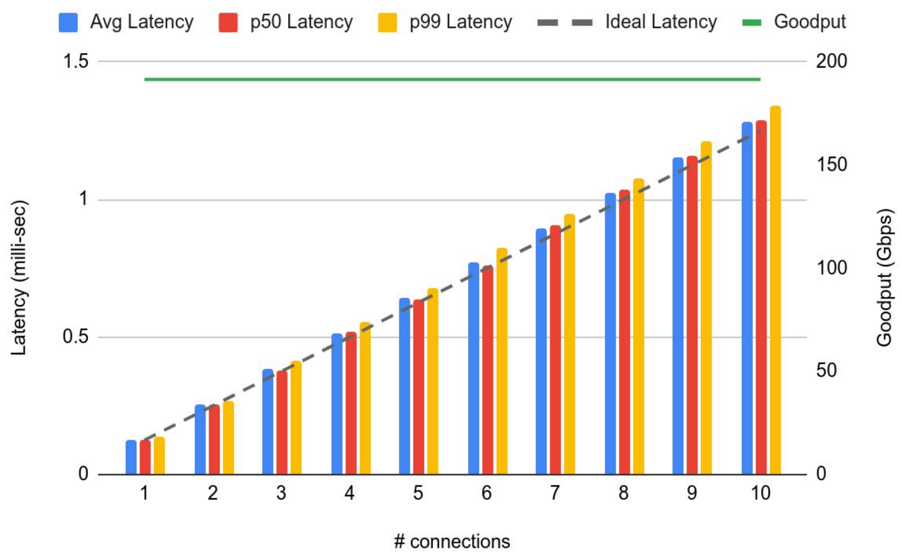
Single Queue Pair Latency

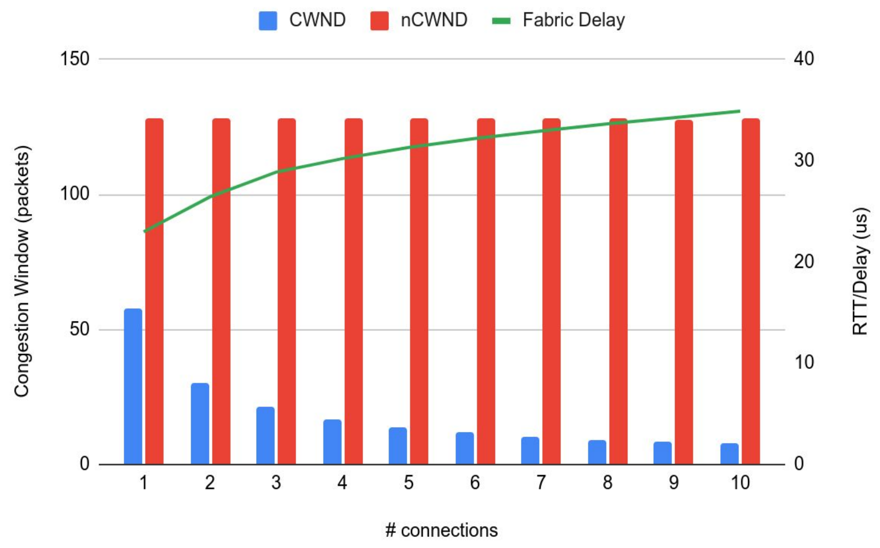


Setup:
Measure message completion time (round-trip time from application posting Op to receiving completion) over message sizes.

Takeaway:
Tight tail latency: p99 latency, median and ideal latency match across message sizes.

Incast with increasing #connections

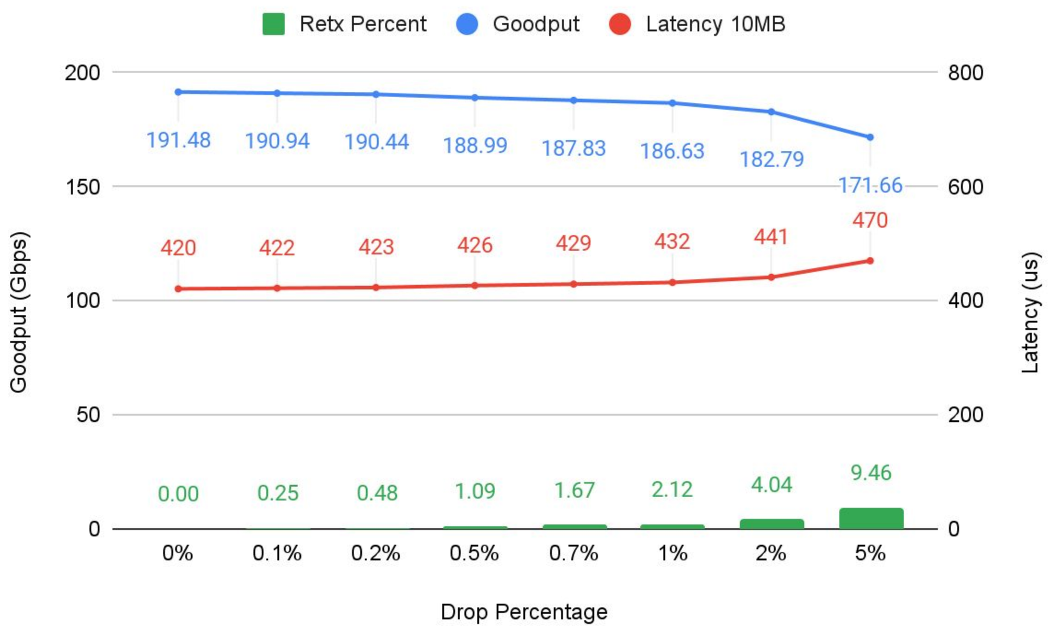




Goodput is saturated at max link speed. Median and Tail latencies are close to the ideal achievable.

Fabric congestion window modulates per #connections. Round-trip time settles at Swift target delay.

Scalability under Packet Losses



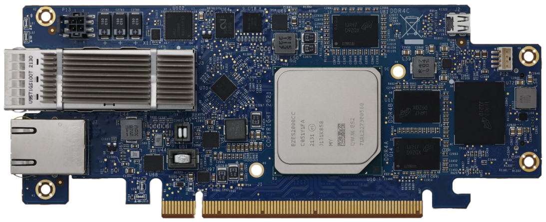
Setup: induced packet losses on one Queue Pair from 0 to 5%.

Takeaways:

Stable goodput and message latency as loss rate increases; graceful degradation at higher losses.

Low retransmission overhead even under high packet loss rate

Why Falcon Matters



Predictable performance for warehouse-scale: low tail latency, massive application bandwidth, mitigating congestion and efficient network utilization.

Efficiency @scale: HW acceleration enables high-bandwidth (200 Gbps), low-latency (~2.0µs one-way) and high Op rate (150Mpps) and connection scaling.

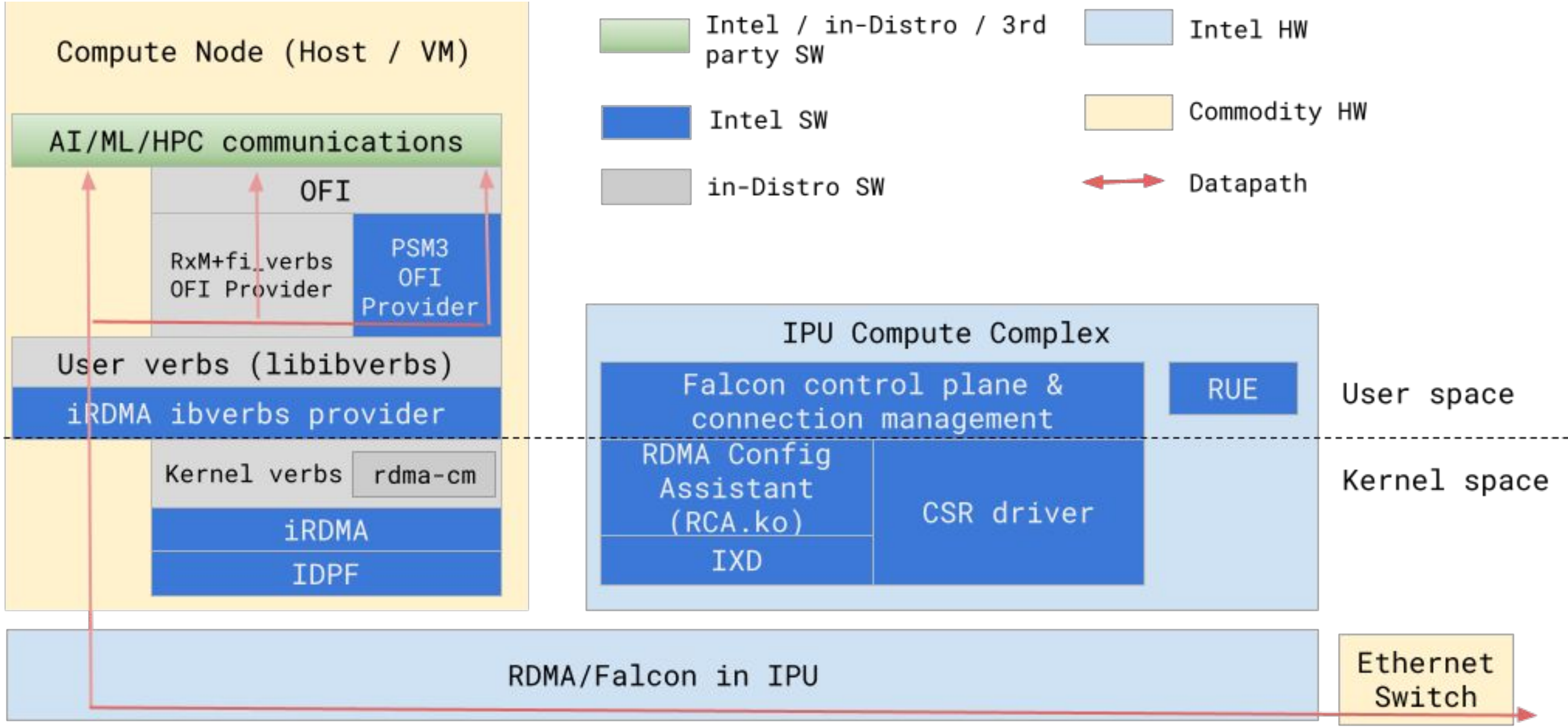
Need of the day: meets requirements of critical workloads, HPC and AI/ML; also good for offloading Storage and RPC.

This Talk:

- Falcon: Motivation, Overview and Performance
- **RDMA/Falcon SW Components**
- Relationship to netdev

RDMA/Falcon SW Components

SW Components



Application Interface

- Falcon supports Standard IB Verbs Interface.
    - Applications work w/o modifications: Userspace verbs under libibverbs using RC (Reliable Connected) and UD (Unreliable Datagram) Queue Pairs.
    - Kernel verbs used for RDMA Connection Management.
    - Control plane is offloaded from the Compute Node to the IPU cores.
    - HW datapath offloaded under SR-IOV.

- Optional SW/HW features available over RDMA-Falcon
    - Optimized libfabric provider from Intel, PSM3.
    - Virtual traffic class for selecting performance profiles.
    - 8K MTU support.
    - Completions indicating dropped UD datagrams.
    - Unordered Queue Pairs with Out of Order completions and data placement for large operations .

Falcon Control Plane and Connection Management

Key functions

Manages resource policies for SR-IOV PCIE functions.

- E.g. Resource isolation for multi-tenancy, cap on HW resources consumed by a Virtual Function.

Connection setup done as a “bump-in-the-wire” model.

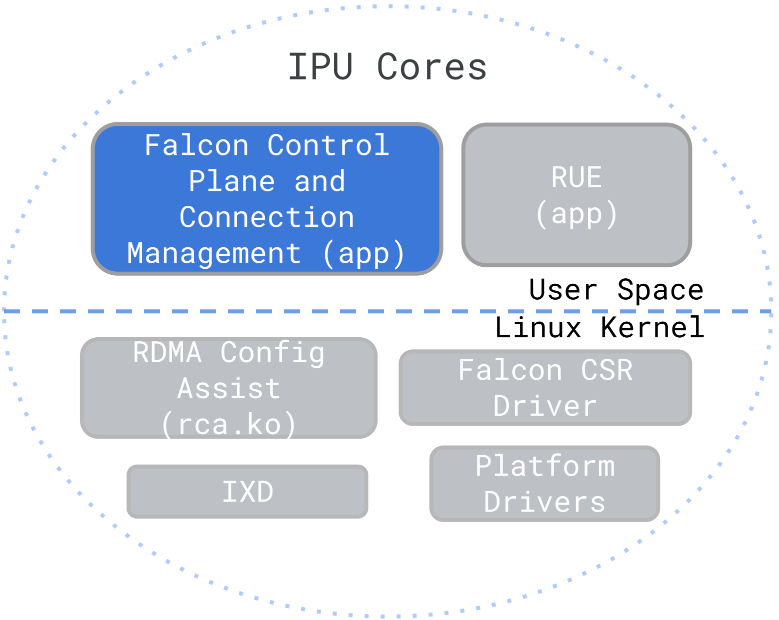
- Transparent to on-host RDMA drivers/applications.
- Optional business logic for connection management, e.g., Virtual Function (VF) lifecycle, VF - VNET mapping, VIP to PIP translation.

Manage the datapath performance.

- Telemetry for monitoring and troubleshooting.
- Configuration management for RDMA-Falcon.

Implementation philosophy

- Runs on IPU cores to enable Cloud Service Providers to operate RDMA.
- Host and VM RDMA SW stacks are Falcon agnostic to ease lift-and-shift applications.



RDMA over Falcon: Connection Setup Protocol

Guest VM / Host View:

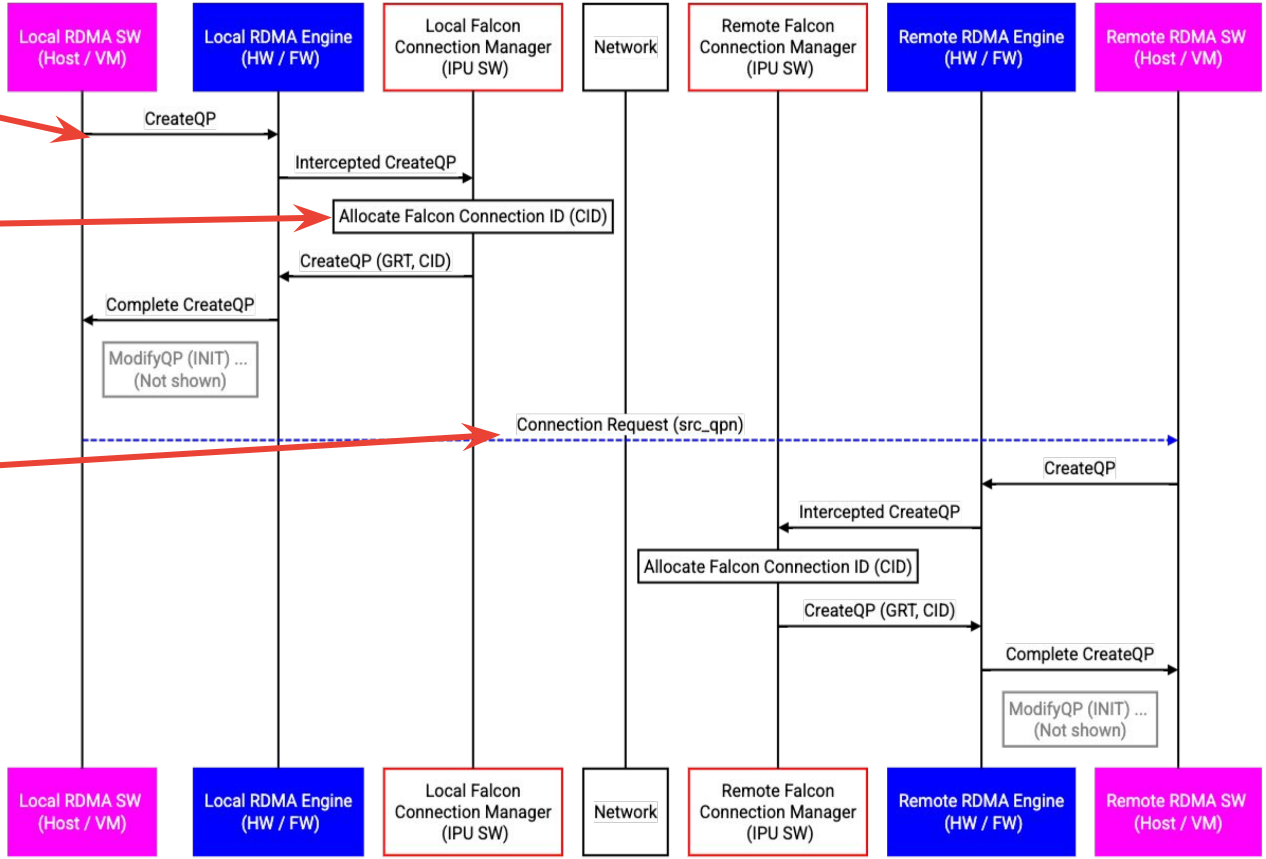
- Step 1: RDMA endpoint allocation, ibv_create_{pd, cq, mr, qp}.
- Step 2: Handshake with peer (rdma_cm or out-of-band).
- Step 3: QP setup with peer info, ibv_modify_qp.
- No changes to upstream (RoCE) software.
- Scalable control ops per second with large number of QPs.

Falcon View:

- Step 1: Intercept ibv_{create, modify, destroy}_{qp, ah}.
- Step 2: CID allocation, congestion control initialization, QoS controls, etc.
- Step 3: Security tunnel assignment (PSP, IP-SEC).
- Step 4: Handshake with peer (optional integration with VPC control plane).
- Step 5: Bring up connection and update RDMA QP/AH context with Falcon connection info.

Falcon RC Connection Setup (part 1: Resource Allocation)

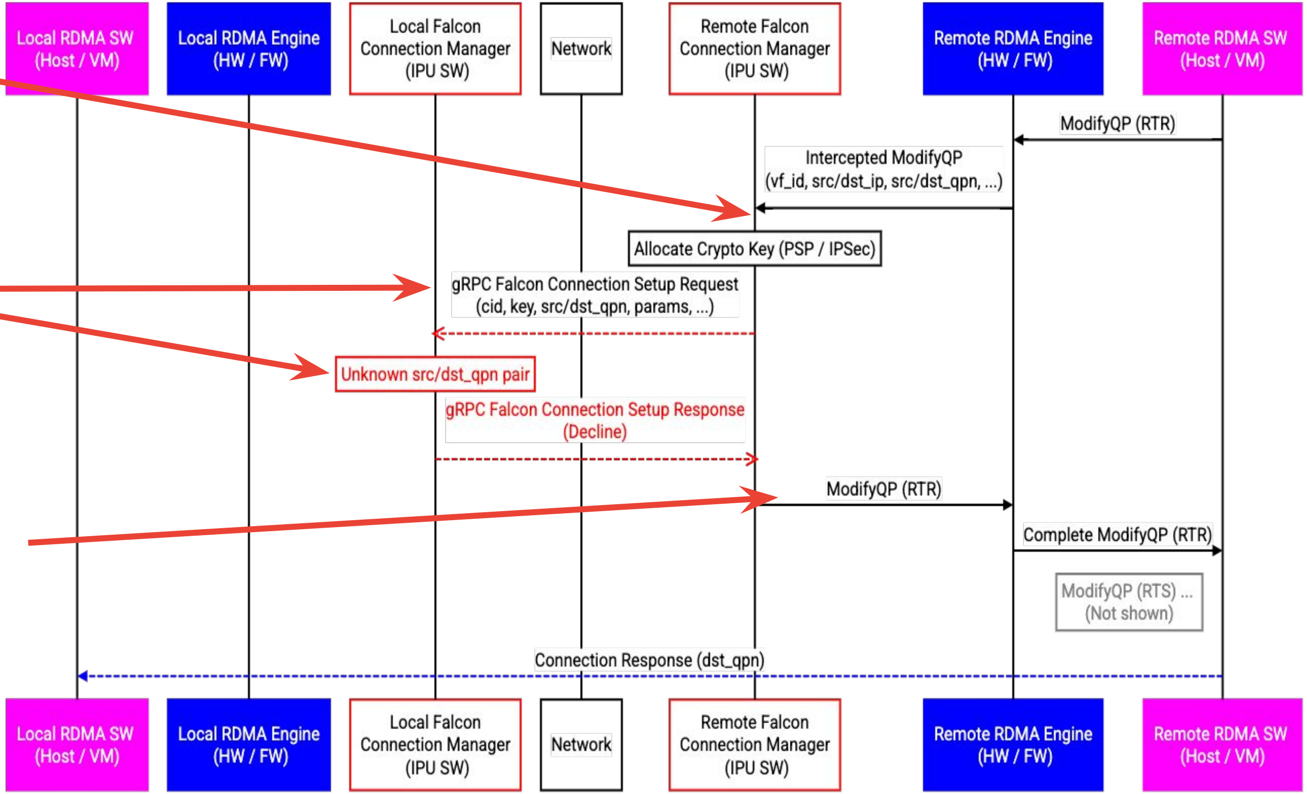
- QP commands are intercepted
- The Falcon CM allocates one CID per RC QP
- Similar flows work for RDMA-CM* as well as most OOB schemes



* A separate connection setup flow for CID - AH binding is necessary to support UD traffic in RDMA-CM

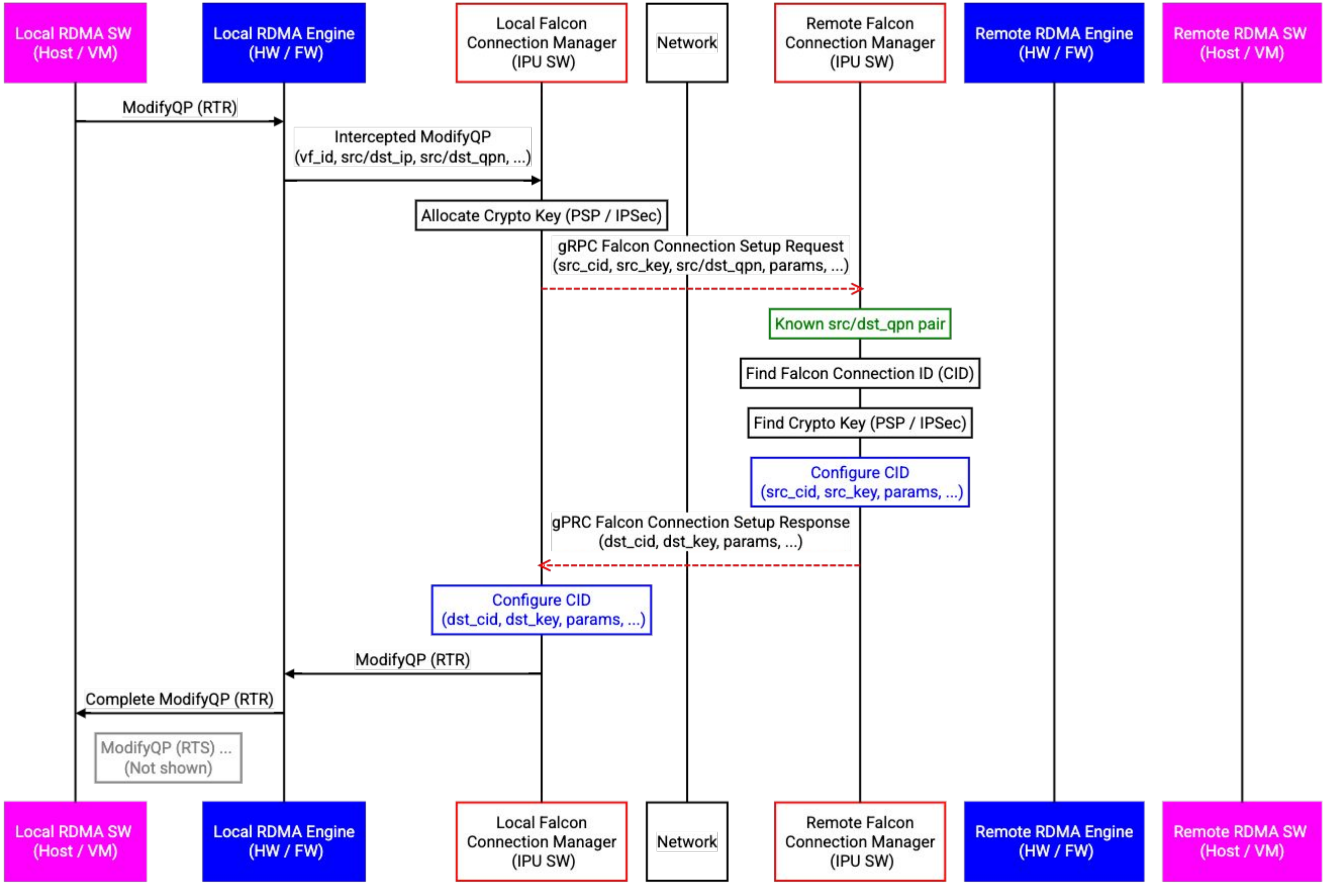
Falcon RC Connection Setup (part 2: Early Handshake)

- The Falcon CM initiates handshake when possible
- Optional: VPC / business logic integration
- ModifyQP (RTR) shall always be completed no matter if the Falcon connection setup is successful or not.



Falcon RC Connection Setup (part 3: Falcon Setup)

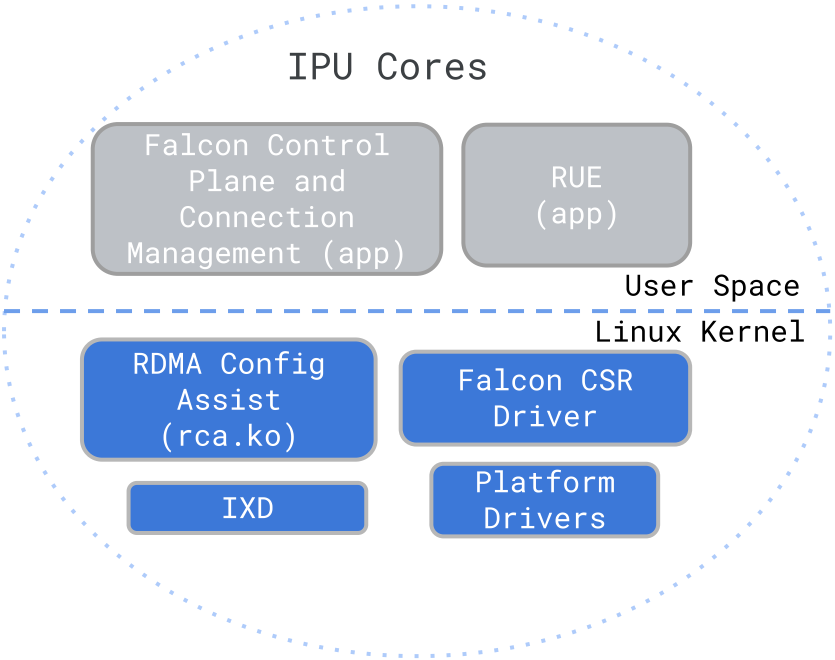
Parallel handshakes improve connection setup performance



Falcon CSR Driver and RCA.ko

Falcon CSR driver, RDMA Configuration Assistant (RCA) and IXD (Control Plane Driver for Intel IPU) are kernel modules.

- Falcon CSR driver provides memory bar and CSR register access to the Connection Manager and SW Rate Update Engine.
- RCA intercepts control path verbs such as QP/AH create/modify/destroy and forwards the commands to Falcon Connection Manager.
    - rca.ko is implemented as a IXD driver auxiliary device.
    - rca.ko takes ownership of a command queue for intercepting ibverbs from host RDMA driver.
    - rca.ko uses netlink for communications with the connection manager running in user space.



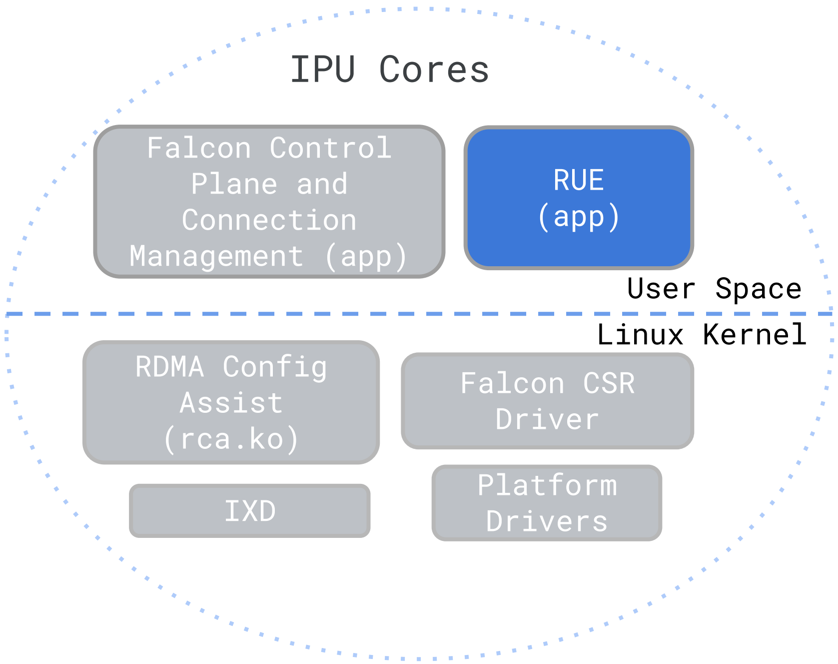
Rate Update Engine (RUE)

Programmable Congestion Control

- Implements Swift for fabric and NIC/host congestion.
- Implements congestion-aware Multipathing, and Load Balancing.
- Provides API for per-connection level congestion stats.

Implementation

- C++ Engine running on IPU Compute Complex.
- Can be upgraded in a hitless manner.
- Processes per-connection Datapath Events.
  - Generates congestion response for connection.
  - One response per RTT for each connection.
- Processes 18M events/sec on one core
  - Minimum DRAM interactions.
  - Batched event processing to reduce barriers.
- Can be scaled upto more cores if needed.



Telemetry (1/2)

Use-cases

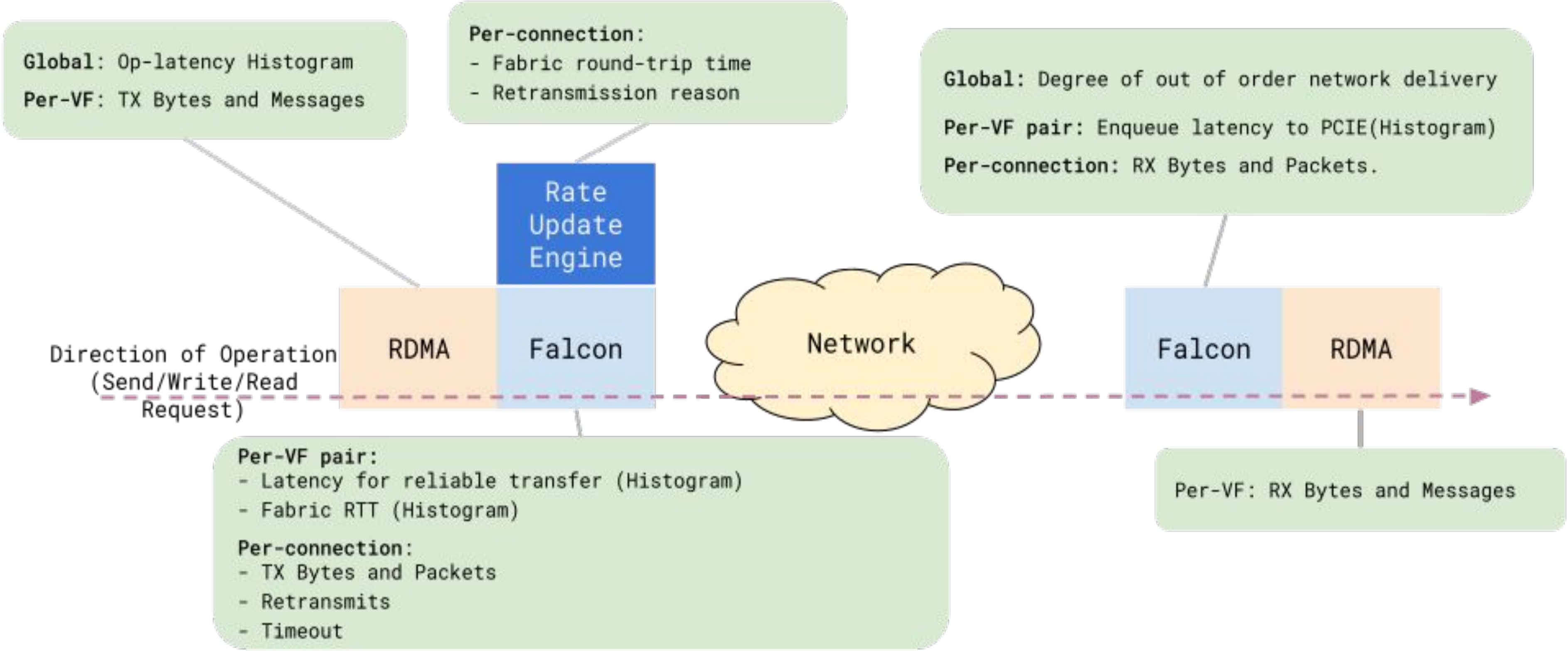
- Production fleet monitoring.
- Debugging network performance and availability.
  - E.g., when an application (on physical host/VM) reports that the network is slow.
- Surface telemetry to applications for monitoring and debuggability.
- E2E network performance tuning.

Our approach

- Latency histograms for visibility into tail latencies.
- Granular telemetry for precise debugging.
  - Per-connection statistics.
  - Per VM-pair statistics for virtualized applications.
- Telemetry collection has minimal impact on NIC performance.

Telemetry (2/2)

Telemetry collected across layers works coherently to identify bottlenecks in end-to-end paths.

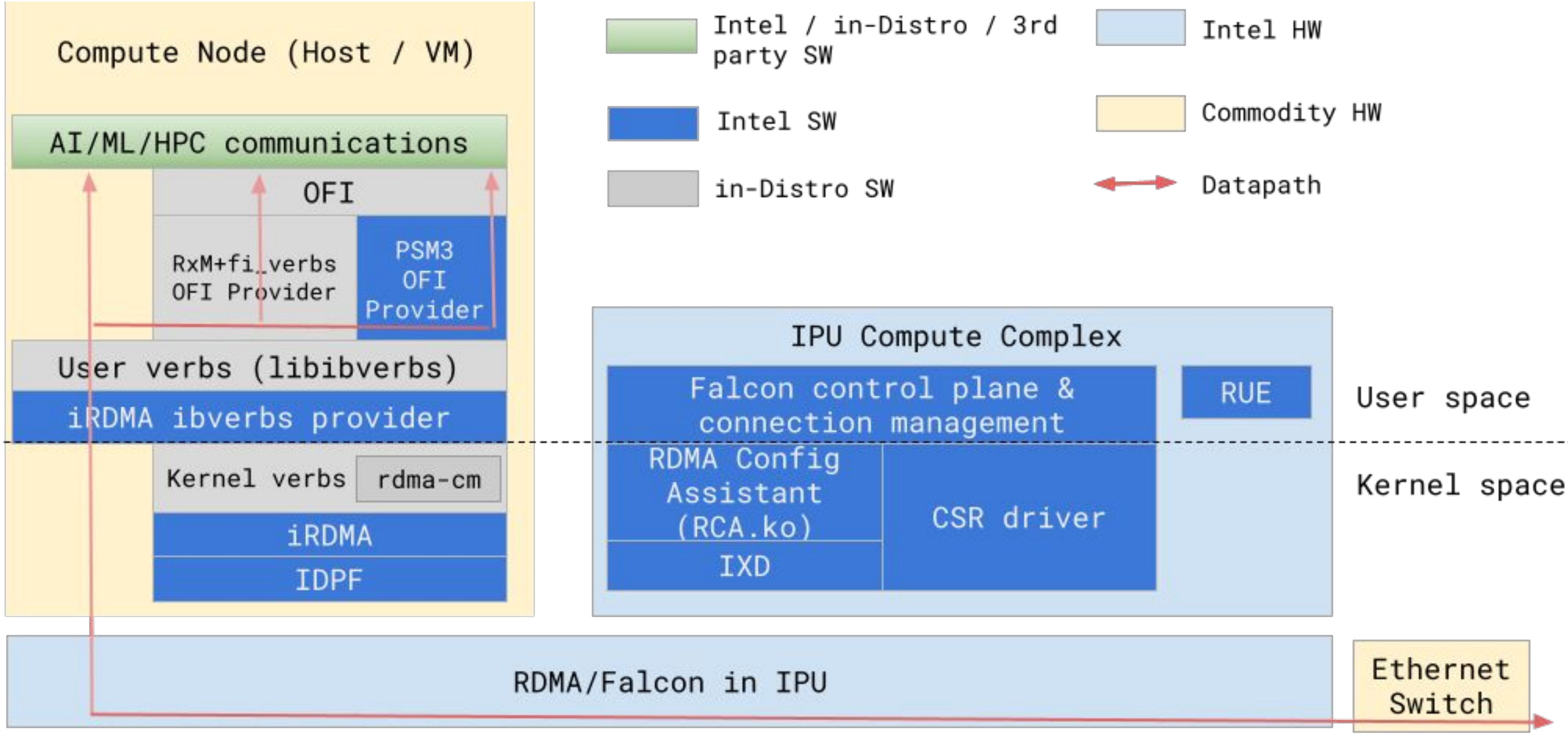


This Talk:

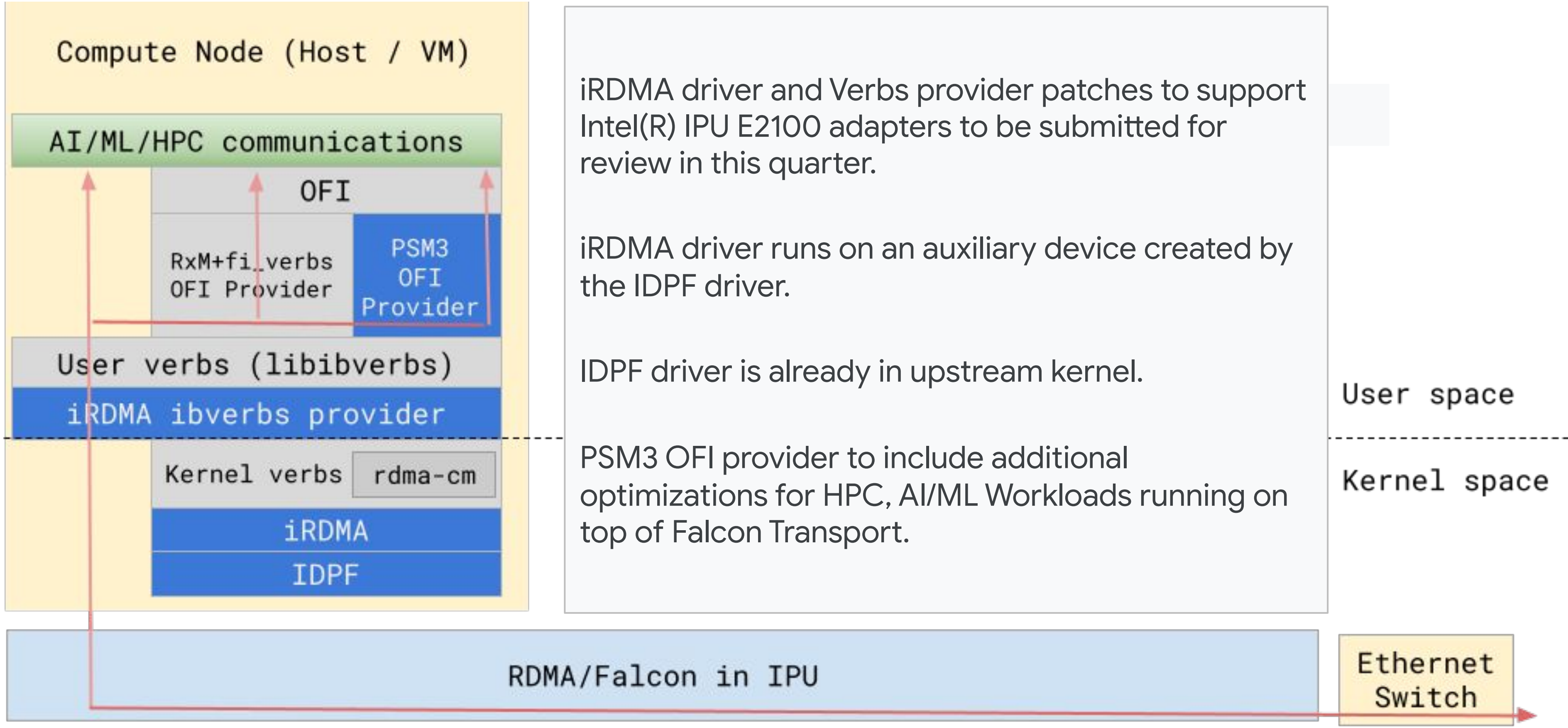
- Falcon: Motivation, Overview and Performance
- RDMA/Falcon SW Components
- **Relationship to netdev**

Relationship to Netdev

SW Components



Host Drivers and Providers - Upstreaming Status



iRDMA driver and Verbs provider patches to support Intel(R) IPU E2100 adapters to be submitted for review in this quarter.

iRDMA driver runs on an auxiliary device created by the IDPF driver.

IDPF driver is already in upstream kernel.

PSM3 OFI provider to include additional optimizations for HPC, AI/ML Workloads running on top of Falcon Transport.

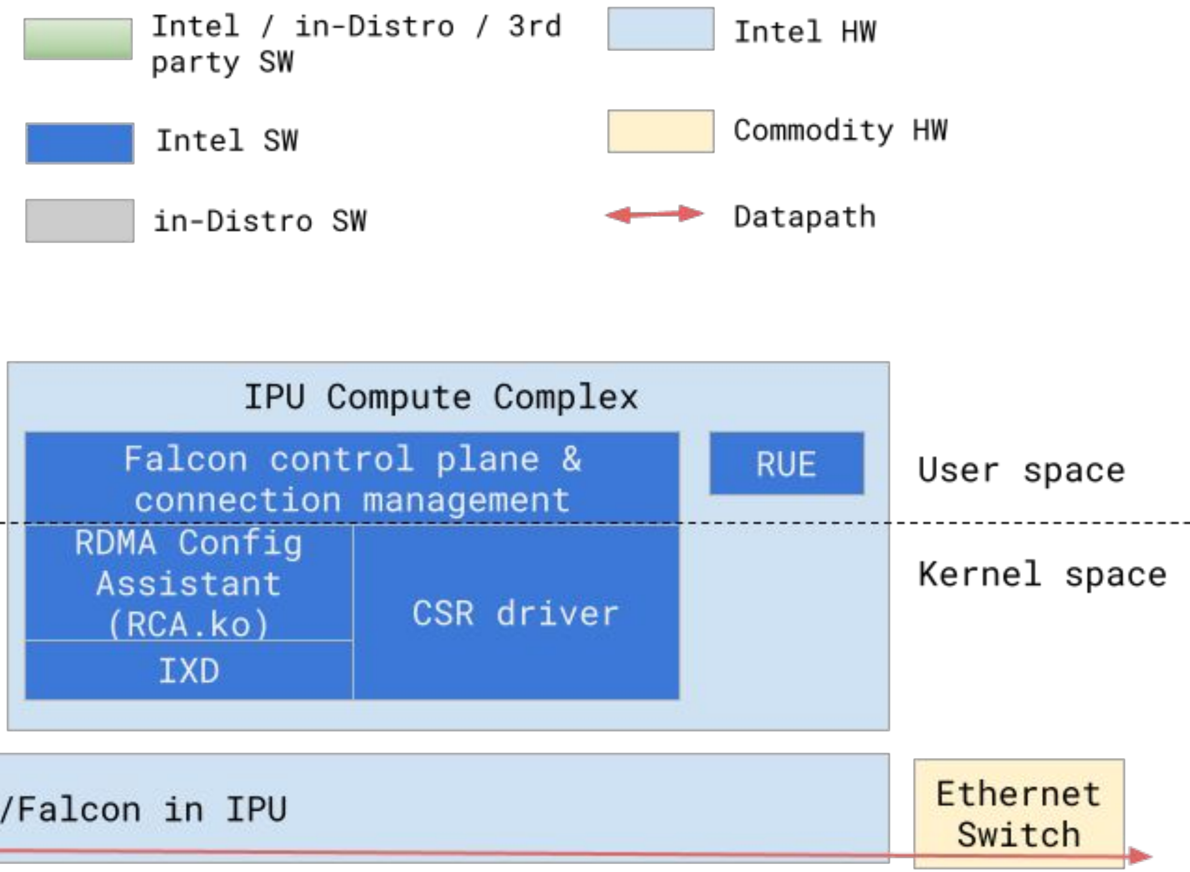
Falcon SW Components - Upstreaming Plan

RCA.ko runs on top of an auxiliary device created by the IXD driver (Intel Control Plane Driver).

IXD driver to be upstreamed.

Plan to upstream both RCA.ko and Falcon CSR driver.

User space SW (Falcon control plane, connection management, SW-RUE) will be open-sourced.



Going Forward

Falcon technology brings 10 years of advances in Low Latency, Isolation and Efficiency to hardware.

Open Technology:

- Falcon @OCP 2023 [slides][talk][Google Cloud blog post].
- Falcon Specifications released in Q1 '24 [https://github.com/opencomputeproject/OCP-NET-Falcon].
  - v0.9 of Falcon Transport, RDMA-over-Falcon, NVMe-over-Falcon.
  - v1.0 of Falcon Transport and RDMA-over-Falcon.
- Further advancements in protocol to be released in future specifications.
- RDMA/Falcon Simulator to be opened up.

Upstreaming changes to use advanced capabilities from Falcon:

- 8KB MTU support. Approach IBTA for inclusion of 8KB MTU support into the specification – could be used by RoCEv2 as well.
- iRDMA driver changes to expose basic telemetry information from Falcon.
- Expose unordered connection and Complete-in-Error to advanced applications.
- Use DirectVerbs as a baseline solution for exposing such new capabilities.

Acknowledgements

Many Google and Intel team members worked on bringing Falcon Technology to the Industry.

Google Team:

Abhishek Agarwal, Prashant Chandra, Kai Huang, Kyuho Jeong, Huadong Liu, Julian Mentasti Mesa, Behnam Montazeri, Ashwin Murthy, Jessica Ramirez, Henry Schuh, Hyojeong Seibert, Naveen Sharma, Kaiyu Shen, Arjun Singhvi, Huangze Tang, Akshat Thakkar, Jaswinder Singh, Sarin Thomas, Hassan Wassel, David Wetherall, Baijun Wu, Matt Wycklendt, Jian Yang, Rui Yang, Kira Yin, and many members of the RDMA-Falcon HW and SW teams.

Intel Team:

Suresh Kumar P Venkata, Purna Chandra Mandal, Abhishek Maheshwari, Arun Kumar Acharya, Pawel Kaminski, Krzysztof Czurylo, Chris Bednarz, Eran Rosenthal, Benny Sum, Animesh Pan, Deb Chatterjee, and many members of the RDMA-Falcon Architecture, HW and SW teams.