SRECON EUROPE MIDDLE EAST AFRICA

HUAWEI

HUAWEI CLOUD

HyperRouter: Lessons learned from building a **Level 4 load balancing service**

Linhua Tang *(aka **James**),* software architect/tech lead

Jayaganesh Kalyanasundaram, principal software engineer

Cloud Reliability Lab, Huawei Ireland Research Center

Agenda

- Scale of Huawei Cloud
- Huawei Cloud Deterministic Operation
- Huawei Cloud Load Balancing Platform Landscape
- Objectives and Components of Level 4 Load Balancing Service
- High performance data plane in user space
- Self-healing and isolation as first-class design principle
- Minimizing dependency with P2P design
- Operational Excellence – Blast Radius Reduction
- Customer Centric Observability with CUJ

Scale of Huawei Cloud

Servers 20% growth

33 public regions

0 P1

incident for last four years

Customers

- Corporation users: **20000+**
- Paying users: **3 million+**
- Developers: **10 million+**

Services

- Cloud services: **895**
- Prod environment changes: **9.52 M/year**

Hardware

- Servers: **2 million+**
- Data Center: **1300+**

Fundamental Services SLO

99.999%

Core services SLO

99.99%

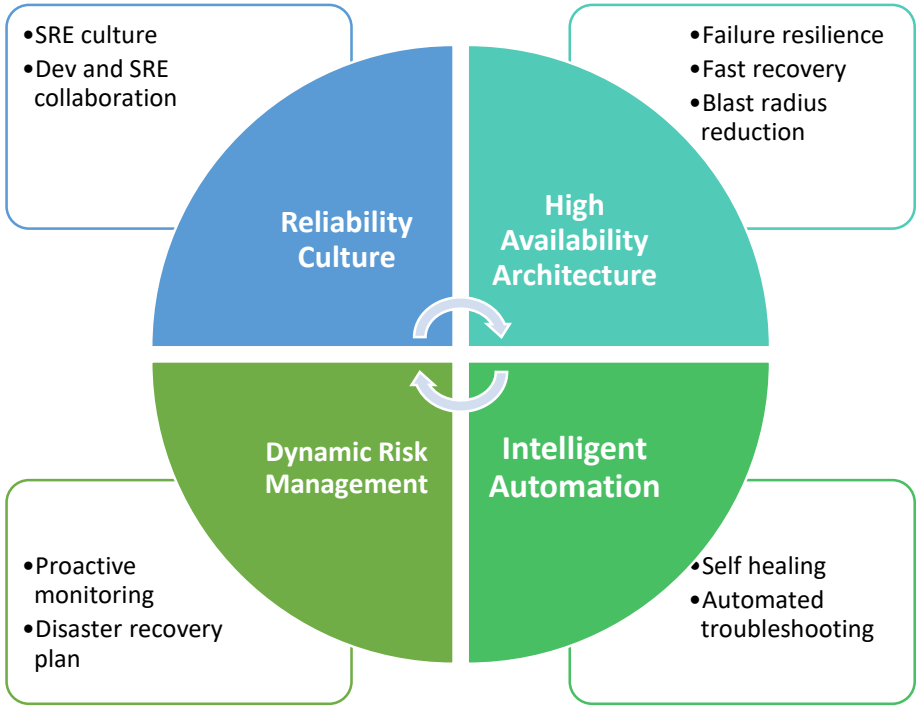
A global storage and computing network with wide coverage, low latency, and meeting customer experience needs

CloudOcean (Global)	CloudSea (Regional)	CloudLake (Edge access)
Scale of 1 million servers China<30ms Overseas<50ms	Scale of 100000 servers Latency<10ms	Scale of 5000 servers Latency<5ms

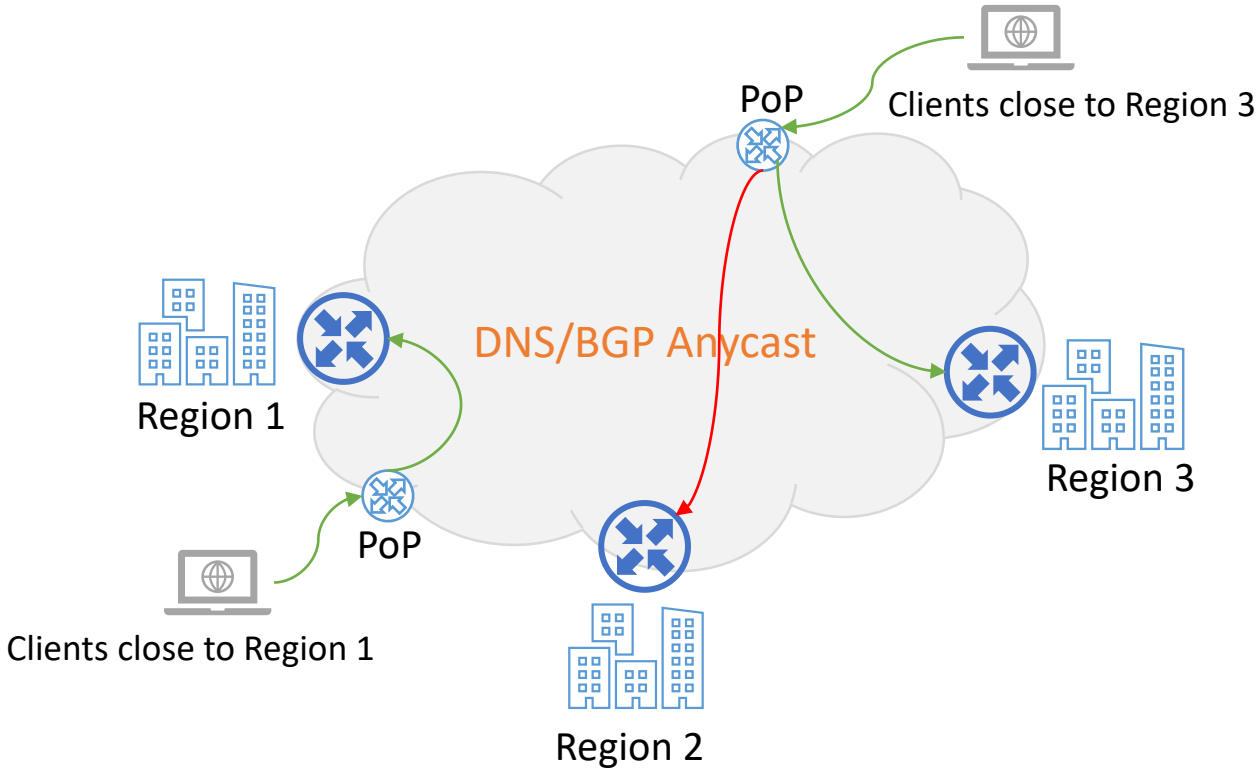
HUAWEI

Huawei Cloud Deterministic Operation

- **Deterministic operation** is a systematic framework to ensure system reliability
  - Predictable performance, predictable scalability, and disaster recovery with certainty.
  - Working backwards: how a system will be operated at design phase?
- **Global Load Balancing platform** is one of the cornerstones of this framework
  - e.g. Cross region, cross AZ traffic steering and automated failover.



Deterministic Operations Framework

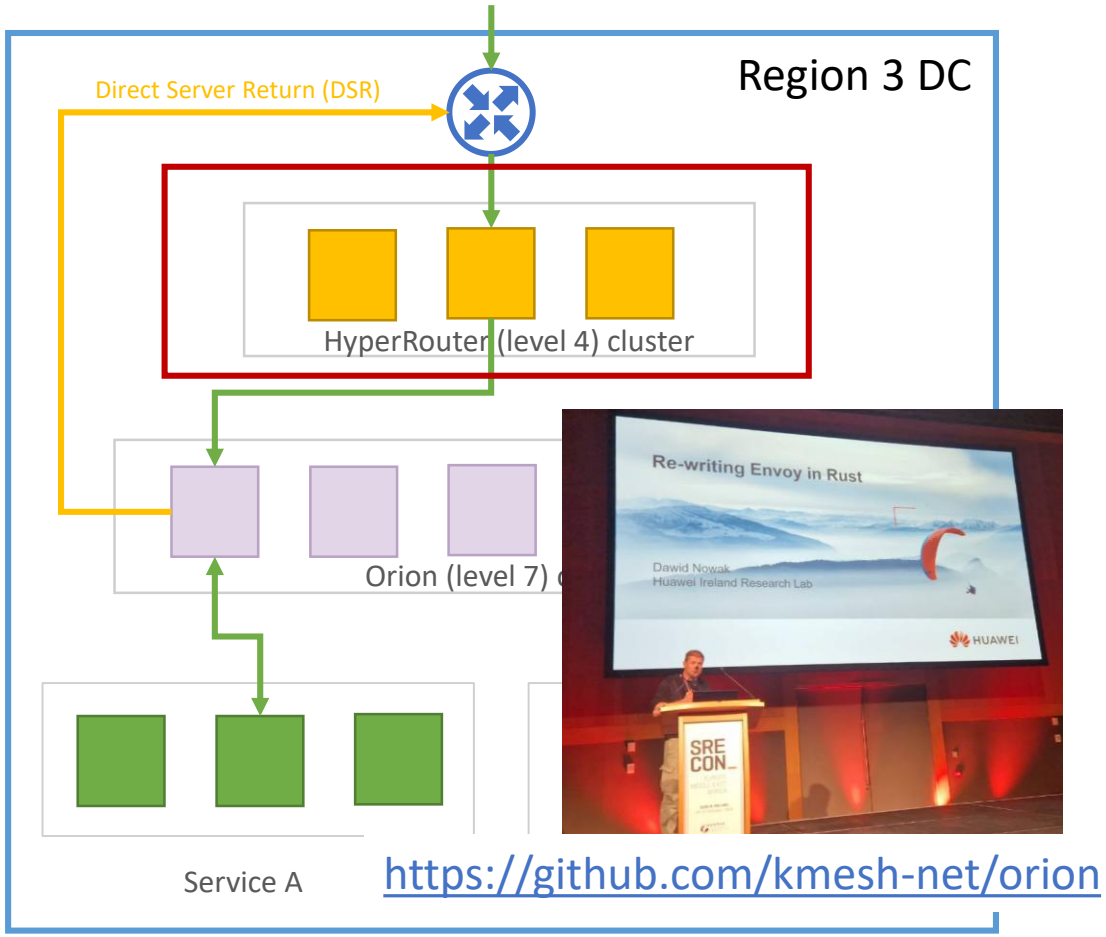


Global Load Balancing

HUAWEI

Huawei Cloud Load Balancing Platform Landscape

- *A load balancer is a system that distributes network traffic across multiple backend servers to improve the availability, scalability and performance of another system.*
- Scope of this presentation:
  - Experiences about building a LB services, not how to use LB.
  - Software LB, not hardware LB.
  - Designed for large-scale multitenancy platform, not for all scenarios.



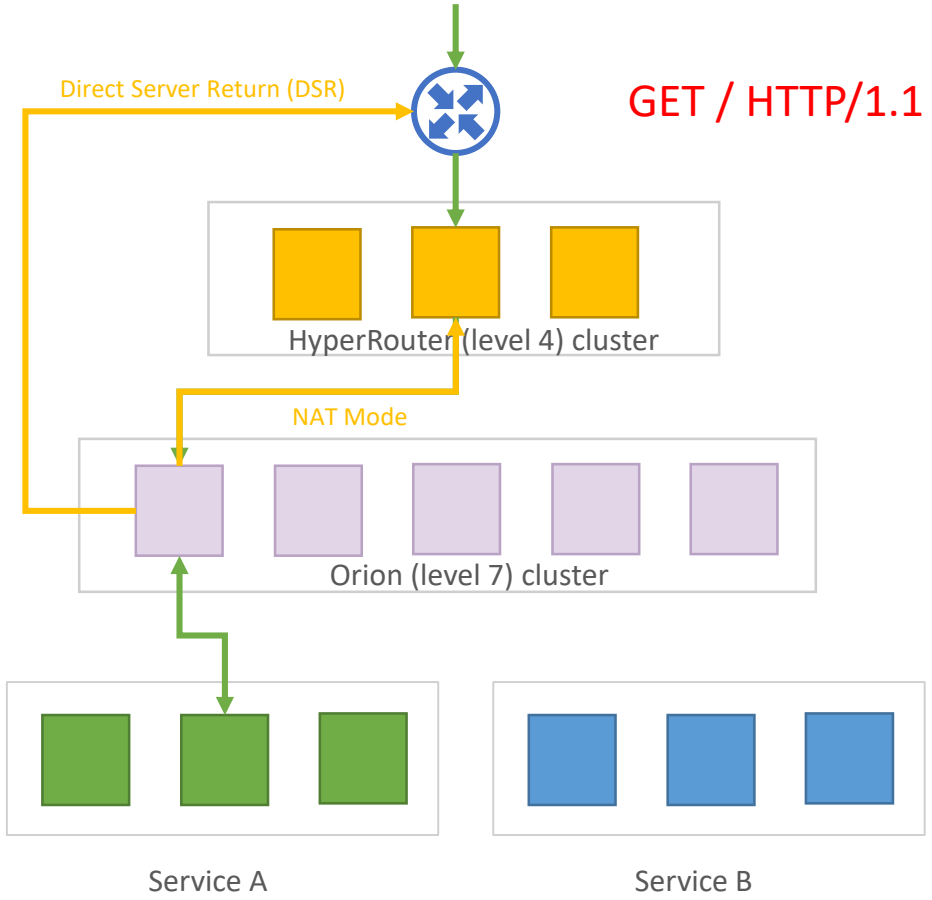
LB services are the reliability foundation of Huawei Cloud services,

but how do we guarantee the reliability of the load balancer itself?

HUAWEI

Co-designing of Level 4 and Level 7

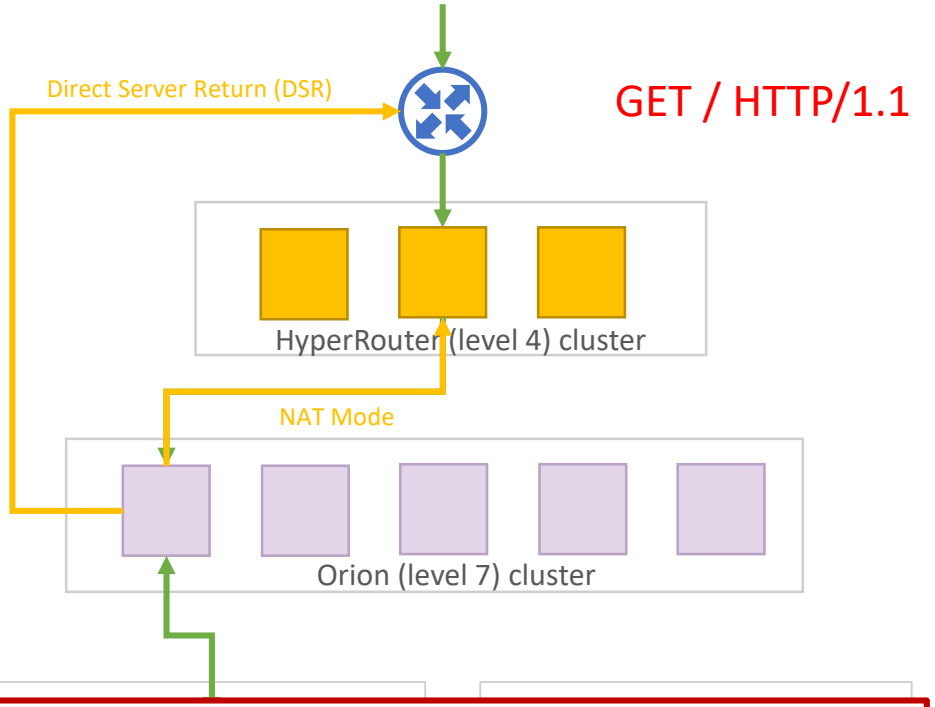
- L7LB offers stateful application features
  - TLS termination
  - HTTP header/body processing
  - Security policies
  - Response transformation
  - L7LB is not resilient to node failure or scaling up/down because ECMP may mess up healthy TCP connections.
- L4LB performs stateless packet forwarding
  - Consistent hashing
  - Connection tracking
  - Direct Server Return (DSR) - reduces the overhead of processing egress traffic



Anycast: multiple server share the same IP addresses
ECMP: distribute traffic across multiple paths with equal cost.

Co-designing of Level 4 and Level 7

- L7LB offers stateful application features
  - TLS termination
  - HTTP header/body processing
  - Security policies
  - Response transformation
  - L7LB is not resilient to node failure or scaling up/down because ECMP may mess up healthy TCP connections.
- L4LB performs stateless packet forwarding
  - Consistent hashing
  - Connection tracking



Lesson 1: co-designing related systems (e.g. L4 and L7 LB) reduces complexity while enhancing overall reliability.

Anycast: multiple server share the same IP addresses
ECMP: distribute traffic across multiple paths with equal cost.

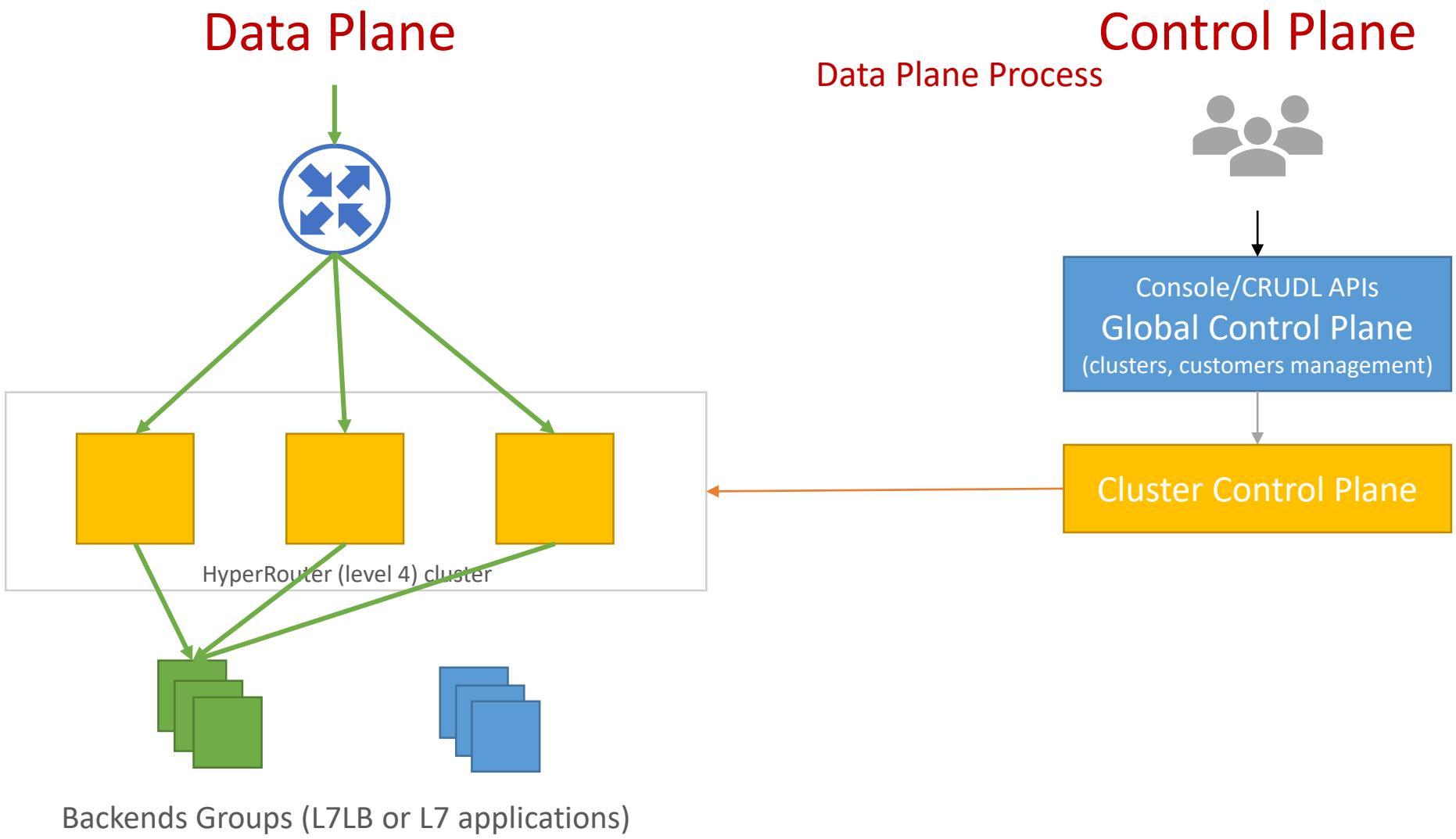
HUAWEI

Objectives of Level 4 Load Balancing Service

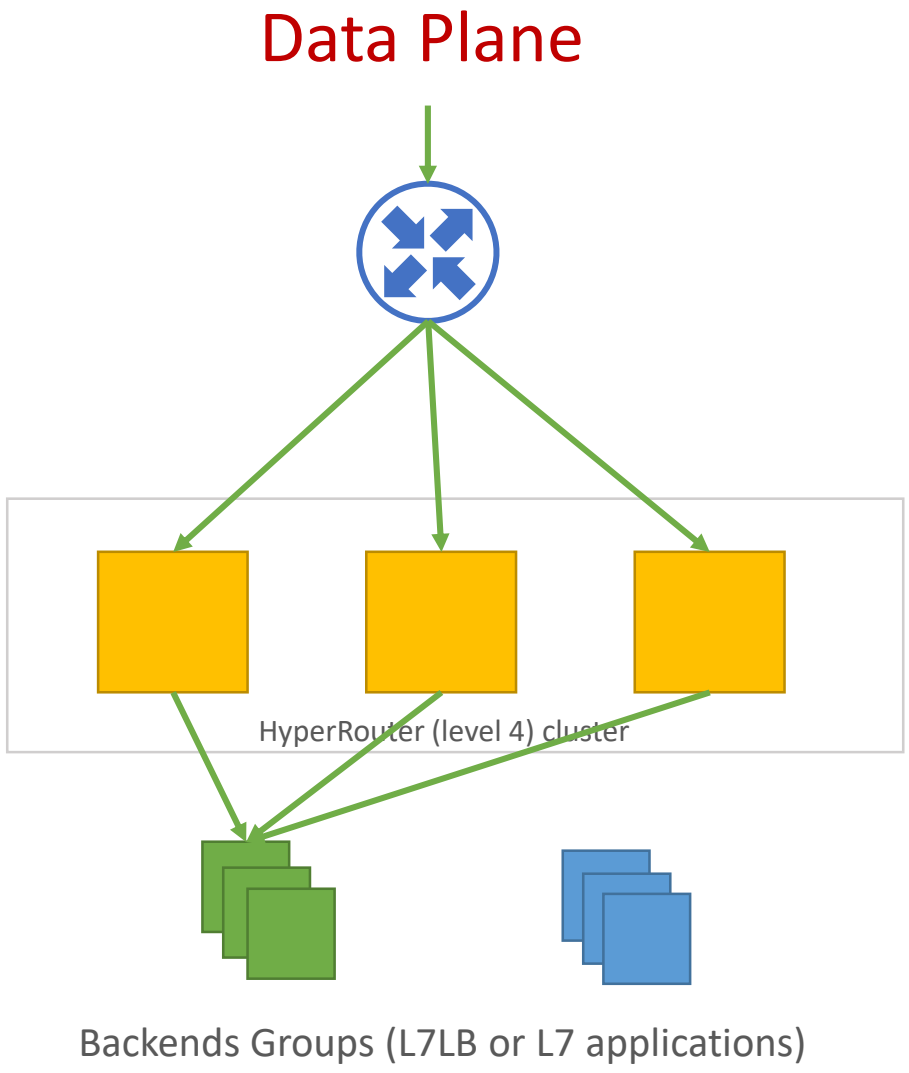
Objectives: simple but not easy

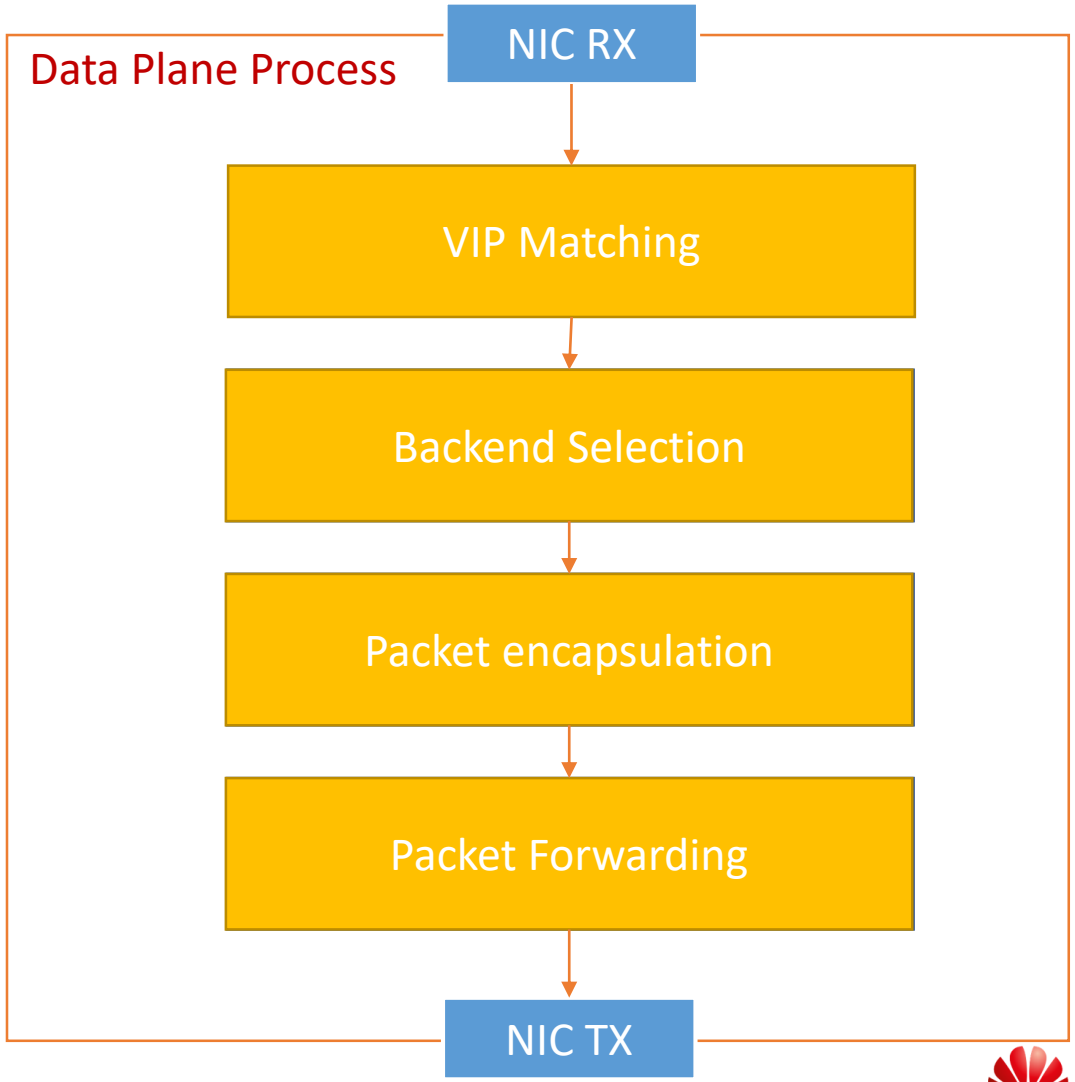
- **High performance** – close to line-rate speed on average packet size with minimized cost, such as 10 millions PPS/CPU core; no significant overhead, ns per packet processing.
- **High scalability** – to handle fast growing traffic of Huawei Cloud (e.g. billions concurrent TCP connections), we need to scale out horizontally across multiple data centers.
- **High reliability** – customers always expect zero downtime for LB services, we have to ensure high availability during operations, resilient to partial hardware and network failures.
- **Future proof** – evolvable for future needs: new features, new capabilities of new NICs, new protocols, new scenarios, etc.

Components of Level 4 Load Balancing Service



Components of Level 4 Load Balancing Service



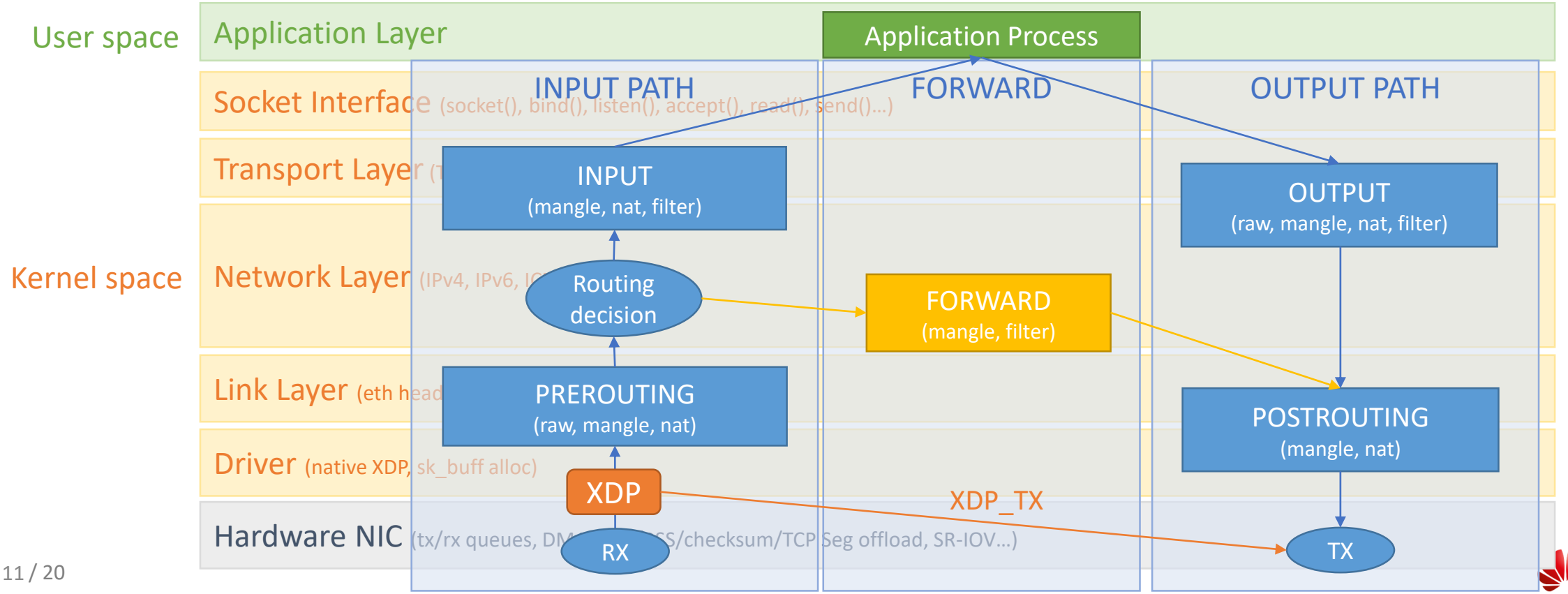


HUAWEI

High performance data plane in user space

Why Linux kernel is the bottleneck (for the specific case)?

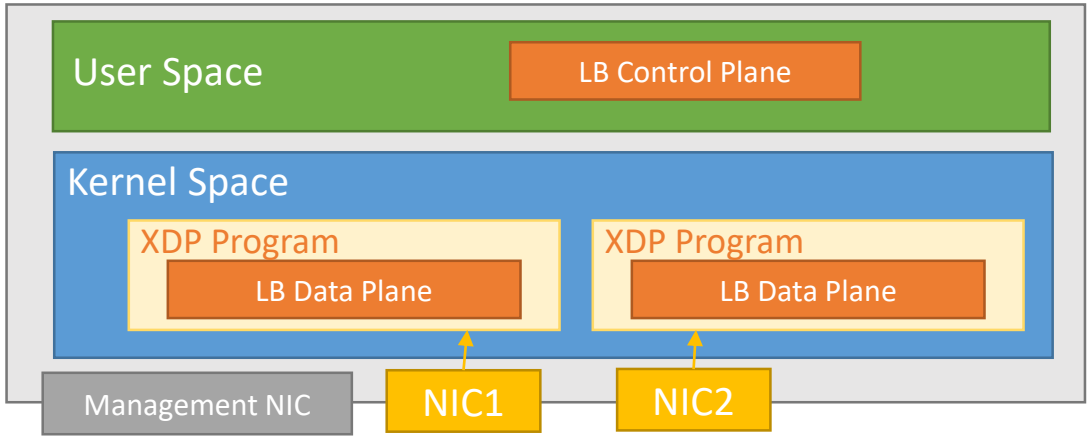
- Linux kernel networking stack is remarkable for general purpose application.
- Flexible but not optimized for ultra-low latency: context switches, memory copies, etc.
- Hard to achieve near line-rate speed: need intensive tuning and more CPU cores.



High performance data plane in user space

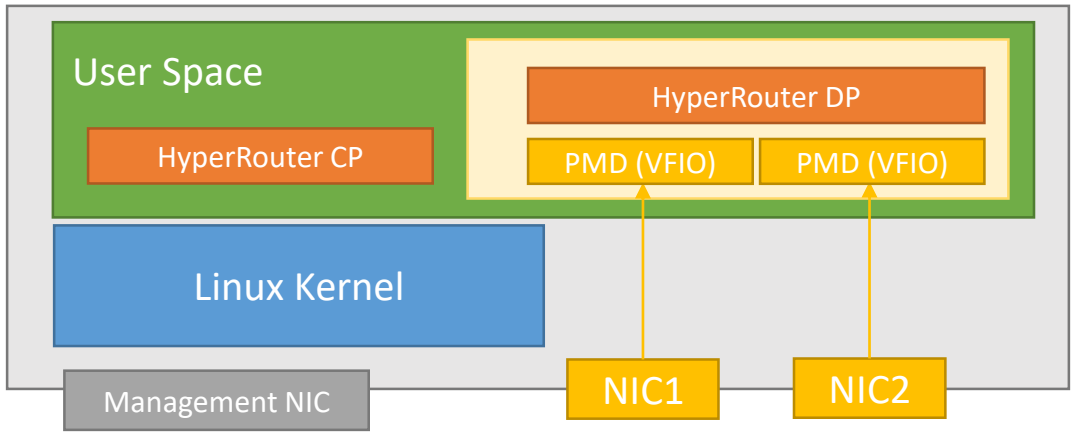
XDP (eXpress Data Path) is a framework that makes it possible to perform high-speed packet processing within BPF applications.

- Kernel space (before socket buffer allocation)
- CPU efficiency (fewer cycles per packet)
- Restricted programming model
- XDP program attached per NIC



DPDK (Data Plane Development Kit) is a set of data plane libraries and NIC polling-mode drivers for processing network packets in user space bypassing Linux Kernel.

- Full control in user-space (C, NIC features)
- Mature multiple NICs support
- ✓ Memory management with huge pages
- ✓ Dedicated NICs and CPUs
- ✓ Polling mode (CPU utilization always 100%)



HUAWEI

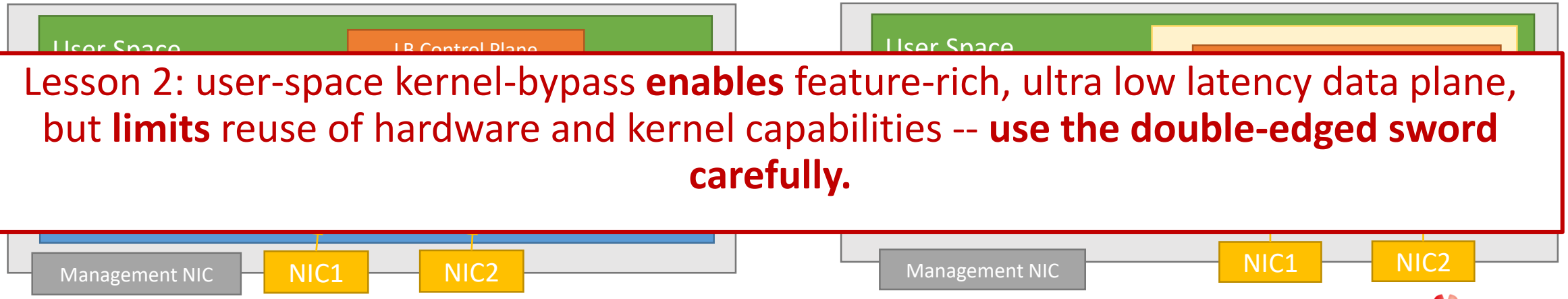
High performance data plane in user space

XDP (eXpress Data Path) is a framework that makes it possible to perform high-speed packet processing within BPF applications.

- Kernel space (before socket buffer allocation)
- CPU efficiency (fewer cycles per packet)
- Restricted programming model
- XDP program attached per NIC

DPDK (Data Plane Development Kit) is a set of data plane libraries and NIC polling-mode drivers for processing network packets in user space bypassing Linux Kernel.

- Full control in user-space (C, NIC features)
- Mature multiple NICs support
- ✓ Memory management with huge pages
- ✓ Dedicated NICs and CPUs
- ✓ Polling mode (CPU utilization always 100%)



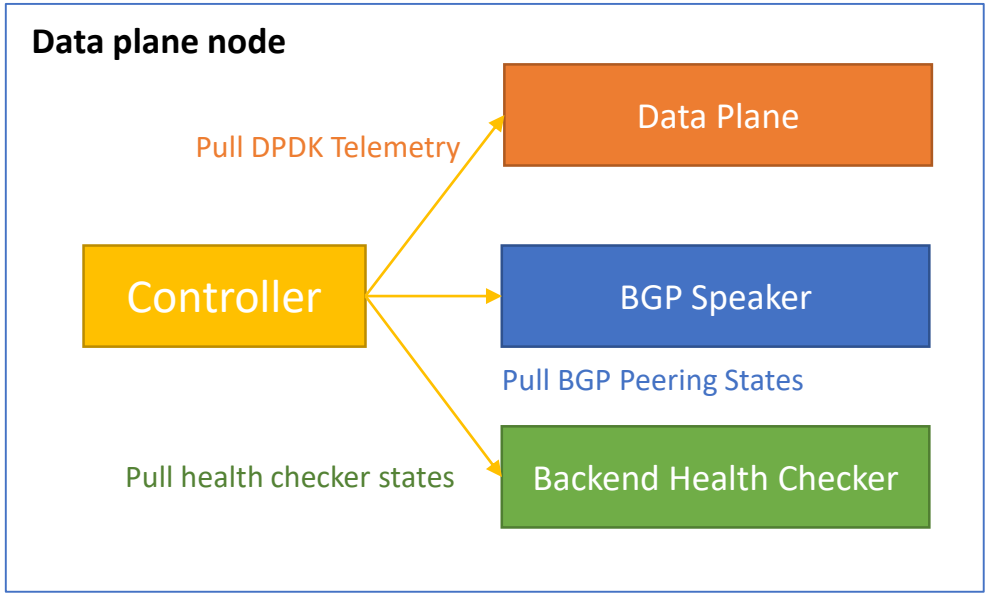
Lesson 2: user-space kernel-bypass **enables** feature-rich, ultra low latency data plane, but **limits** reuse of hardware and kernel capabilities -- **use the double-edged sword carefully.**

HUAWEI

Self-healing and isolation as first-class design principle

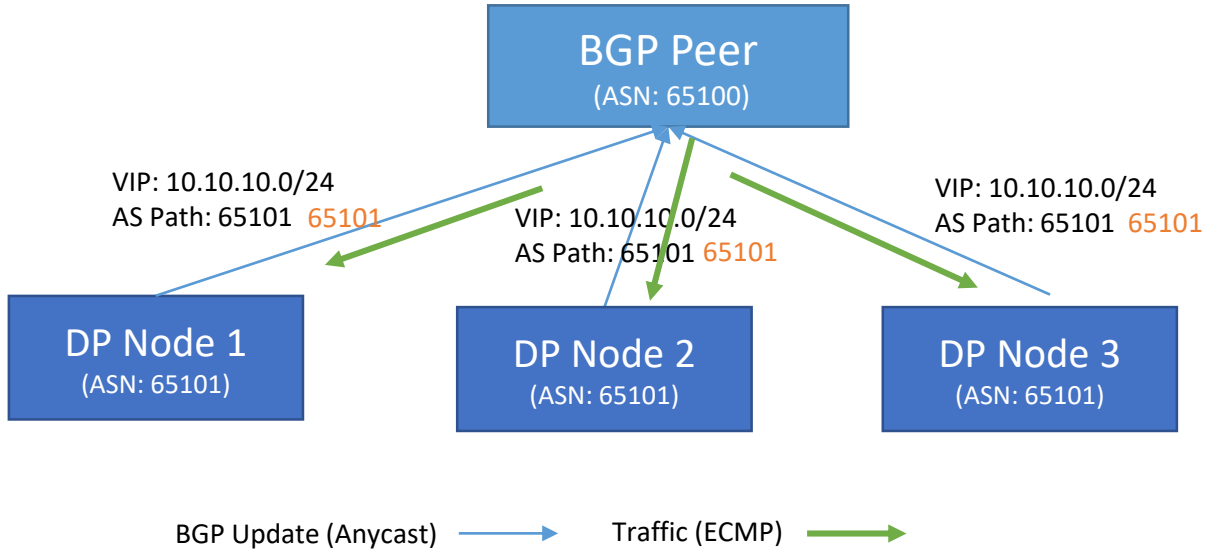
Self health checking:

- Controller pull telemetries and states
- Controller validate the health states
- Controller makes decision and take actions (Restart processes, reconfiguration, isolation)



Node isolation:

- Anycast and ECMP are the foundations for the isolation mechanism
- Withdraw BGP vs decrease priority (prepending AS-Path)



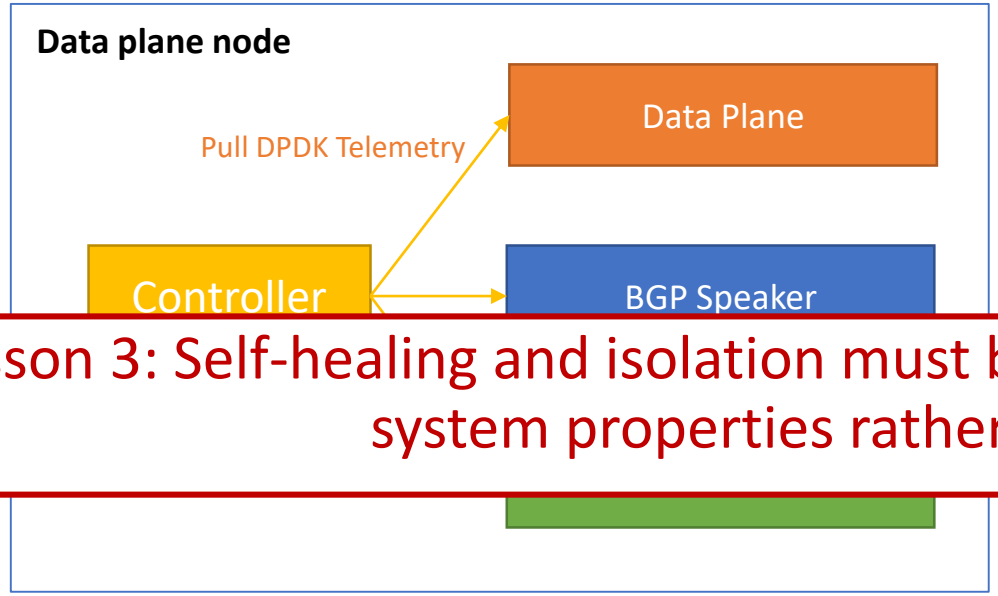
Self-healing and isolation as first-class design principle

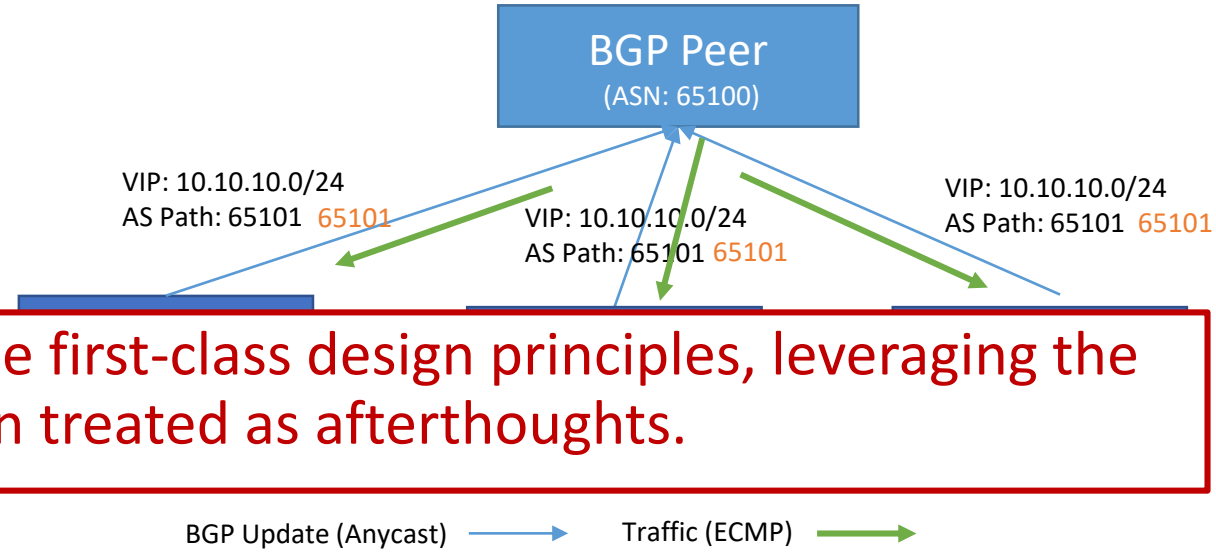
Self health checking:

- Controller pull telemetries and states
- Controller validate the health states
- Controller makes decision and take actions (Restart processes, reconfiguration, isolation)

Node isolation:

- Anycast and ECMP are the foundations for the isolation mechanism
- Withdraw BGP vs decrease priority (prepending AS-Path)

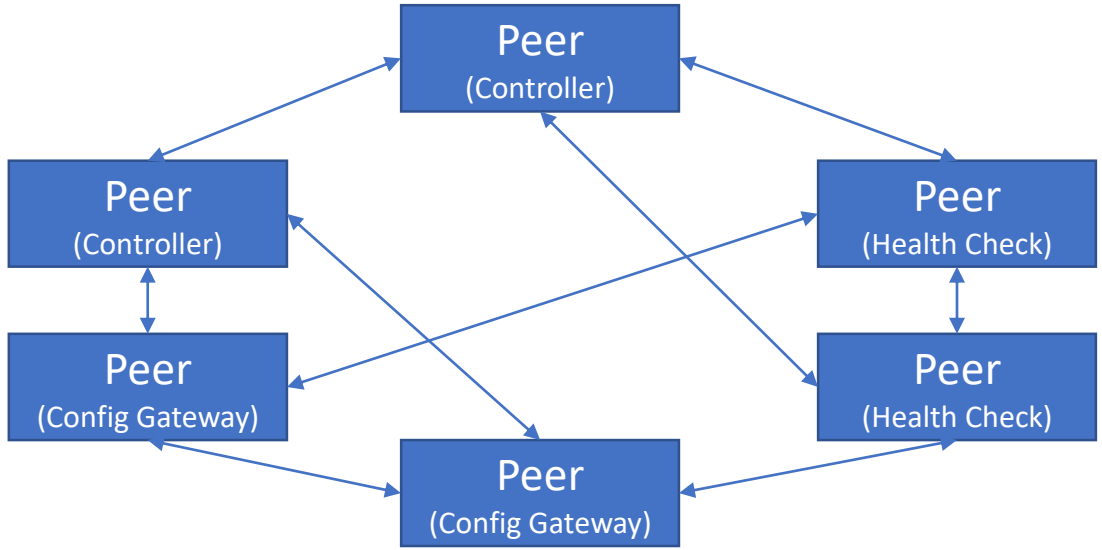




Lesson 3: Self-healing and isolation must be the first-class design principles, leveraging the system properties rather than treated as afterthoughts.

Minimizing dependency with P2P for high resilience

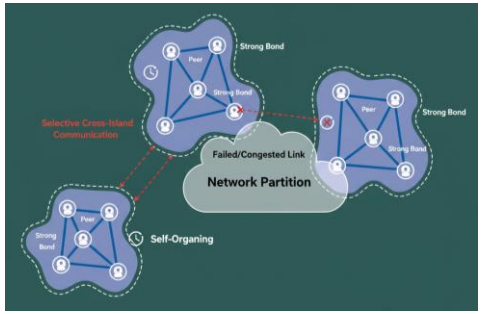
- Principles
  - Minimize external dependency
  - Resilient to network partition
  - Self-healing
- Peer to peer vs centralized control plane
- Built from scratch with minimized external dependency.



P2P Control Plane: A P2P control plane is a decentralized network architecture where peers collaborate to manage and control network resources, rather than relying on a centralized server. Unlike traditional client-server control planes, a P2P control plane uses distributed mechanisms for functions like auto-discovery, configuration, and status monitoring, providing increased robustness, scalability, and network partition tolerance.

New P2P Datastore Design: Key Innovations

- Full Data Replication
  - All peers store <key, value, version> triplets
  - Gossip sync → eventual consistency
- Bond-Based Mesh Communication
  - Peers linked via bidirectional RPC bonds
  - Reliable & efficient transfer
- Islands for Partition Resilience
  - Peers form “islands” in partitions
  - Controlled cross-island traffic
- Scalable & Balanced
  - O(log P) hops in islands
  - Linear CPU/bandwidth scaling
- Resilient to Flaky Networks
  - Mesh reorganizes in O(P) ticks
  - Tolerates failures & congestion



Peer-to-peer framework

Application Layer
Role specific logic, subscribe to specific messages

Communication Layer
gRPC based bonds (connections) management, message propagations

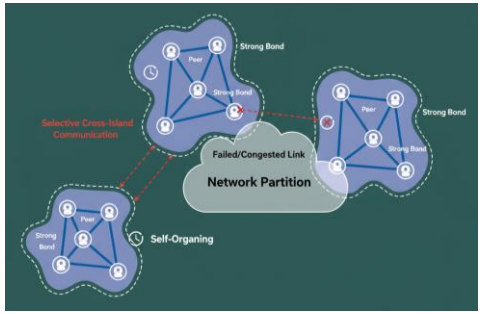
KV-Storage Semantics
Process message in RAM, version conflict resolving

Persistence Layer
Local Disk, Remote Storage

HUAWEI

New P2P Datastore Design: Key Innovations

- Full Data Replication
  - All peers store <key, value, version> triplets
  - Gossip sync → eventual consistency
- Bond-Based Mesh Communication
  - Peers linked via bidirectional RPC bonds
  - Reliable & efficient transfer
- Islands for Partition Resilience
  - Peers form “islands” in partitions
  - Controlled cross-island traffic
  - Linear CPU/bandwidth scaling
- Resilient to Flaky Networks
  - Mesh reorganizes in O(P) ticks
  - Tolerates failures & congestion



Peer-to-peer framework

Application Layer
Role specific logic, subscribe to specific messages

Communication Layer
gRPC based bonds (connections) management, message propagations

Persistence Layer
Local Disk, Remote Storage

Lesson 4: Peer-to-peer architecture is a strong choice for systems that must minimize external dependencies.

HUAWEI

P2P Design: Formal Methods Verification

Bonding eventual consistency:

- As a **liveness property**

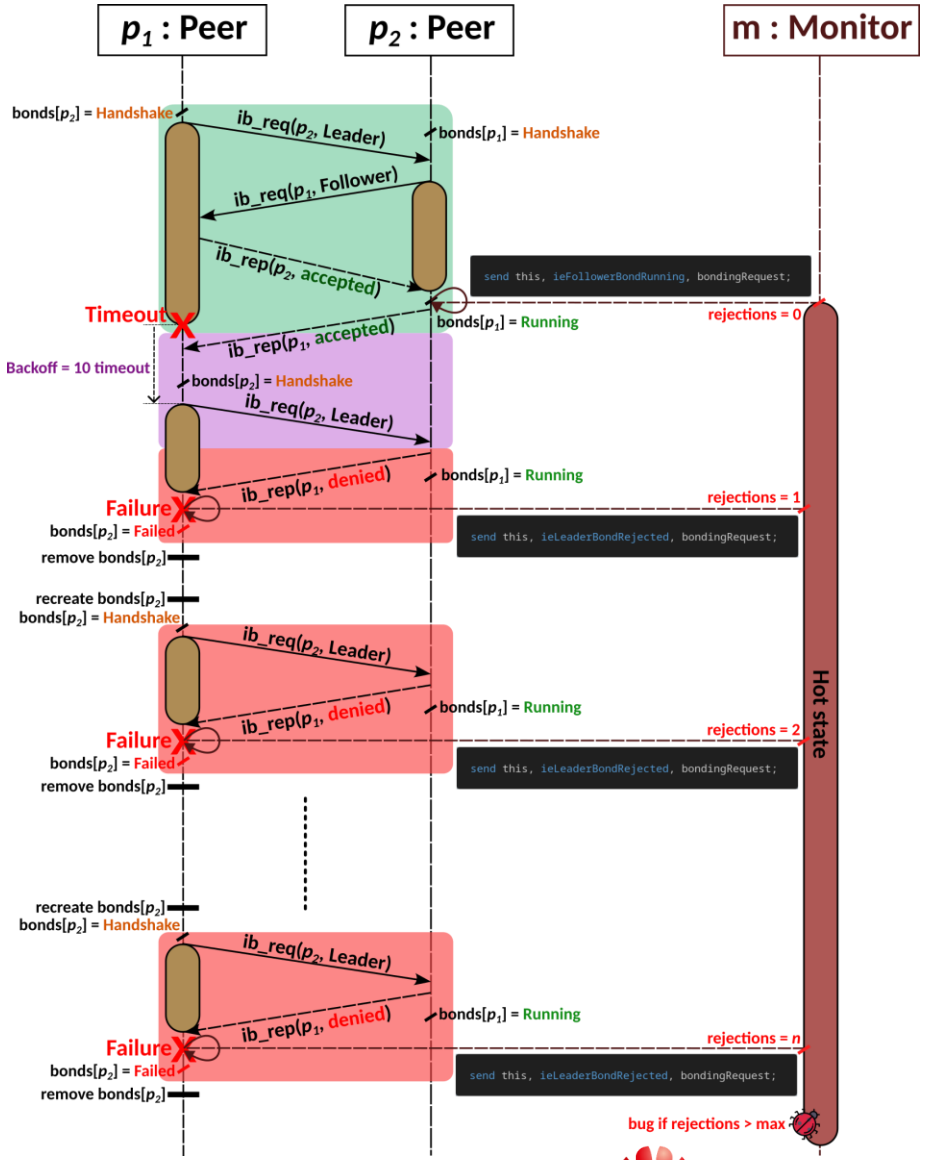
$$p_2.\texttt{bonds}[p_1].\texttt{state} = \texttt{Running} \rightsquigarrow p_1.\texttt{bonds}[p_2].\texttt{state} = \texttt{Running}$$

- This $\rightsquigarrow$ means ***"leads to"***
- The property **could eventually be satisfied by the leader peer**, **but only after an unpredictable number of retries if the system falls into this situation...**

P and TLA+ Modelling

P Modeling: build a formal model of the P2P protocol in P to verify correctness and reliability requirements.

Bug discovered: A bug in the peer bonding logic, preceding the synchronization phase, caused peers to persist in bonding-retry loop for an unpredictable number of retries.



HUAWEI

Operational Excellence – Blast Radius Reduction

Blast Radius Reduction matters for multi-tenancy LB service

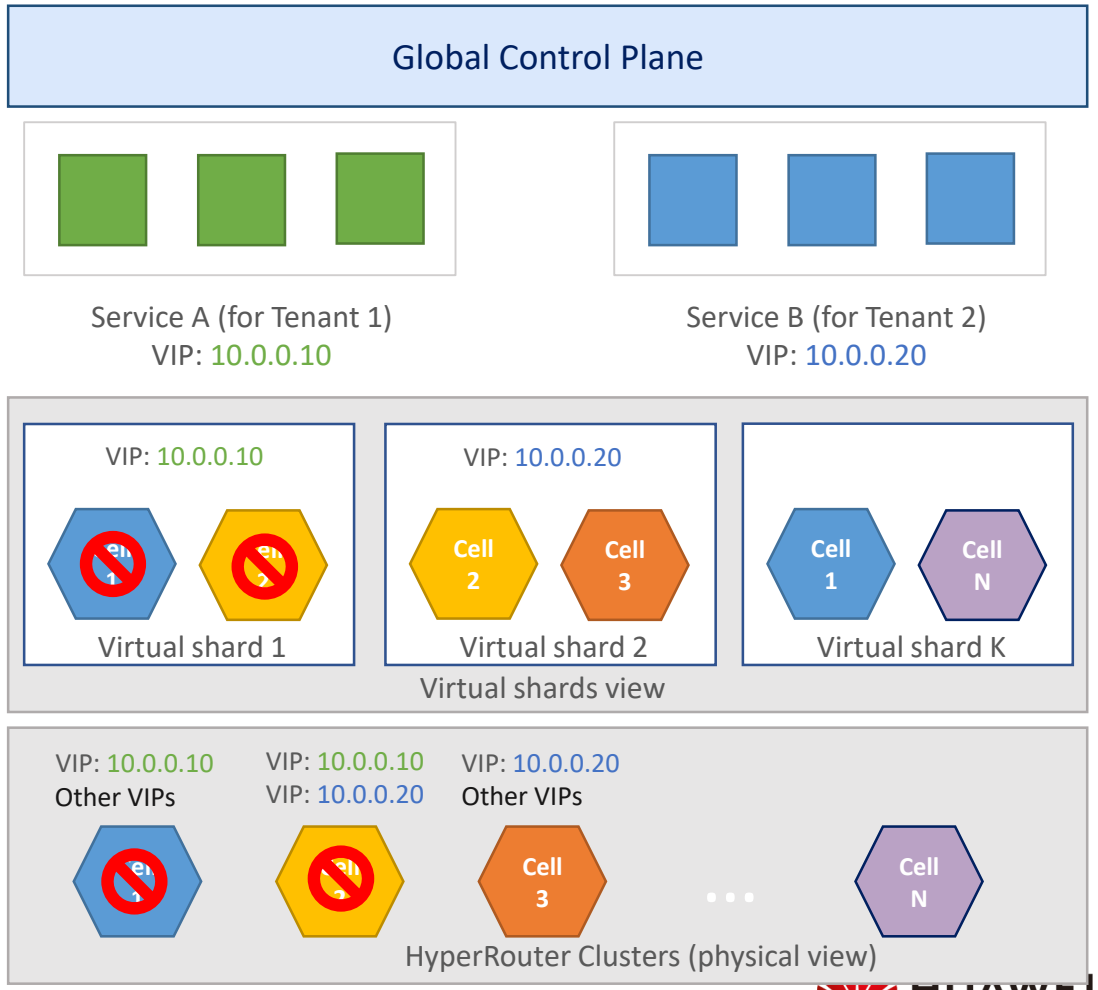
- Noisy neighbors
- Hardware failures

How Cell-base architecture and shuffle sharding work for L4LB?

- Control plane plays the key role
- Cell (cluster) management
- Virtual shards management
- Tenant service and virtual shard allocation and relocation

Static sharding: 1/(N/r) = 1/(10/2) = 1/5

Shuffle sharding: 1/C(N, r) = 1/C(10, 2) = 1/45



Refers to *Shuffle Sharding: Massive and Magical Fault Isolation*
https://aws.amazon.com/blogs/architecture/shuffle-sharding-massive-and-magical-fault-isolation/

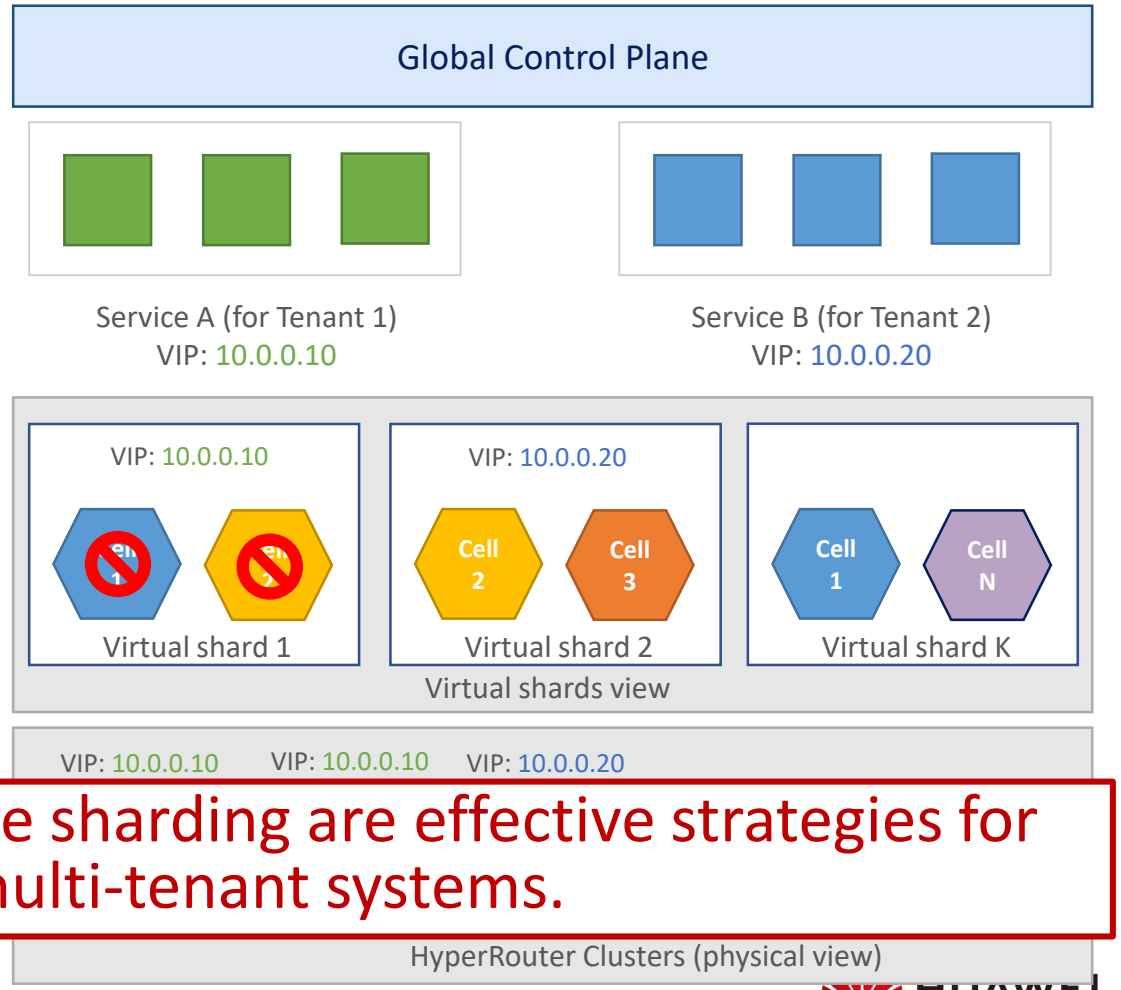
Operational Excellence – Blast Radius Reduction

Blast Radius Reduction matters for multi-tenancy LB service

- Noisy neighbors
- Hardware failures

How Cell-base architecture and shuffle sharding work for L4LB?

- Control plane plays the key role
- Cell (cluster) management
- Virtual shards management
- Tenant service and virtual shard allocation and relocation



Static sharding: 1/(N/r) = 1/(10/2) = 1/5

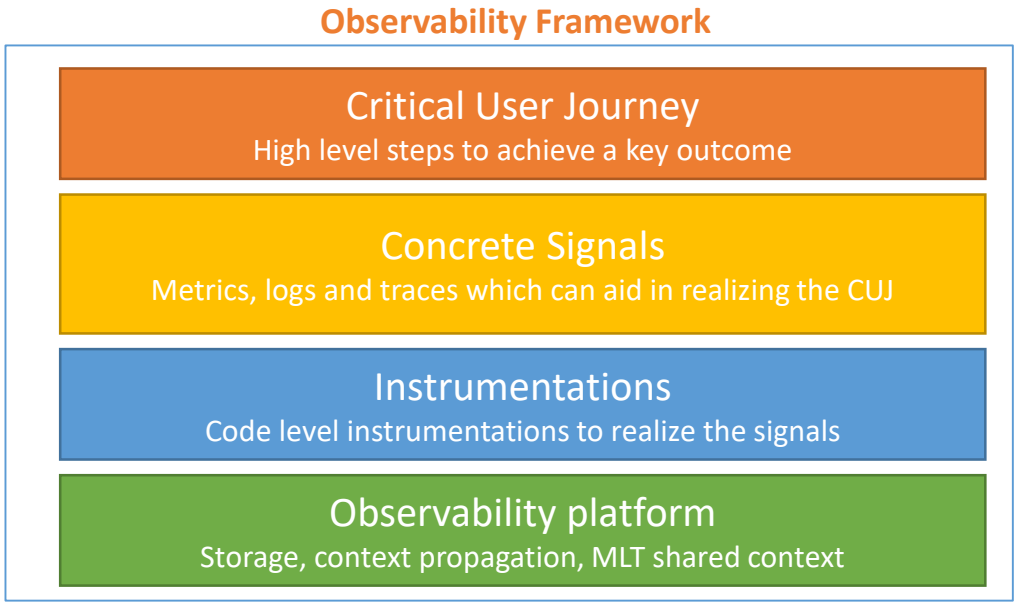
Lesson 5: Cell-base architecture and shuffle sharding are effective strategies for reducing blast radius in multi-tenant systems.

https://aws.amazon.com/blogs/architecture/shuffle-sharding-massive-and-magical-fault-isolation/

Observability

How can Observability be done correctly?

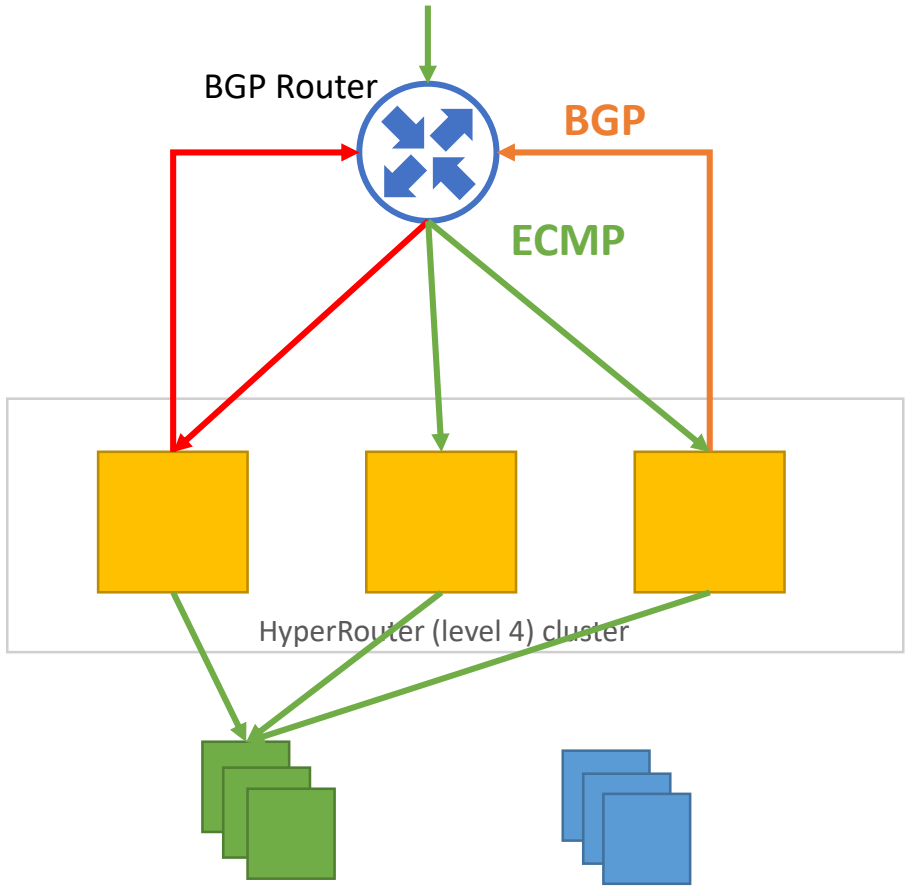
- Choosing the right signals to measure and the corresponding instrumentations
- From a bunch of preconfigured signals, filtering out those which matter the most
- Signal behavior root caused to tactical production/code level configurations



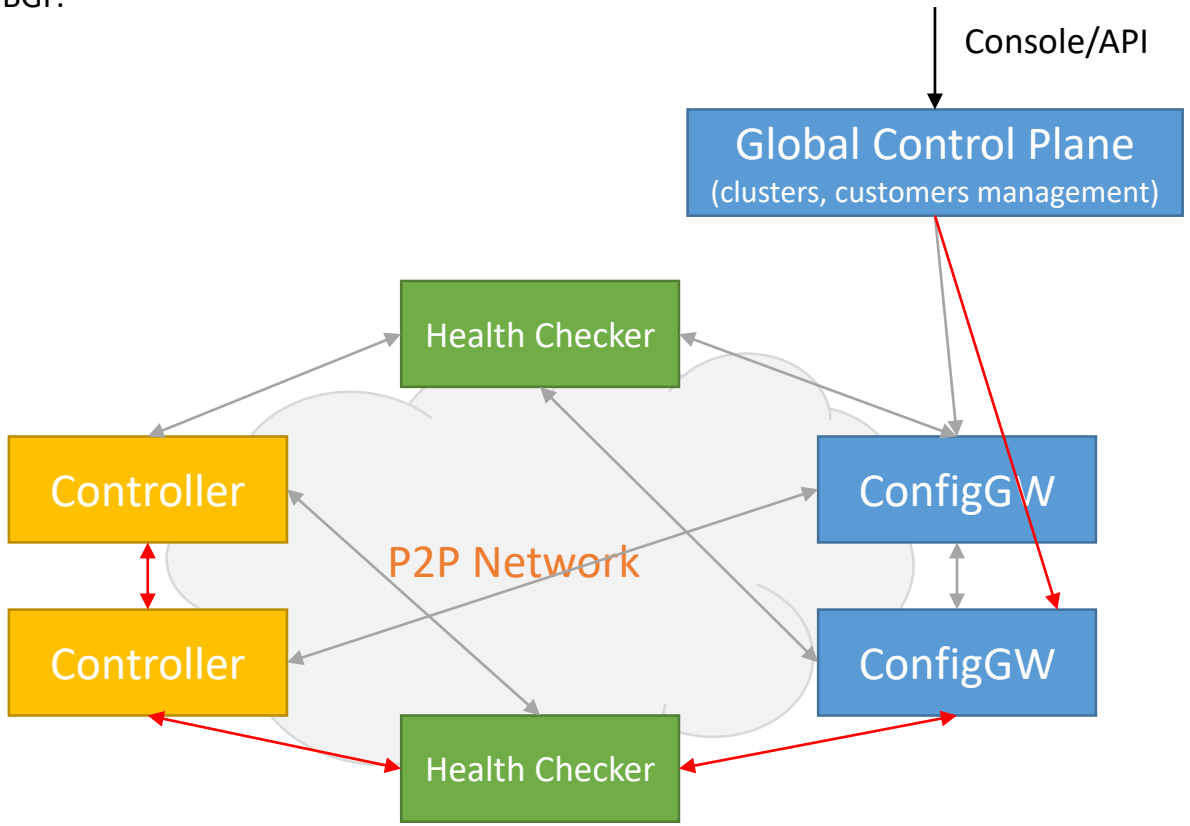
Critical User Journey: A critical user journey (CUJ) is the sequence of essential actions a user must take to experience the core value of a product. It represents the minimum viable path a user follows to achieve the key outcome that defines the product's value.
Mapping a CUJ involves defining the product's core value, identifying the key user segments for whom the journey is critical, and charting the end-to-end sequence of user actions that deliver that value, focusing only on the steps that truly matter to the user achieving the core outcome.

Customer Centric Observability

Data Plane: The system accepts the VIP to be resolved and load balances them onto the available physical backend and the chosen backend processes the packet and returns the DNS resolution response to the original caller.

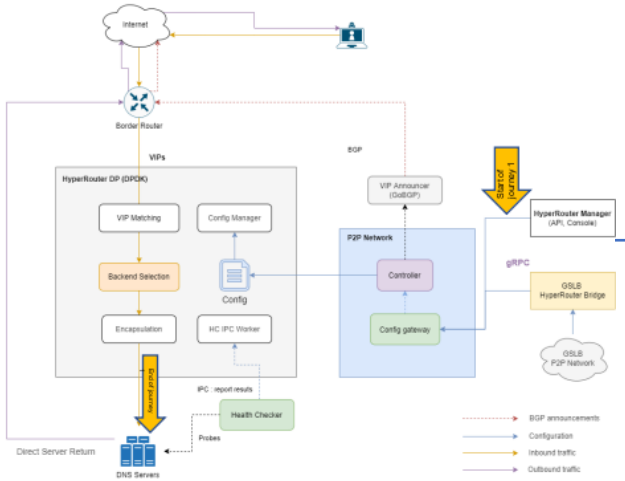


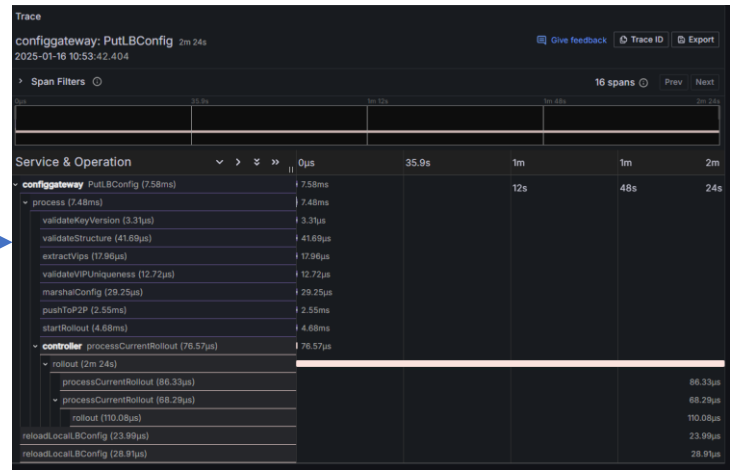
Control Plane: The system receives the LB configurations from an external system (the UI which allows the user to enter them) and populates the new virtual LB routes on the BGP.



HUAWEI

Realizing CUJs with an example



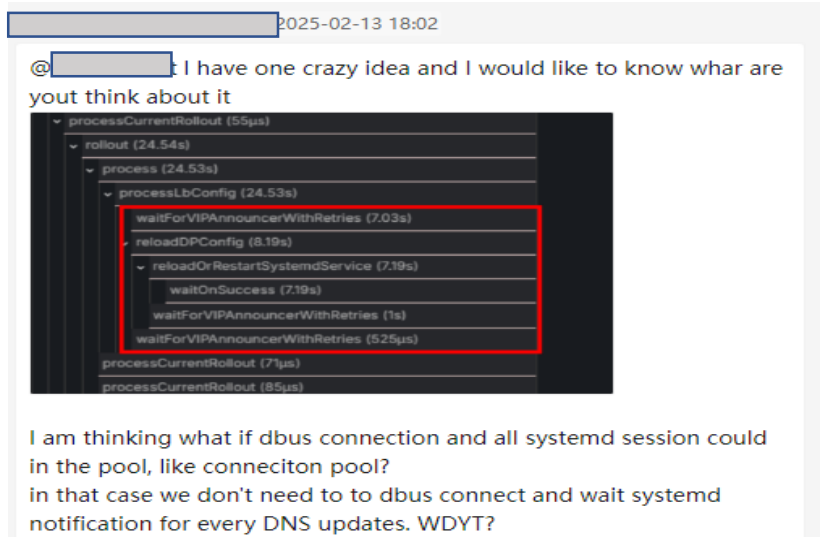


Helps realize:

1. Performance of the operator journey requests
2. Breakdown of performance metrics by the individual components
3. Capture the e2e journey with the blackbox of DPDK interactions as the dataplane monitoring. Answers questions like Data plane latency which is otherwise not possible.

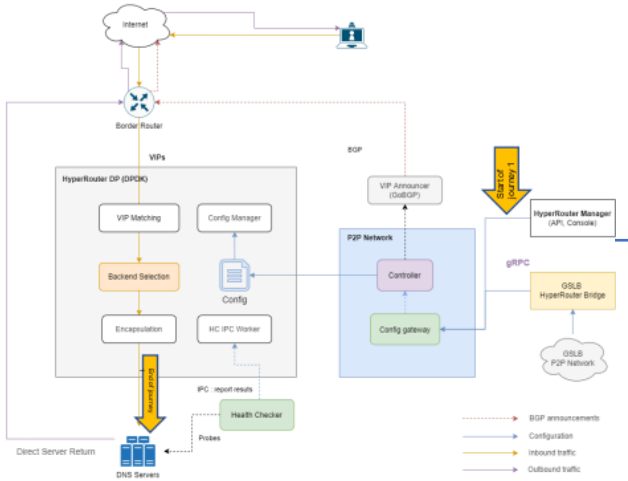
Control Plane CUJ: The system receives the LB configurations from an external system (the UI which allows the user to enter them) and populates the new virtual LB routes on the BGP.

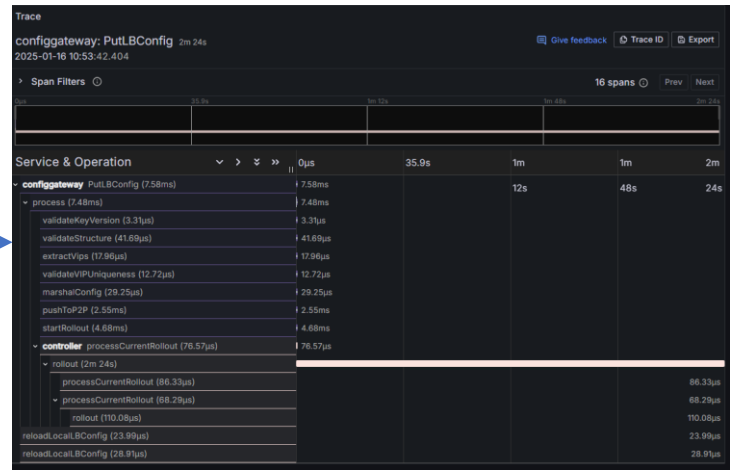
"Engineers of the Hyper Router team did not had in depth visibility into connections and how they operate. Trace based CUJs provided them with in depth visibility into how system works. Hence they already started improving their system based on this information." – developer in the team



HUAWEI

Realizing CUJs with an example



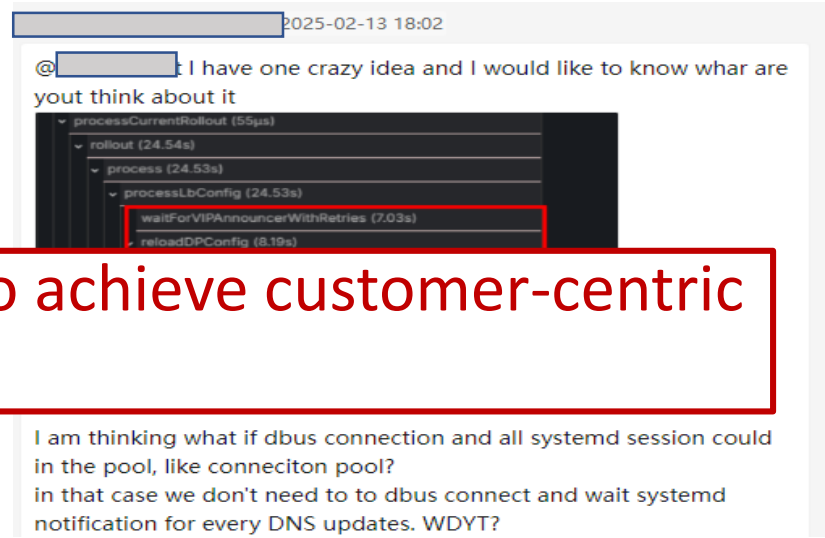


Helps realize:

1. Performance of the operator journey requests
2. Breakdown of performance metrics by the individual components
3. Capture the e2e journey with the blackbox of DPDK interactions as the dataplane monitoring. Answers questions like Data plane latency which is otherwise not possible.

Control Plane CUJ: The system receives the LB configurations from an external system (the UI which allows the user to enter them) and populates the new virtual LB routes on the BGP.

"Engineers of the Hyper Router team did not had in depth visibility into



Lesson 6: CUJ is powerful framework and practice to achieve customer-centric observability.

improving their system based on this information." – developer in the team

HUAWEI

Recap

- Lesson 1: co-designing L4 and L7 load balancing services reduces complexity while enhancing overall reliability.
- Lesson 2: user-space kernel-bypass enables feature-rich, ultra low latency data plane, but limits reuse of hardware and kernel capabilities -- use the double-edged sword carefully.
- Lesson 3: Self-healing and isolation must be the first-class design principles, leveraging the system properties rather than treated as afterthoughts.
- Lesson 4: Peer-to-peer architecture is a strong choice for systems that must minimize external dependencies.
- Lesson 5: Cell-base architecture and shuffle sharding are effective strategies for reducing blast radius in multi-tenant systems.
- Lesson 6: CUJ is powerful framework and practice to achieve customer-centric observability.

Thank you

Q&A