# QP-based SRv6 Load Balancing Deployment

draft-lll-srv6ops-qp-aware-srv6-lb

Yisong Liu(China Mobile)

Changwang Lin(New H3C Technologies)

Jinming Li(China Mobile)

IETF 125

# Motivations

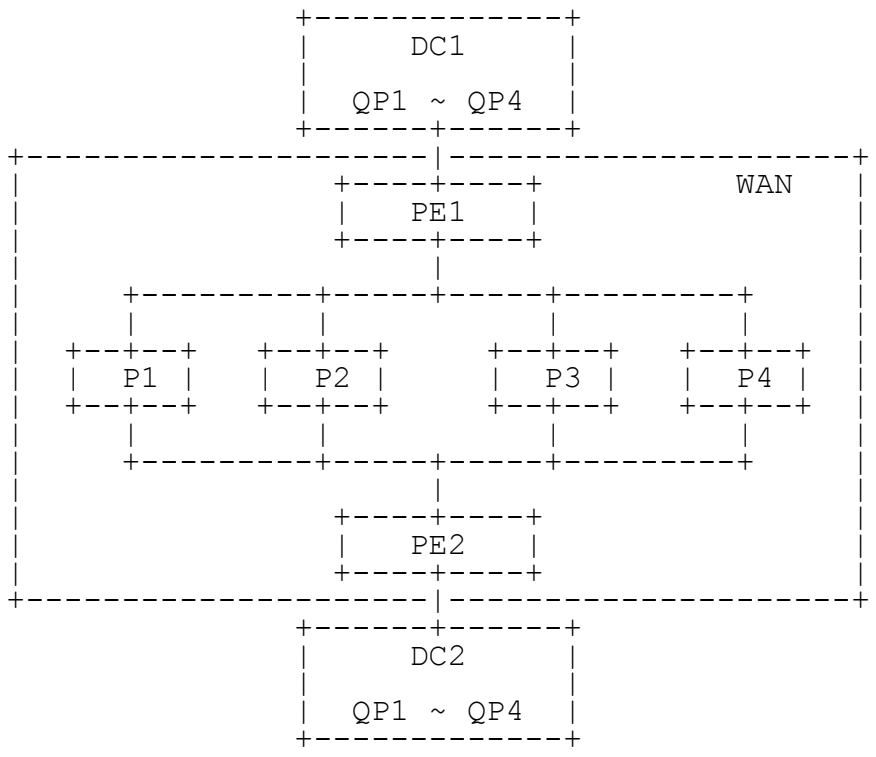
- **Traffic Characteristics of AI Clusters**
  - High-performance GPU-to-GPU communication based on RoCEv2/RDMA
  - Long-lived, high-throughput persistent connections
  - Relatively stable 5-tuple fields
  - Bound to long-term Queue Pair (QP) instances.
- **Challenges**
  - Link Hotspots and Congestion Caused by ECMP Hash Collisions
    - Due to fixed 5-tuple fields and the presence of AI "elephant flows," multiple similar large flows are hashed onto the same physical link, instantly triggering congestion.
  - Lack of Dynamic Adaptability
    - Traffic is "bound to long-term Queue Pair (QP) instances." Even when congestion is detected on a specific link, traditional ECMP cannot dynamically migrate existing long-lived flows to idle links.
  - Need for Finer-Grained Load Balancing
    - The industry has attempted to introduce finer-grained load balancing mechanisms, such as Packet-level load balancing. However, these approaches introduce complexities related to packet reordering

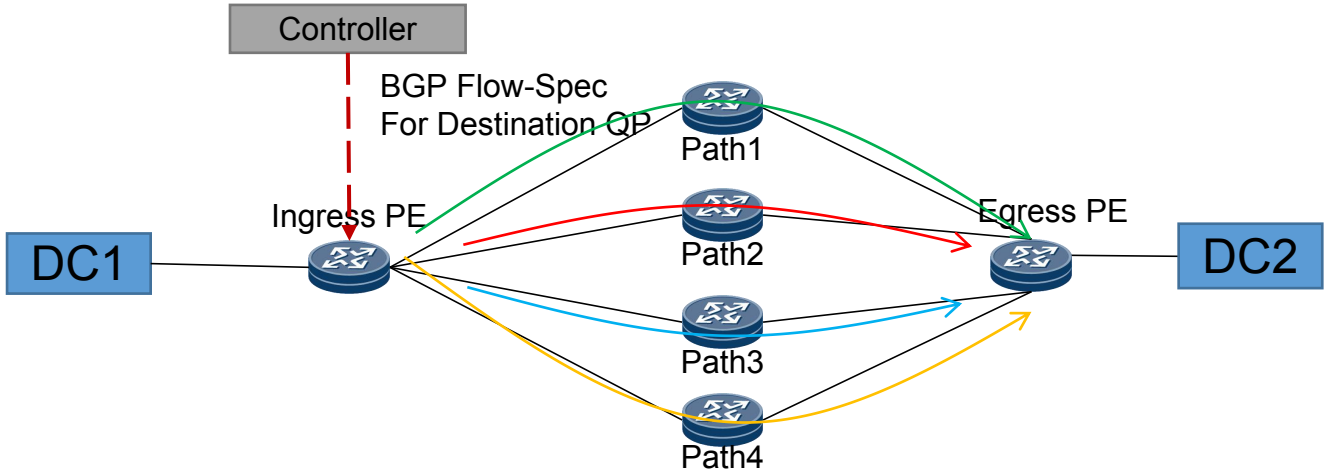
# QP-based SRv6

**Solution**:
As QP is the basic unit of RDMA communication, each connection has a unique QP ID. Even if multiple flows share similar IP addresses and ports, their QP IDs will always be different.
Leveraging this property, we can not only split large flows into finer-grained flows using QP, but also improve the performance of ECMP scheduling based on the combined dimension of "5-tuple + QP", while effectively reducing the dependence on randomization at terminal network cards.

- **STEP1. QP-to-Policy Mapping**
  - The ingress device parses the RoCEv2 packet header (BTH) and extracts the destination QP ID.
  - Map the traffic to a specific SRv6 policy according to the QP ID.
- **STEP2. Enhanced Hash-Based SL Scheduling**
  - Within the selected SRv6 policy (which may contain multiple candidate paths or Segment Lists), an enhanced hash algorithm takes the QP as input to perform deterministic fine-grained load balancing and select a specific Segment List.

# use case




**1.Generate BGP-FS rules that include Destination-QP**

BGP-FS Route:

FS Filters:

Destination: 203.0.113.0/24
source address: 198.51.100.0/24
protocol: UDP
destination port: 4791
source port: 10001
Destination-QP value: QP1/QP2

FS Action:

Redirect Flow to Policy1

**2.Map QP ranges to SRv6 policies**

| QP Range | SRv6 Policy Name |
|---|---|
| QP1, QP2 | Policy 1 |
| QP3, QP4 | Policy 2 |

**3.Use 5-tuple + QP as the hash key to select paths:**

* QP1 Packet: SRv6 Policy 1 -> SL1 (PE1->P1->PE2)
* QP2 Packet: SRv6 Policy 1 -> SL2 (PE1->P2->PE2)
* QP3 Packet: SRv6 Policy 2 -> SL1 (PE1->P3->PE2)
* QP4 Packet: SRv6 Policy 2 -> SL2 (PE1->P4->PE2)

# Advantages

- **Finer-grained traffic steering**
  - Networks can achieve finer-grained traffic steering based on Queue Pair (QP) identifiers, enabling superior load balancing across Equal-Cost Multi-Path (ECMP) groups and enhanced isolation between communication streams.
- **From random distribution to controlled scheduling**
  - Instead of only depending on random settings at end devices, this method gives network operators more precise traffic control.
  - It changes how we balance AI traffic from a passive “random spreading” using hash functions, to clear, direct control from the controller.
  - Besides, it builds the protocol basis for future QP scheduling that can adjust automatically based on real-time network congestion.

# Next Steps

- Absorb suggestions from the WG.
- Any questions or comments are Welcomed.

# Thanks