rtgwg
Internet-Draft
Intended status: Standards Track
Expires: 3 September 2026

D. Yuan
ZTE Corporation
Y. Zhang
China Unicom

2 March 2026

Hybrid-Function-Chain (HFC) Framework
draft-yuan-rtgwg-hfc-framework-02

Abstract

With the maturity of cloud native application development, applications can be decomposed into finer grained atomic services. On the other hand, as a distributed computing paradigm, fine grained micro-services could be deployed and implemented in a distributive way among edges to make computing, storage and run-time processing capabilities as close to users as possible to provide satisfied QoE. Under the circumstances analyzed, a Hybrid-Function-Chain (HFC) framework is proposed in this draft, aiming to wisely allocate and schedule resources and services in order to provide consistent end-to-end service provisioning.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 3 September 2026.

Copyright Notice

Table of Contents

1. Brief Introduction of HFC 2
2. Conventions Used in This Document 5
2.1. Abbreviations 5
2.2. Requirements Language 5
3. problem statement 5
3.1. problem 1: dynamically service instances management . . . 6
3.2. problem 2: service routing determination without network state awaring/Limitations of Current Service Scheduling . 6
3.3. problem 3: consistent end-end SLA guarantee 7
4. Hybrid-Function-Chain use case 7
4.1. case 1: Traffic lane used in RAG service chain over multiple clusters 7
4.2. case 2: Adaptive Link Scheduling Pipeline for Intelligent Video Enhancement Based on Agent Mesh and HFC 9
5. Hybrid-Function-Chain (HFC) Framework 11
5.1. Administration Plane 11
5.2. Control Plane 14
5.3. Forwarding Plane 15
6. SRv6-based HFC Implementation 16
7. Security Considerations 20
8. Acknowledgements 20
9. IANA Considerations 20
10. Normative References 20
Authors' Addresses 20

1. Brief Introduction of HFC

In the context of cloud native, applications are often no longer provided in the form of monolithic services, but are decomposed into a series of cloud native services deployed in distributed clusters, with inter-connections and joint provision to the outer side.

Traffic lanes, for instance, have emerged and been commonly used for environmental isolation and traffic control of the entire request chain of services for grayscale publishing scenarios, would be reckoned as a typical example for Hybrid-Function-Chain (HFC). In fact, the creation of traffic lanes is still executed by various existing network API configurations of the cluster. The service routes are always configured in the cluster and identified endpoints under a service name to implement various scheduling strategies and perform load balancing schemes among multiple optional instances.

Edge computing, as a distributed computing paradigm, the core idea is to make computing, storage and service processing as close to the clients and customers as possible to reduce latency, improve response speed and enhance data security. When applications are further deconstructed into atomic services as analyzed previously, service inter-connections MAY not only exist in adjacent clusters deployed in a same edge, but also happen with network paths connecting remote edge data centers. Thus, incremental requirements would be raised correspondingly. Relevant use cases and requirements are discussed in [I-D.huang-rtgwg-us-standalone-sid].

Correspondingly, this draft proposes a Hybrid Function Chain (HFC) architecture aimed at providing end-to-end and consistent service provisioning capabilities which includes multiple service endpoints and corresponding connected network paths. Compared to conventional schemes and patterns, HFC is granted with multiple features and connotations.

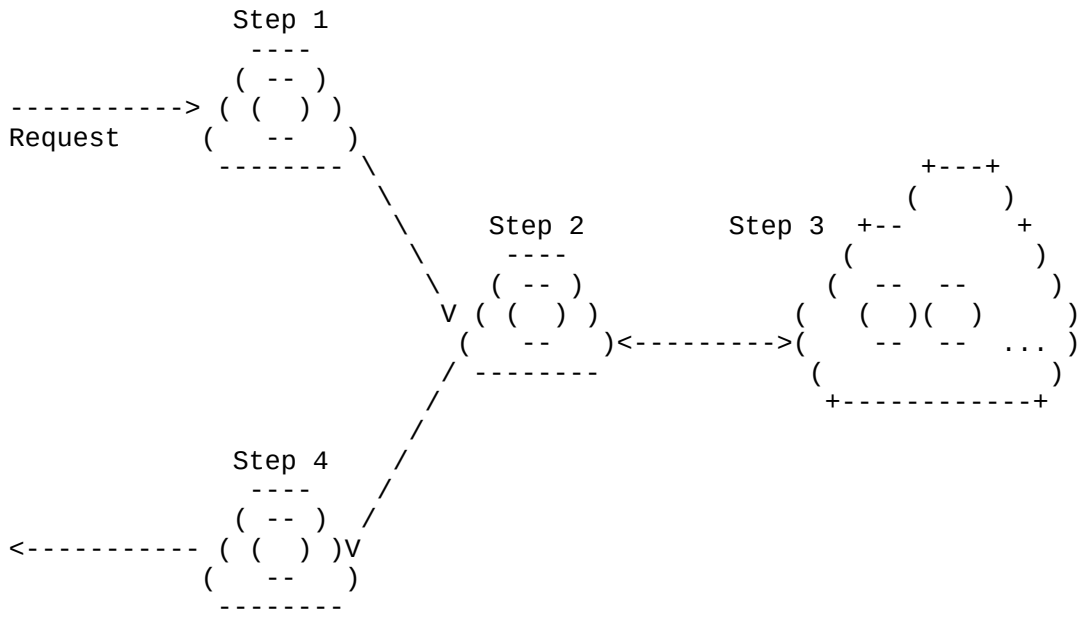


Figure 1: HFC across Multi-edges

* Hybrid service types and distributed forms: Considering the deployment phase, services and application functions can be deployed in one or multiple clusters in the form of containers, or deployed based on virtual machines. Service instances can be deployed with multiple instances with dynamic implementation and released based on a Serverless framework. Based on the run-time state, microservices and atomic functions form a diverse set of external services, and correspondingly, often raise various requirements for the resources and network capabilities. Compared to conventional Service Function Chain (SFC), the service functions targeted and discussed in HFC contains functions from both underlay network and application (L7) services.

* Hybrid inter-connections between service instances: The inter-connection, interaction, and collaboration schemes between upstream and downstream functions are always allocated and implemented within various forms. For instance, upstream functions MAY propose unidirectional notifications. Bidirectional requests and responses MAY also be observed between upstream function and single or multiple downstream functions. Multiple inter-connection forms MAY be implemented simultaneously in an overall HFC.

* Hybrid stacks and techniques: For conventional SFC in which Firewalls and Accelerators MAY be included, service functions are not tended and deployed to terminate data packets. However, for HFC functions, packets and payloads are always terminated at endpoints and payloads would be reorganized and regenerated. The provisioning of HFC process requires collaboration among multiple techniques and extends across TCP/IP stacks.

Based on the concepts of HFC proposed here, this draft further analyzes HFC framework and deconstructs it into several planes with incremental functions and features based on conventional network and service mesh techniques.

2. Conventions Used in This Document

2.1. Abbreviations

1. Terminology

* HFC: Hybrid Function Chain.

* SFC: Service Function Chain.

* QoE: Quality of Experience.

* SLA: Service Level Agreement.

* SRv6: Segment Routing over IPv6.

* OAM: Operation, Administration and Maintenance.

* APM: Application Performance Management.

2.2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. problem statement

3.1. problem 1: dynamically service instances management

In cloud-native architectures, atomic service capabilities are typically deployed as virtualized applications across distributed cloud nodes, managed via container orchestration systems. Service Mesh has emerged as a predominant framework for distributed service management. In standard deployments, the Service Mesh control plane maintains visibility of service instances across multi-domain environments by storing and managing stateful records of Endpoint IP addresses. However, in cloud-native and edge computing scenarios, service instances representing atomic capabilities are dynamically instantiated and terminated across a vast number of edge nodes. This volatile behavior imposes significant overhead on the management plane. Every lifecycle event of a service instance (registration or deregistration) necessitates frequent updates to the centralized or distributed service registry. Furthermore, for the execution of Service Chaining at the application layer, if atomic services are concatenated solely based on IP identifiers, the Service Mesh infrastructure must guarantee continuous accuracy of these dynamic Endpoints. Given the potential for millions of concurrent, short-lived microservice events across the network, the resource requirements for maintaining such a high-frequency update state become a critical bottleneck for server-side management entities.

3.2. problem 2: service routing determination without network state awaring/Limitations of Current Service Scheduling

Current microservice scheduling mechanisms, particularly in cross-cluster scenarios (e.g., Service Mesh architectures), primarily rely on the Control Plane to determine service routing for endpoints. These routing decisions are typically enforced through static configurations of target services across clusters via network APIs. However, a significant gap exists because the Service Mesh Control Plane lacks real-time visibility into the underlying network connectivity states between distributed services. In environments with redundant service instances, the originating endpoint remains unaware of potential differentiated service chains that may exist in the data plane. Consequently, the system cannot leverage optimized scheduling heuristics to select the most efficient path. This lack of dynamic, network-aware path selection prevents the infrastructure from providing deterministic end-to-end Service Level Agreement (SLA) guarantees.

3.3. problem 3: consistent end-end SLA guarantee

In traditional Layer 3 (L3) Service Function Chaining (SFC), packets traversing various Service Function (SF) endpoints are processed without terminating the underlying transport protocol. Consequently, the essential attributes and metadata of the packet are preserved across the chain, allowing for the consistent enforcement of the original steering and scheduling policies. Conversely, in Layer 7 (L7) service chain processing, the application-layer proxy or service instance terminates the incoming packet entirely upon reception. Following the execution of local service logic, a new packet is generated and encapsulated for transmission to the subsequent service node. This termination results in the loss of original flow context and scheduling metadata. As a result, the initial scheduling policy cannot be transparently reapplied to the newly instantiated packet, leading to a fragmentation of the control logic. This architectural limitation manifests as a lack of cohesive, end-to-end scheduling consistency across the entire service path.

4. Hybrid-Function-Chain use case

4.1. case 1: Traffic lane used in RAG service chain over multiple clusters

In an enterprise-grade global intelligent Q&A platform, the core RAG service is decomposed into three sequentially executed microservices deployed across geographically distributed data centers: Service A (Retrieval Service), Service B (Prompt Synthesis Service), and Service C (LLM Generation Service). These three services form a complete service chain A -> B -> C. User query requests have differentiated requirements for the end-to-end performance of this entire chain based on business type: real-time conversations demand ultra-low latency, while in-depth analysis tolerates higher latency but requires guaranteed bandwidth for large-volume data transfer.

Scenario Description: To meet these differentiated requirements, the operations team utilizes the HFC framework to define two global, end-to-end traffic lane strategies. Each strategy specifies a unified resource and network SLA preference for the entire three-node service chain A->B->C:

1) Low-Latency Chain Lane (low-latency-chain): Designed for real-time conversations. Policy Requirement: The end-to-end processing latency of the service chain A->B->C must be under 300 milliseconds. Therefore, this lane policy mandates: a) Service Instance Selection: Priority scheduling to lightweight instances of Services A, B, and C deployed in edge clusters closest to the user. b) Network Path Requirement: All inter-node network connections between Service A and Service B, and between Service B and Service C (whether cross-cluster or not), must use low-latency dedicated network links.

2) High-Throughput Chain Lane (high-throughput-chain): Designed for in-depth analysis. Policy Requirement: The service chain needs to process and transmit large volumes of retrieved context and generated content. Therefore, this lane policy mandates: a) Service Instance Selection: Can be scheduled to more powerful instances of Service B and Service C (possibly located in core data centers) capable of handling large contexts. b) Network Path Requirement: All network connections between Service A and B, and between B and C, are allocated high-bandwidth, high-throughput, cost-effective backbone network links to ensure efficient transmission of large data packets.

End-to-End Orchestration Flow under the HFC Framework:

1) Traffic Identification and Tagging: The API Gateway in City-A identifies the service type based on request characteristics (e.g., path or headers) and marks the request with the corresponding HFC lane identifier, e.g., x-hfc-chain-sla: low-latency.

2) Unified Path Calculation: The HFC Control Plane, upon receiving the tagged request, treats it as an integral chain requiring sequential execution of A->B->C. Based on the lane policy and real-time resource status, the control plane calculates a complete path covering all three service nodes and the network segments between them in one go. Every hop of this path (A instance, A-B network link, B instance, B-C network link, C instance) adheres to the SLA requirements of that lane.

3) Chained Path Encapsulation and Execution: The HFC Control Plane delivers the calculated complete path (likely encoded as an SRv6 Segment List containing multiple service SIDs and network SIDs) to the HFC Gateway in City-A. The gateway encapsulates this path into the outbound packet.

* Traffic is first directed to the lane-specified Service A instance.

* After Service A completes processing, its HFC proxy, following the path information in the packet, steers the traffic via the specified low-latency or high-bandwidth link to the Service B instance.

* After Service B completes processing, its HFC proxy similarly guides the traffic to the Service C instance via the specified next-hop link based on the path information.

4) Full-Chain Observability and Assurance: HFC's observability system tracks the request's entire journey through A ->(A-B network) - B -> (B-C network) -> C, collecting metrics at each stage to verify whether the entire chain meets the preset end-to-end SLA objectives.

In this manner, the HFC framework ensures that the complete path from the first service node to the last conforms to the network transmission quality required by the business, achieving fine-grained, SLA-aware global traffic governance for chained microservice architectures.

4.2. case 2: Adaptive Link Scheduling Pipeline for Intelligent Video Enhancement Based on Agent Mesh and HFC

In a cloud-edge collaborative video content enhancement platform driven by an Agent Mesh (intelligent agent collaboration network), the core processing workflow is designed as a fixed-sequence serial service chain, collaboratively executed by multiple intelligent agents, consisting of three processing nodes:

* Node A: Video Preprocessing Agent (Responsible for decoding, denoising, basic quality analysis).

* Node B: Intelligent Enhancement Agent (Responsible for core algorithm processing, e.g., super-resolution, HDR).

* Node C: Post-processing and Packaging Agent (Responsible for composition, encoding, and output packaging).

These three nodes, deployed as independent agents in geographically distributed data centers (e.g., Node A at the edge, Node B in the regional cloud, Node C in the central cloud), are coordinated by the Agent Mesh to form a cross-cluster end-to-end processing chain A -> B -> C.

Core of the Scenario: The video stream must sequentially pass through agent nodes A, B, and C. However, for the two network transmission links - from Node A to Node B and from Node B to Node C - there are two options available for each: a High-Throughput link and a Low-

Latency link. The system's intelligence lies in the Agent Mesh comprehensively analyzing the real-time processing status reported by each agent, making link selection decisions, which are then accurately executed by the HFC framework. This enables dynamic scheduling of the most suitable network link for each "node-to-node" transmission segment.

Dynamic Scheduling Example:

1) Agent Sensing and Decision: The video stream enters Node A (Video Preprocessing Agent) for processing. This agent reports its analysis results (e.g., {frame_complexity: high, data_volume: large}) in real-time to the Agent Mesh. The Agent Mesh makes a comprehensive judgment and decides: the transmission from Node A to Node B needs to prioritize the stable transfer of large data volumes, with less emphasis on ultra-low latency. Therefore, the Agent Mesh issues an instruction via the HFC Administration Plane: "For segment A to B, use the High-Throughput link."

2) HFC Policy Execution: The HFC Control Plane receives this instruction. At the egress of Node A, it steers the traffic onto the High-Throughput backbone network instead of the low-latency private line, ensuring efficient delivery of massive intermediate data to Node B.

3) In-Chain Continuous Optimization: After Node B (Intelligent Enhancement Agent) completes processing, it reports its new status (e.g., {data_volume: medium, realtime_requirement: high}) to the system. Based on this, the Agent Mesh performs a new round of judgment: the transmission from Node B to Node C becomes critically dependent on low latency to guarantee the real-time nature of the final output. Consequently, the Agent Mesh issues a new instruction: "For segment B to C, switch to the Low-Latency link."

4) HFC Dynamic Reconfiguration: The HFC Control Plane responds to the instruction. At the egress of Node B, it dynamically switches the traffic path from the default link to the Low-Latency private line, ensuring rapid data arrival at Node C for final packaging.

Summary: In this scenario, the topology of the service chain (A->B->C) is fixed. However, the "road quality" (network SLA) on the links is intelligently decided by the Agent Mesh based on real-time sensing and dynamically scheduled by the HFC. Their collaboration implements a fine-grained management model of "static nodes, dynamic links", ensuring the end-to-end performance of the entire processing chain is always optimal.

5. Hybrid-Function-Chain (HFC) Framework

Hybrid-Function-Chain (HFC) framework generally includes administration plane, control plane and forwarding plane. Based on conventional framework of network and cloud native practices, several incremental functions and features are added and deployed.

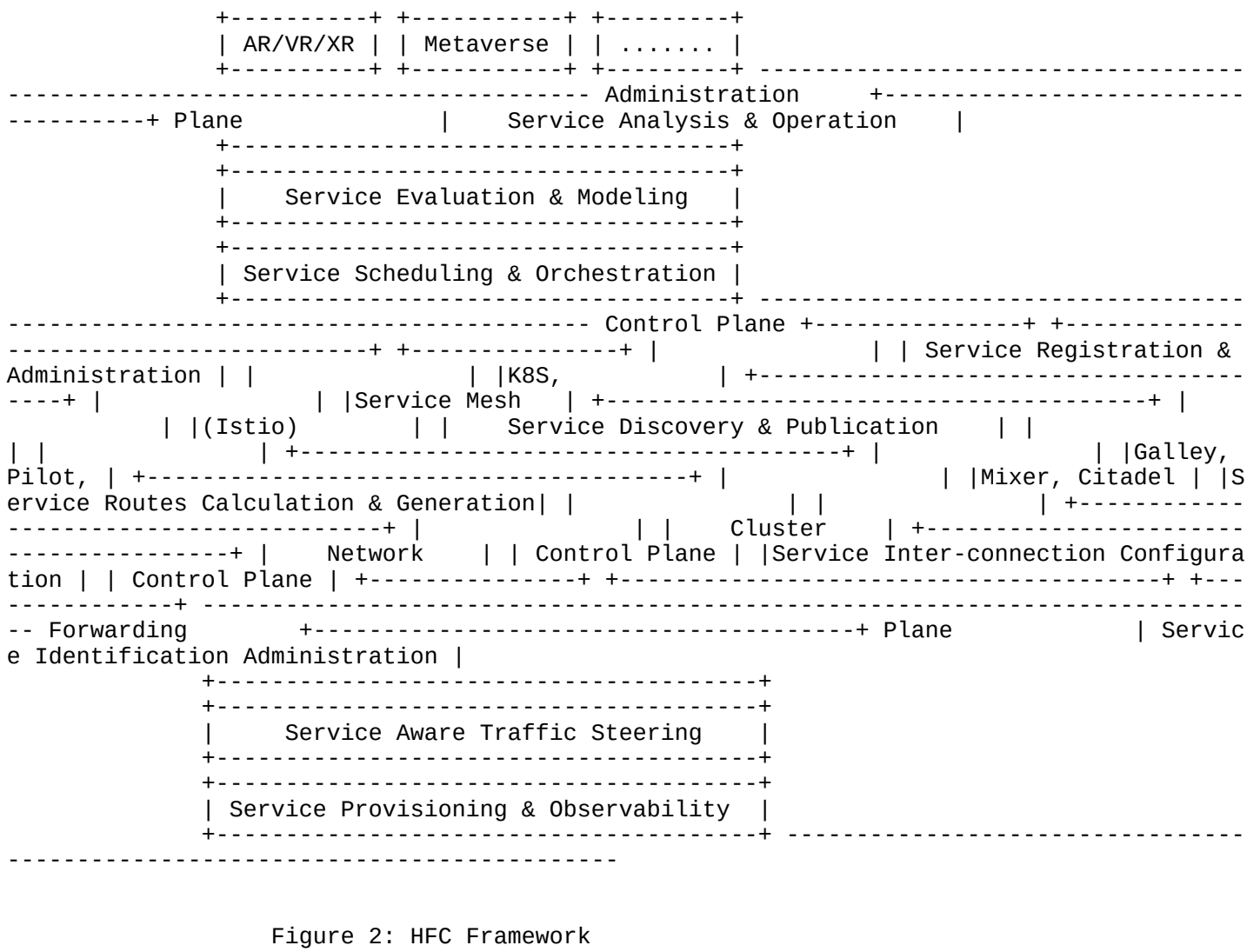


Figure 2: HFC Framework

5.1. Administration Plane

Service Analysis and Operation: The increasingly complex and diverse applications and services display different characteristics and features for outer users. In terms of orchestration for distributed micro-services, Service Analysis and Operation interprets the external and internal forms of overall applications and services as corresponding deconstruction patterns.

* Deep learning: The overall deep learning process would be decomposed into several successive or related phases and steps, data pre-processing, model training, prediction and estimation, model evaluation based on rewards functions, data storage and API interactions for instance.

* Live Broadcast: Relevant micro-services MAY include user

authentication, live stream administration, live recording, online payment and data migration.

The above deconstructed microservices have serial, parallel, unidirectional, and bidirectional relationships, and their interconnection and collaboration are comprehensively presented as user and outer oriented applications.

Service Evaluation and Modeling: There are different and various resource requirements raised by multiple microservices, including but not limited to:

* Computing resources and network capabilities: Computing-related services MAY be sensitive to computing resources, related indicators include CPU cores, available memories, floating number calculation. On the other hand, large amount of data transmission MAY be implemented between upstream and downstream services. Thus, the network connecting them would have to reserve sufficient bandwidth and provide abilities of low packet loss rate.

* Constraints for inter-connection patterns: the inter-connection patterns between upstream and downstream services and functions MAY be classified as unidirectional, bidirectional, one-to-one and one-to-many. Furthermore, due to security concerns for instance, relevant services MAY be deployed at adjacent endpoints or a same edge center geographically.

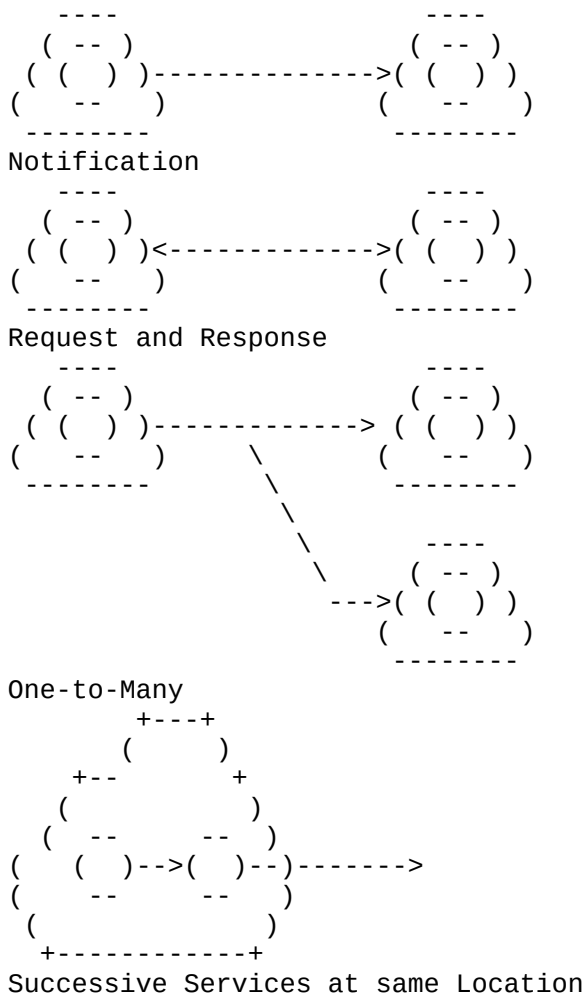


Figure 3: Inter-Connection between Services and Functions

Service Orchestration and Scheduling: Service administration would customize strategies or specific algorithms depending on circumstances of infrastructure and required proposals. Providing low-latency experiences or achieving load balance among available instances and resources SHOULD be selected as specific inclinations for further scheduling.

5.2. Control Plane

Service Registration and Administration: Based on the results and conclusions analyzed by Service Analysis and Operation, the overall service and included micro-services MAY be represented and administered by corresponding identifications. In this draft, Service IDs for micro-services and HFC IDs for HFC processes and services are generally defined. Therefore, HFCs and corresponding micro-services would be displayed and labeled in the control plane. Appropriate identifications would facilitate indicating the service traffic of the workflow.

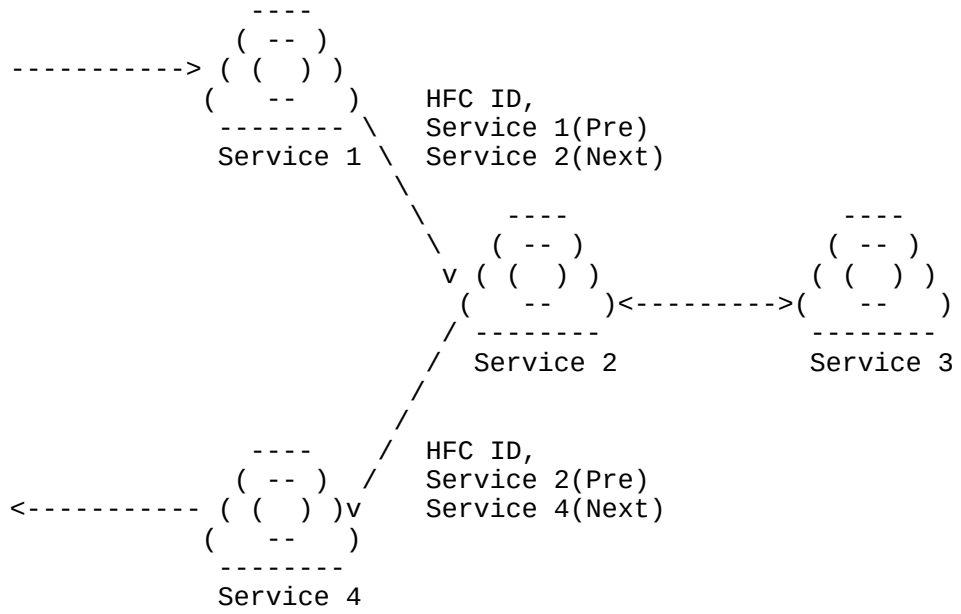


Figure 4: Service Administration by Identification

Service Discovery and Publication: Depending existing and mature control plane protocols and interfaces, distributed services and capabilities of infrastructures SHOULD be able to be collected. Relevant schemes include extended IGP, BGP, BGP-LS, RESTful, Telemetry. The information learned in the control plane MAY include:

* Computing resources related to services of specific instances.

* Deployment of service instances or possible and scheduled resources utilization.

* Network topology and corresponding TE capabilities.

Service Routes Calculation and Generation: Based on the information collected in Service Discovery and Publication, service routes would be calculated to determine appropriate instances and forwarding paths. Service Routes Calculation and Generation SHOULD follow the intentions identified in Service Orchestration and Scheduling. According to Service Registration and Administration, service routes could be distributed and indexed by HFC and service identifications.

Service Inter-connection Configuration: Within conventional schemes for services inter-connection, configurations would be disposed for endpoints distributed in multiple clusters. Istio, for instance, replys APIs including ServiceEntry, VirtualService and DestinationRules to describe inter-connections and relevant principles. Considering the framework of HFC proposed in this draft, service routes would be translated into corresponding configurations issued to clusters for revising API files.

5.3. Forwarding Plane

Service Identification Administration: Traffic would be able to be steered according to identifications distributed from the control plane. Also, the service identifications would inherit and re-generate from the previous ones in the workflow. Proxies, sidecars or gateways SHOULD be able to administer the inheritance and renewal relationship. Suppose an HFC application includes Service ID 1, ID 2 and ID 3, an identifier of {HFC, Service ID 2} implies that the successive function is expected to be Service ID 3. Correspondingly, the identifier would be modified as {HFC, Service ID 3}.

Service Aware Forwarding: Service routes entries would be distributed from the control plane and involved entities and devices would perform traffic forwarding accordingly. Relevant entries include:

* Service aware forwarding entries for edges routers in which forwarding paths are indexed by HFC IDs and Service IDs.

* Service identification administration entries for sidecars, proxies and gateways in which inheritance and correlations would be specified.

Service provisioning and observability: By implementing and performing OAM or APM schemes, forwarding plane would monitor the circumstances and performance of traffic flows. With detections of failures and possible degradations, forwarding plane would be able to support recovering, enhancing and provisioning for traffic flows.

6. SRv6-based HFC Implementation

Based on SRv6, forwarding paths orchestrated for an end-to-end HFC service including specific implementations would be able to be encoded in an SRH, in order to achieve consistent service provisioning across multiple endpoints deployed in distributed clusters and even edge clouds.

The overall paths would be explicitly indicated in the Segment List or be generally displayed. Correspondingly, a strict mode and a loose mode are proposed in this draft.

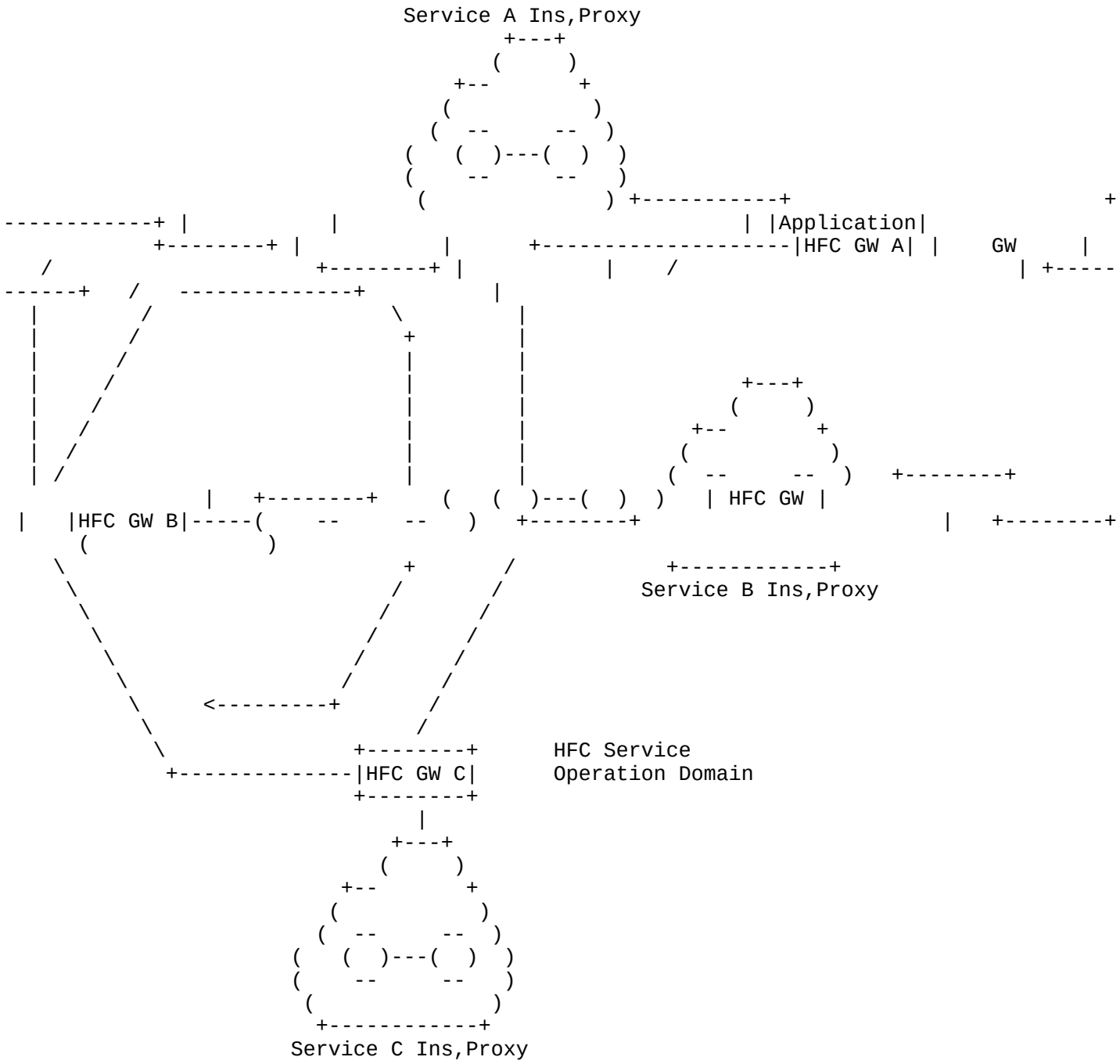


Figure 5: HFC Domain and end-to-end Delivery

* When a service request accesses a corresponding application gateway, the service is identified as an HFC service agreed and dealed in a previous contract. Suppose the HFC service is decomposed into micro-services A, B and C. The HFC service is named with an identification, HFC ID. To process and fulfill the

HFC service, the traffic would be steered to an access HFC gateway with packets carrying HFC ID and Service ID list. The endpoint for this HFC service would be configured as an SRv6 SID at the access gateway.

* The access gateway identifies the HFC ID and Service ID list carried in the packet according to the SID filled in the destination field. Indexed by HFC ID and next Service ID (Initial Service ID), an HFC policy would be determined. Segment List of the HFC policy would be encoded to the packet with an Insert or Encapsulation pattern. Under a strict HFC mode for instance, Service SIDs for each HFC gateway and binding SIDs for interconnected forwarding paths would be included in the Segment List. Afterwards, the traffic would be steered to next HFC gateway.

* With above displayed, HFC GW A is the first passing stand. HFC GW A would identify the next Service ID in the packet and implement a local lookup according to a Service SID. Therefore, the traffic would be steered into a local or adjacent edge cluster. Furthermore, HFC GW A MAY remove the SRHs and record the segment list and segment left in a local cache.

* When a flow aiming to a target service instance, a service pod for instance, it would be intercepted by a proxy or sidecar. The proxy or sidecar records the HFC ID and next Service ID in the packets and identifies returning traffic accordingly. Based on the HFC service lists, next service would be determined and its Service ID would overwrite the original one.

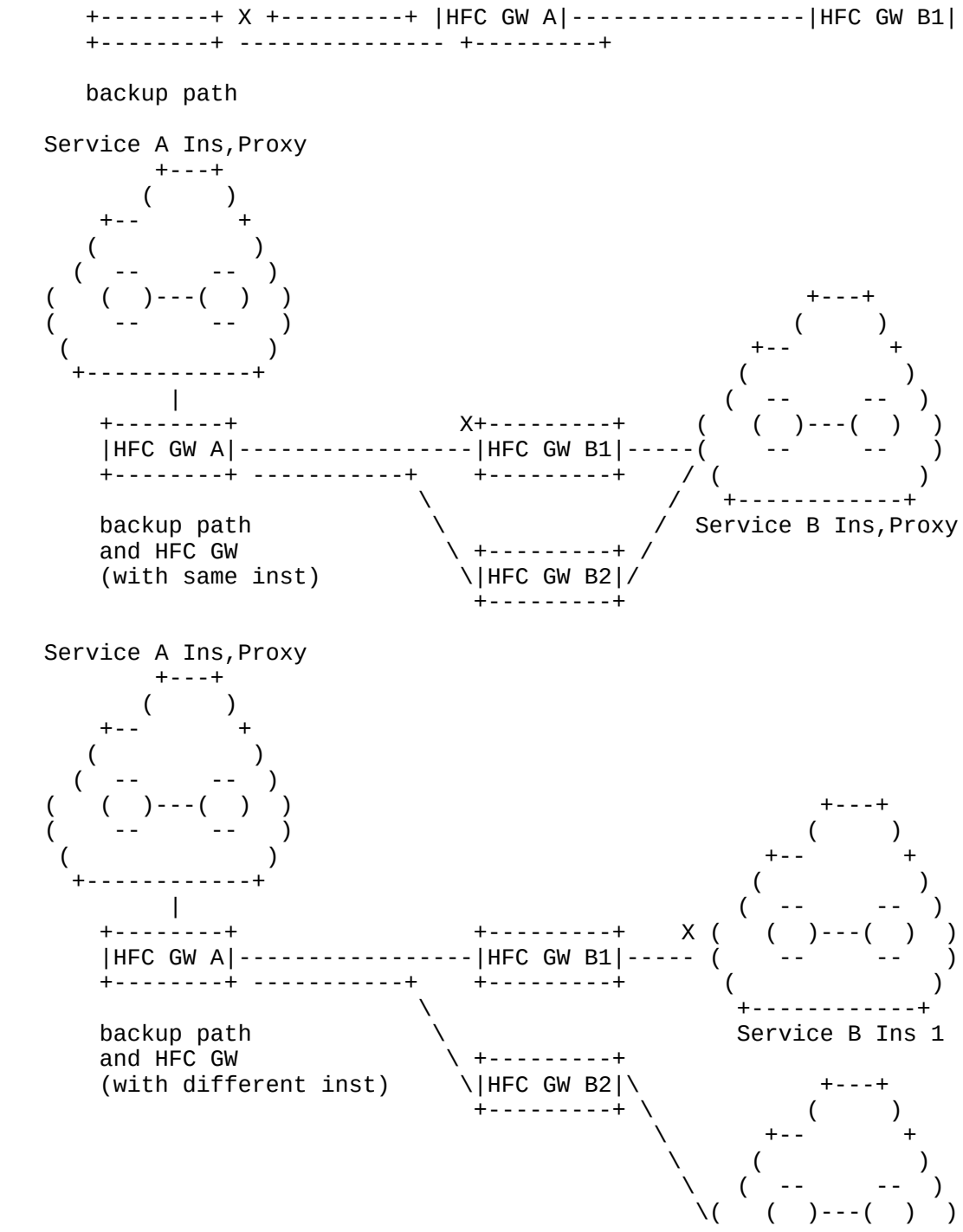
* Since the traffic flow returns to HFC GW A, an original segment list and relevant information would be re-attatched according to the records in the local cache. Afterwards, the traffic would be steered to next HFC GW similarly.

In the above SRv6 based HFC implementation, several SRv6 SIDs MAY be generally defined in this draft:

* HFC Service SID: correlates with an HFC service forwarding table indexed by HFC ID and next Service ID, aiming to determine a forwarding path indicated by segment lists.

* HFC Cache SID: correlates with an HFC local forwarding table and local cache table, aiming to record the forwarding information in the cache and forward the traffic to a local endpoint.

* HFC Inherit SID: correlated with an HFC local cache table, aiming to determine and re-attach a matched original forwarding path.


+--------+ X +---------+ |HFC GW A|-----------------|HFC GW B1|
+--------+ --------------- +---------+
backup path
Service A Ins,Proxy
|HFC GW A|
|HFC GW B1|
|HFC GW B2|
backup path
and HFC GW
(with same inst)
Service B Ins,Proxy
Service A Ins,Proxy
|HFC GW A|
|HFC GW B1|
|HFC GW B2|
backup path
and HFC GW
(with different inst)
Service B Ins 1

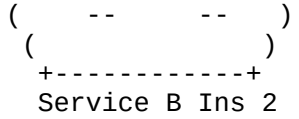


Figure 6: HFC Protection

Furthermore, SRv6 based HFC SHOULD support other features. For instance, backup paths would be orchestrated and accordingly configured at HFC GWs. End-to-end service observability would be achieved by distributed tracing and relevant schemes. More detailed implementation designs would be discussed in future works.

7. Security Considerations

To be discussed in future versions of this document.

8. Acknowledgements

TBA.

9. IANA Considerations

TBA.

10. Normative References

[I-D.huang-rtgwg-us-standalone-sid]
Huang, D., Liang, J., Yang, F., Zhang, Y., and D. Yang, "Use Cases-Standalone Service ID in Routing Network", Work in Progress, Internet-Draft, draft-huang-rtgwg-us-standalone-sid-01, 1 July 2024, <https://datatracker.ietf.org/doc/html/draft-huang-rtgwg-us-standalone-sid-01>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

Authors' Addresses

Dongyu Yuan
ZTE Corporation
Email: yuan.dongyu@zte.com.cn

Yan Zhang
China Unicom
Email: zhangy1156@chinaunicom.cn