CONTROL AND MANAGEMENT OF NETWORK SERVICE QUALITY BASED ON SRV6

XIAO YANG[1] ,WANG TAORAN[1] ,LIU PINLE[1]

[1]School of Information and SoftWare Engineering,University of Electronic Science and Technology of China , Chengdu,China
E-MAIL: 2022091203016.std.uestc.edu.cn, 2022090915024.std.uestc.edu.cn, 2022091602007.std.uestc.edu.cn

Abstract:

with the development of Internet and communication technology, Quality of Service(QoS) has become an important indicator to measure and manage network performance, in order to meet the needs of diversified network applications, solve the problem of network congestion and meet the needs of users. At the same time, SRv6, as an extended protocol based on IPv6, provides the possibility of innovation and development for the network architecture through flexible routing and service customization.P4, as a data plane programming language that has emerged in recent years, has excellent programmability and adaptability for designing high-performance network systems. In depth study of QoS and SRv6 will provide new ideas and solutions for network optimization, user experience improvement and network sustainable development.

Keywords:
QoS; SRv6;P4;gRPC;Network Application

1. Introduction

Network architectures that separate the control plane from the data plane have gained increasing popularity. This approach, commonly known as Software-Defined Networking (SDN)[6], offers a structured software environment for developing network-wide abstractions while potentially simplifying the data plane. By decoupling control and data, SDN enables centralized management and control of network resources, facilitating the implementation of network-wide policies and services.

A key motivation for adopting SDN is to address the limitations of traditional networks, specifically their inability to provide predictable network delay and jitter. Furthermore, with the ongoing evolution of network architecture, segment routing (SR) has garnered significant attention. SR is an architecture that can be deployed on the IPv6 data plane[1], and its implementation offers benefits for achieving a seamless network migration towards IPv6. By leveraging SR, network performance can be improved, and intelligent routing and customized services can be enabled.

In this context, our study focuses on SRv6 (Segment Routing with IPv6) to explore its potential in enhancing network performance, facilitating intelligent routing, and enabling the delivery of customized services. SRv6 extends the capabilities of SR by utilizing IPv6 addresses as routing instructions[2], providing a flexible and scalable solution for network routing and service implementation.

Overall, the growth of SDN and the exploration of SRv6 exemplify the industry's efforts to enhance network performance, enable intelligent routing, and support the deployment of customized services within modern network architectures.

2. Background and Related Work

In IP carrier networks, several challenges arise. One such challenge is the Island Problem, where despite the unification brought by MPLS, different domains like IP backbone networks, metropolitan area networks (MANs), and mobile carrier networks remain isolated from each other. This isolation necessitates the use of complex technologies such as cross-domain VPNs[3], resulting in intricate deployment of end-to-end services. Furthermore, congestion and resource allocation present another challenge. Traditional queuing mechanisms often employ tail-drop, which only detects congestion when the queue is completely filled. Proactive queue management algorithms and Quality of Service (QoS) scheduling algorithms are crucial to prevent congestion, minimize queuing delays, and allocate resources based on diverse business requirements. Lastly, challenges in network device access and IP technology extension emerge due to scalability issues with IPv4 and

compatibility problems with IPv6. The growth in device access led to the development of IPv6, but its incompatibility with IPv4 requires a network-wide upgrade, posing deployment challenges for applications. These challenges collectively highlight the complexities faced in IP carrier networks.

Therefore, we completed the following work. Based on the SRv6 protocol and IPv6 header extension, intelligent traffic scheduling is implemented in the IPv6 network device cluster, effectively improving the service quality of the IPv6 network cluster. The system ensures compatibility with ordinary IPv6 equipment, thereby enabling rapid service deployment and smooth evolution of network infrastructure. By distinguishing traffic types, applying appropriate policies based on the current state of the cluster, and calculating the most appropriate path for each traffic flow, our system solves network problems such as congestion, uneven resource allocation, and poor quality of experience for low-latency applications. This "best effort" strategy improves the overall network health of the entire cluster.

3. Design

For the design of this system, we chose to use gRPC to implement remote connections, which solves the challenge of network environment fluctuations by providing efficient data transmission and enhanced data security..

Consider that the implementation of network monitoring functions in actual network scenarios involves obtaining real-time data on link delays, bandwidth utilization, and other related metrics to support core system functions such as calculating the best path. In this study, we utilize the P4 programming language to implement in-band network telemetry (INT), an in-band network sensing technology. INT enables monitoring of link data by inserting INT headers and metadata (MD) into packets. When a data packet enters the INT system, the INT module mirrors the data packet, inserts the INT header and MD, and forwards it to the intermediate node. At the last hop, the switch matches the INT header, inserts the last MD, extracts the telemetry information, and forwards it to the telemetry server using mechanisms such as gRPC[4]. The telemetry server decodes the information and reports it to the upper telemetry application. Implementation involves defining INT-related headers in a P4 program, developing a parser for INT header parsing, and leveraging remote controllers such as the ONOS controller, which include pre-built applications that support INT. A packet capture using the Wireshark process is started on each node and the stored JSON file is updated every five seconds with the latest link information from the captured packets.

In our design, we use the IPv6 flow label field to distinguish traffic.To achieve scheduling for different types of traffic, it is essential to distinguish between various flows. The Flow Label is a reserved field in the IPv6 protocol. Following the pioneering work of the IETF in the IPv6 protocol, we use the flow label, source address, and destination address as a triplet to identify flows. According to the standard of the Hybrid MI Flow Label scheme, we define the structure of the flow label representation as shown in the diagram below. In this representation, all zero positions in the flow label are used to indicate packets that have not been labeled.

Figure 1. the flow label

The first three bits	Definition
000	Default
001	Stream labels are defined using random numbers
010	Replace the flow label with the value in the hop to hop extension header
011	PHB identification number
100	Format with port number and protocol is used in flow labeling
101	A new definition of structure(MI)
110	reserve
111	reserve

Figure 2. Definition of first 3 digits of flow table

Within the context of our research, we explore the utilization of specific bit fields to differentiate traffic flows in terms of their bandwidth, buffer requirements, real-time tolerance, and delay requirements. The bandwidth bit field consists of six bits, where the first bit indicates the minimum bandwidth requirement (0) or the maximum bandwidth requirement (1), and the remaining five bits represent the corresponding bandwidth values. Similarly, the buffer requirement tag employs five bits to designate the buffer size, with each bit value corresponding to a predefined buffer value. Additionally, the last five bits pertain to the tolerable delay standard value for communication flows, where each bit value signifies a predefined delay threshold.Considering our particular application scenario, we currently focus solely on the bandwidth requirement, real-time tolerance bit, and delay requirement fields to distinguish flows. For traffic flows necessitating low-latency conditions, such as video streaming, we adopt a strategy where the real-time tolerance position is set to 1. Subsequently, the first bit of the bandwidth value is set to 1 to indicate the minimum bandwidth requirement. Finally, we specify the precise values for both the delay and bandwidth requirements to meet the desired objectives.To distinguish traffic flows, we

leverage the Flow label field present in the IPv6 header. As the native IPv6 protocol lacks strict guidelines for the utilization of this field, we adopt the hybrid-MI flow label solution, which enjoys broader acceptance. In our approach, we define and parse the IPv6 protocol header within the P4 code, without introducing additional modifications. We then proceed by rewriting the host code in Mininet, converting the mininet host into an IPv6 host class capable of accommodating special IPv6 protocols. Subsequently, we incorporate code within the class to enable the switch to modify the flow label when transmitting IPv6 datagrams.In accordance with the hybrid MI flow labeling scheme, for traffic flows with stringent low-latency requirements like video streams, we set the real-time tolerance bit to 1. Accordingly, we assign a value of 1 to the first bit of the bandwidth field to denote the minimum bandwidth requirement. Finally, we specify the exact values for the required delay time and bandwidth. Conversely, for HTTP streams transmitted using the TCP protocol[5], we set the real-time tolerance bit to 0 as per the established scheme.Intelligent path selection for traffic classificationIn order to achieve effective utilization of network resources, different path selection modes are implemented according to the type of flow. Intelligent path selection takes into account the following metrics: bandwidth, latency, traffic. And give it different weights according to different modes. Suppose there is a path P, including n links l_1, l_2, . . . , ln, then the following method is considered for its measurement.

Our design implements delay-based path selection. Since delays are superimposed along links, the sum of link delays should be used for calculations:

$$f(p) = \sum_{i=1}^{n} delay(l_i) \qquad (1)$$

Among them, $delay(l_i)$ represents the delay of link l_i. For paths with high latency requirements, the requirements can be met by calculating $f(p)$ with the lowest latency.

Both delay and balancing modes can be achieved by calculating link weights and the shortest path. Suppose the non-negative weight graph $G =< V,E >$, V is the point set of all points in the graph, and E is the edge set of all edges in G. select

The source point is s, and the point set V is divided into a set S and a set Q. The elements of the set S are the points for which the shortest path has been confirmed, and the elements of the set Q are the points for which the shortest path has not been confirmed. The algorithm maintains $Q = V - S$. Dijkstra's algorithm first initializes $S = \phi$, $Q = G.v$, $dis(s) = 0$, and the rest $dis(v_i) = INF$, and then repeats the following operations:

1. From the set Q, select a node v_i with the smallest dis. Move v_i into the S collection.

2. Perform relaxation operations on all outgoing edges of v_i.

Repeat the loop until . At this time, $dis(v_i)$ is the shortest distance from the source point s to v_i.

For bandwidth-based intelligent path selection. On a path, the maximum bandwidth depends on the link minimum bandwidth. That is,

$$f(p) = min\{band(l_i)\}, i = 1,2, \dots, n \qquad (2)$$

$band(l_i)$ represents the bandwidth of link l_i. For paths with high bandwidth requirements, the path with the highest calculated $f(p)$ meets the requirements. Path selection in bandwidth mode uses a method similar to Dijkstra's, but the node information update method changes. The specific method is as follows: first initialize $S = \phi$, $Q = G.v$, $dis(s) = 0$, and the remaining $flow(v_i) = 0$, then repeat the following operations:

1. From the set Q, select the one with the largest $flow(v_i)$ Node v_i. Move v_i into the S collection.

2. Perform bandwidth update operations on all outgoing edges of v_i. The bandwidth update operation is:

$$newlink = min(flow(v), band) \qquad (3)$$

, band is the bandwidth from node v to node w. If $newlink > flow(w)$, it means that the path to w can become larger by passing through node v. At this time, $flow(w)$ and the predecessor node of w are updated, $pre(w) = v$. Repeat the loop until $Q = \phi$. At this time, $flow(v_i)$ is the maximum bandwidth from source point s to v_i.

Since the routing of the SRv6 protocol is essentially based on IPv6 forwarding, and in actual applications the switch must also have other common forwarding functions, our system functions rely on the architectural foundation jointly developed by ONOS-P4.

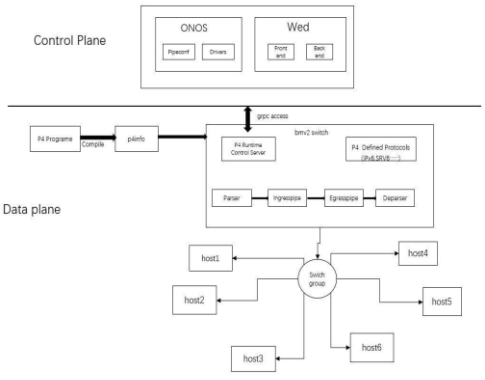

Figure 3. Architecture

4. Conclusions

As an upstart of the new generation network, SRv6 has many protocol advantages. SRv6 is based on IP reachability, making it easier to interconnect different network domains. SRv6 is based on native IPv6, making it easier to seamlessly integrate with applications and achieve seamless integration of cloud networks. In the current environment of the advent of the Internet of Things era, the popularization of 5G technology, and the wild growth of the cloud business industry, SRv6 has become a hot topic in the new generation of networks. P4 is also a hot topic technology in the current SDN development. It has stronger programmability than the old generation SDN represented by the OpenFlow protocol, and also provides strong support for starting the next generation of SDN technology era.

Acknowledgements

This paper is supported by University of Electronic Science and Technology of China, and the IEEE Systems.

References

[1] S.Previdi et al., “Segment routing architecture”, RFC 8402: Segment Routing Architecture, https://www.rfc-editor.org/rfc/rfc8402.html, July 2018

[2] C. Filsfils, N. K. Nainar, C. Pignataro, J. C. Cardona and P. Francois, "The segment routing architecture", Proc. IEEE Global Commun. Conf. (GLOBECOM), pp. 1-6, 2015.

[3] E.Rosen and Y.Rekhter, ”BGP/MPLS VPNs”, RFC 2547: BGP/MPLS VPNs, https://www.rfc-editor.org/rfc/rfc2547.html, March 1999

[4] A.Adamson and N.williams, “Remote Procedure Call (RPC) Security Version 3”, RFC 7861: Remote Procedure Call (RPC) Security Version 3, https://www.rfc-editor.org/rfc/rfc7861.html, November 2016

[5] T.Berners-Lee, R.Fielding, H.Frystyk, “Hypertext Transfer Protocol -- HTTP/1.0”, RFC 1945: Hypertext Transfer Protocol -- HTTP/1.0, https://www.rfc-editor.org/rfc/rfc1945.html, May 1996

[6] www.opennetworkingfoundation.org”