Technical Disclosure Commons

Defensive Publications Series

July 2021

'EVENT MESH' TRIGGERED METHOD FOR HYBRID CLOUD CHAINING VIA TUNNELING

Akram Sheriff

Dave Zacks

Tim Szigeti

Drew Pletcher

Nagendra Kumar Nainar

'EVENT MESH' TRIGGERED METHOD FOR HYBRID CLOUD CHAINING VIA TUNNELING

AUTHORS:
Akram Sheriff
Dave Zacks
Tim Szigeti
Drew Pletcher
Nagendra Kumar Nainar

ABSTRACT

The evolution of cloud native application architectures has yielded infrastructures (comprising microservices, etc.) entailing a variety of challenges. The need exists for an optimum 'Event Mesh' type of technique for sending data via asynchronous event handling mechanisms across cloud scale data center sites (e.g., in different regions) that is agnostic to any underlying facilities. Such a sharing of contextual data between clusters across hybrid cloud environments may be referred to as 'Hybrid Cloud Chaining' within the context of the techniques that are presented herein. To address the types of challenges that were described above, techniques are presented herein that provide an adaptive Event Mesh-driven method to identify an optimum traffic engineered path, through the use of Segment Routing over Internet Protocol (IP) version 6 (SRv6) elements, for performing 'cloud chaining' across clusters in a hybrid cloud environment.

DETAILED DESCRIPTION

The evolution of cloud native application architectures has yielded infrastructures that are built on top of microservices for decoupling traditional monolithic applications into multiple functional service components such as, for example, a single sign-on (SSO) or authentication service, a transaction check service, a backend service, an inventory service, etc. With a service-oriented architecture (SOA) an application developer still has the challenge of removing the communications issues between the SOA services. Otherwise, the application developers would lose their agility as they must add network-centric communication code. While microservices development is dominated by a Hypertext Transfer Protocol (HTTP) representational state transfer (REST or RESTful)

paradigm, SOA developers use a variety of solutions to address the network-centric issues that are associated with distributed message services including, for example, web services based on the Web Services Description Language (WSDL) and Simple Object Access Protocol (SOAP), RESTful HTTP, Extensible Markup Language (XML) remote procedure calls (RPCs), and message brokers. The extensive use of web services and the inherent orchestration complexity based on technology such as Business Process Execution Language (BPEL) has pushed too much functionality into the network component (e.g., middleware, process flow software, etc.) causing many kinds of inter-process communication (IPC) and interconnecting services problems.

Traditionally, in a client-server web application architecture, a solution to these types of issues involves the use of a message queue broker. However, in a cloud native application architecture the use of publish-subscribe-based event brokers have aided applications that are built in the same site in the physical infrastructure but not across different physical infrastructures. There are still issues, as discussed below, for which a service mesh (SM) or a non-service mesh (NSM) is unable to handle for an event-triggered microservices-based application architecture in such an infrastructure environment. The issues that were referred to above include, for example:

- A lack of support for asynchronous event or stream processing.
- Most traffic shaping and network services only apply to synchronous request-reply message exchange patterns and HTTP transport.
- Being limited to connection-oriented routing and the targeting of transport connections, not the routing of actual data.
- A service mesh is currently mostly available in Kubernetes (or K8) clusters, including variants, because it is the Kubernetes network abstractions that make it feasible to implement east-west routing.
- Microservices that use raw Transmission Control Protocol (TCP) network connections will only benefit from Transport Layer Security (TLS) between sidecars and the telemetry data that is gathered by the sidecars.
- A service mesh cannot be used in hybrid cloud environments where an integration or movement of contextual data from a Kubernetes cluster to a Non-

K8 based application container orchestration cluster is required for traffic flow in an east-west direction.

This mandates the need for an optimum ‘Event Mesh’ type of technique for sending data via asynchronous event handling mechanisms across cloud scale data center sites (in, for example, different regions) which is agnostic to any underlying facilities (such as, for example, distributed message queuing bus architectures like the Advanced Message Queuing Protocol (AMQP), Apache Kafka, Apache pulsar, Knative, etc.). This sharing of contextual data between K8 clusters or between a K8 and a non-K8 cluster across hybrid cloud environments may be referred to as ‘Hybrid Cloud Chaining’ within the context of the techniques that are presented herein.

To address the types of challenges that were described above, techniques are presented herein that support an Event Mesh-based tunneling workflow that allows for the exchange of data between K8 clusters. The same tunneling technique also supports the integration of K8 and a non-K8 environments via an Event Mesh.

Aspects of the techniques presented herein propose novel workflows for interconnecting cloud scale data centers across different sites through a Primary-Secondary tunneling interface by using an Event Mesh-based technique in hybrid cloud architectures. When connecting Kubernetes clusters to a multi-cloud management system through Kubernetes API hosting the main technical challenge lies in establishing cluster tunnels through optimum traffic engineered paths across regions.

Figure 1, below, illustrates elements of a multi-cloud Event Mesh, as described above, according to aspects of the techniques presented herein.

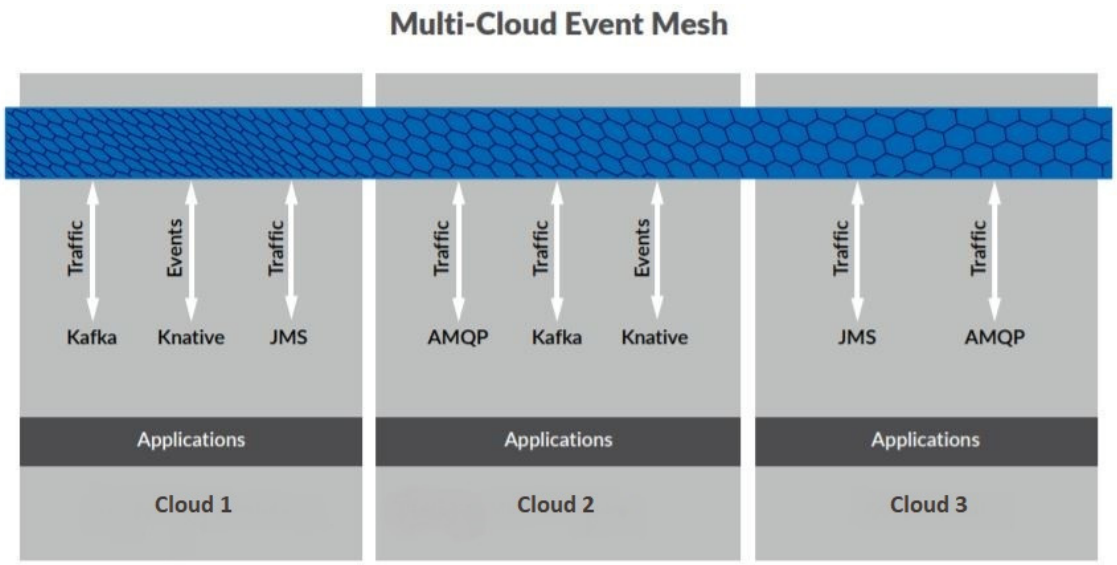


Figure 1: Illustrative Event Mesh

Figure 2, below, depicts various cloud scale data center interconnections, as described above, according to aspects of the techniques presented herein.

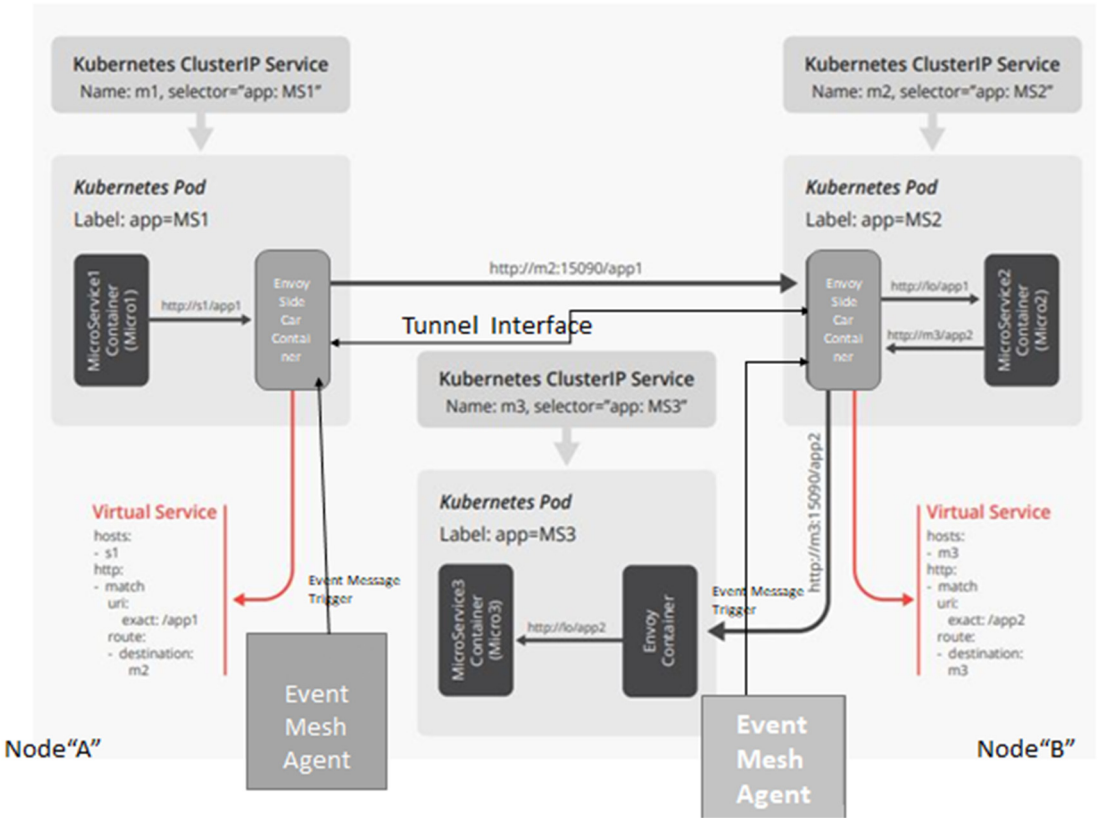
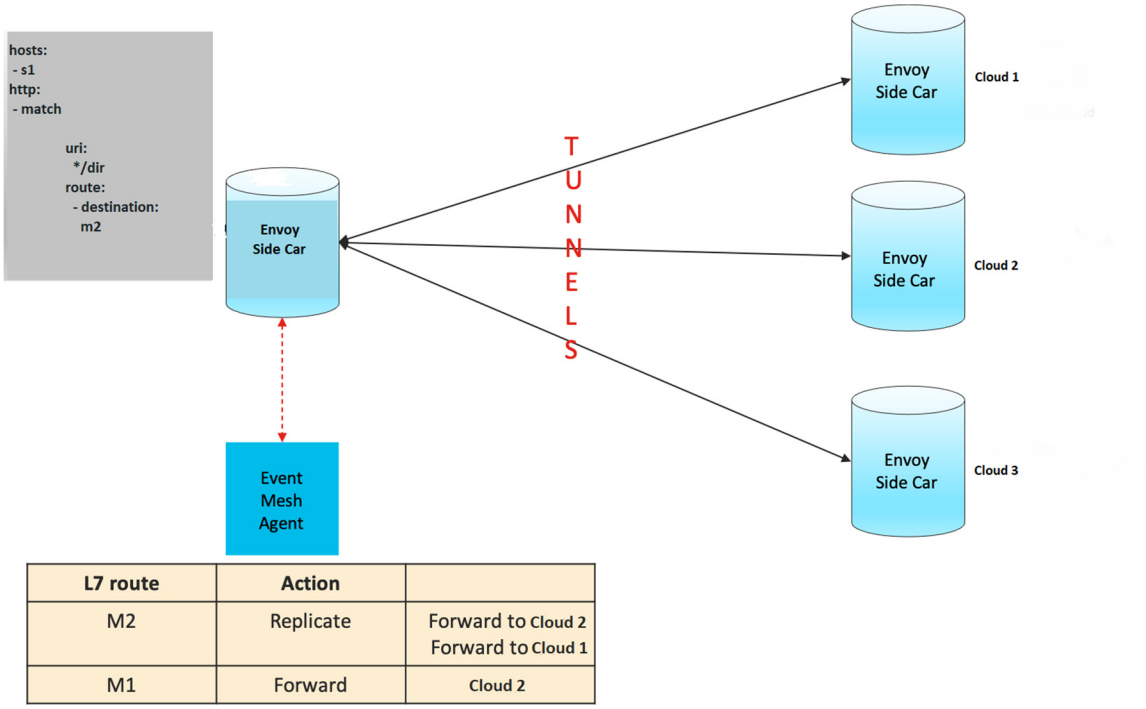


Figure 2: Exemplary Interconnections

Figure 3, below, illustrates aspects of the techniques presented herein and will be described in the narrative that is presented below.



L7 route	Action	
M2	Replicate	Forward to Cloud 2 Forward to Cloud 1
M1	Forward	Cloud 2

Figure 3: Illustrative Event Mesh Agent

As depicted in Figure 3, above, a virtual service lookup will result in the forwarding of an incoming request to M2 where M2 is another service that is hosted both on Cloud 1 and Cloud 2. The Event Mesh Agent is augmented with performance awareness that dynamically modifies the local table based on the performance.

As illustrated in Figure 3, above, initially the local table may identify replicating and forwarding the request to both of the instances that are running on Cloud 1 and Cloud 2. Further, additional details may be included in the overlay header (such as, for example, Segment Routing over Internet Protocol (IP) version 6 (SRv6)) to differentiate which request is the primary request and which one is the backup request. The requests may be differentiated using a flag in the header, a flag and id, a tag-length-value (TLV), etc. The inbound response is used to dynamically compute the performance and modify the table for optimal Event Mesh forwarding between the instances.

Under to aspects of the techniques presented herein an Event Mesh Agent may operate a sidecar container in the Layer 2 (L2) or Layer 3 (L3) layer alongside a network security manager to create the optimum egress route for microservice application packets to be routed to another cluster in a different cloud data center (either K8 or non-K8). It is important to note that an Event Mesh Agent may employ SRv6 or other on-demand routing protocols.

Under further aspects of the techniques presented herein a dynamic secured tunnel may be created across the distributed cloud scale data center regions to use the Event Mesh acting as tunneling agents with different tunnel identifier attributes (e.g., 128 bit or 256 bit) and a time-bound token.

SRv6 allows a source node to steer a packet through an ordered list of segments that are encoded as IPv6 addresses. Such a list is stored or embedded inside a new Segment Routing Header (SRH) which is part of the IPv6 header. Every segment is associated with a Virtual Network Function (VNF) that is placed at a specific location in the network. A function could for instance be a VNF sitting in a different public cloud across different sites or regions. Aspects of the techniques presented herein leverage the SRv6 header's embedded list of segments and apply an Exclusive-OR (XOR) operation among the segments to yield a static seed that may be used in the encryption process of the tunnel that is created across K8 clusters.

Under further aspects of the techniques presented herein, as illustrated in Figure 3, above, the primary and secondary tunnels that are used for the replication of traffic may be selected by the Event Mesh Agent that is interconnected to the Envoy proxy. For example, SRv6 elements may be leveraged to create optimum primary and secondary tunnels for traffic replication with those particulars then conveyed to the interconnected Envoy proxy in a multi-cloud architecture.

In summary, techniques have been presented herein that support an adaptive Event Mesh-driven method to identify the optimum traffic engineered path, through the use of SRv6 elements, for performing ‘cloud chaining’ across clusters in a hybrid cloud environment. The sharing of contextual data between clusters across hybrid cloud environments may be referred to as ‘Hybrid Cloud Chaining’ within the context of the presented techniques.