June 13, 2019

HIGH AVAILABILITY FOR CLUSTER OF STATEFUL SERVICES BEHIND LOAD BALANCER IN PRIVATE/PUBLIC CLOUD

Kent Leung

Louis Zhijun Liu

Jeremy Felix

Andrew Ossipov

Mohamed Mahmoud

HIGH AVAILABILITY FOR CLUSTER OF STATEFUL SERVICES BEHIND LOAD BALANCER IN PRIVATE/PUBLIC CLOUD

AUTHORS:
Kent Leung
Louis Zhijun Liu
Jeremy Felix
Andrew Ossipov
Mohamed Mahmoud

ABSTRACT

Techniques are described herein for using stateful availability to maintain a session when auto-scaling a stateful service instance cluster or when a service instance fails. This has many advantages over current approaches in which public cloud and Kubernetes deployments can only provide stateless availability due to a number of limitations when load balancing a cluster of service instances.

DETAILED DESCRIPTION

The industry lacks support for high availability when traffic is load-balanced to a cluster of stateful service instances (e.g., firewall (FW)) in a public cloud. Existing solutions are not stateful, and as such sessions are lost when auto-scaling or when the service instance fails. This is a generic issue that is prevalent in the public cloud but also in any use case when a load balancer changes the destination Internet Protocol (IP) address and/or port number. For example, the Kubernetes environment also has this issue because the service label can cause the destination IP address and port to change to a serving container instance.

A flow entering a public cloud application is typically destined to a Virtual IP (VIP) address that is owned by an edge Load Balancer (LB). The LB changes the destination IP (DIP) address to a service instance IP address (e.g., FW IP address). This affects the 5-tuple flow header and pins the flow to a particular service instance. When a service instance fails, the flow can theoretically be redirected by an LB to another service instance. However, the 5-tuple flow header is no longer generic, and the receiving service instance needs to determine the original flow to retrieve its state.

Flows can be identified in the flow tuple either as the DIP address set to the VIP address or the DIP address set to the FW IP address. The VIP address is a virtual IP address that is shared among the service instances. The FW IP address is the IP address of a service instance. It would be preferable for the DIP address to be set to the FW IP address as seen on the wire and expected by the user. Internally, the FW IP address is remapped to the VIP address. But only the FW IP address is exposed and the VIP address is hidden for internal operation.

Solutions are described herein for establishing a consistent way for any service instance to identify a flow for the purposes of retrieving its flow state to handle the flow statefully. A first solution is to rewrite the DIP address to the VIP address when the packet is received on an external network interface of a service instance. A second solution is to maintain a VIP address based flow tuple (key) to FW IP address (value) mapping to recover the original flow tuple.

Figure 1 below illustrates the first solution. Here, the flow is identified based on the tuple with a common VIP address in place of the DIP address. The actual DIP address (FW IP address) is stored in the flow state. As represented by the green box, flows received on any service (e.g., FW) instance are mapped to a common tuple by changing the DIP address to the VIP address. “Old” indicates a new flow created from recovered flow states. “New” indicates a new flow created from scratch (received flow).

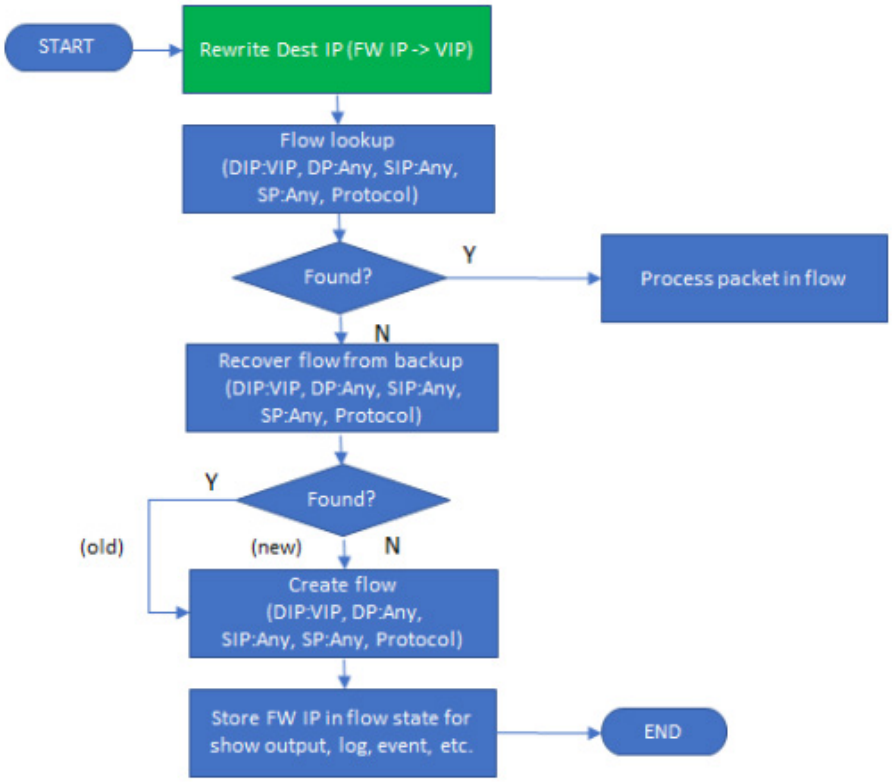

Figure 1

Figure 2 below illustrates the second solution. Flow mapping is based on a tuple with a common VIP address in place of the DIP address to obtain the original destination IP address (e.g., FW IP address for an old service instance). The original destination IP address is then used in the tuple for an original flow lookup to recover the flow state that populates the new flow state. The original service instance that had served the flow is "FW-O." The current service instance that received the flow is "FW-N." The flow mapping lookup is used to obtain the original DIP address which is in turn used to look up the original flow. "Old" indicates a new flow created from recovered flow states. "New" indicates a new flow created from scratch (i.e., received flow). A Distributed Database (DDB) may be used for the state lookups.

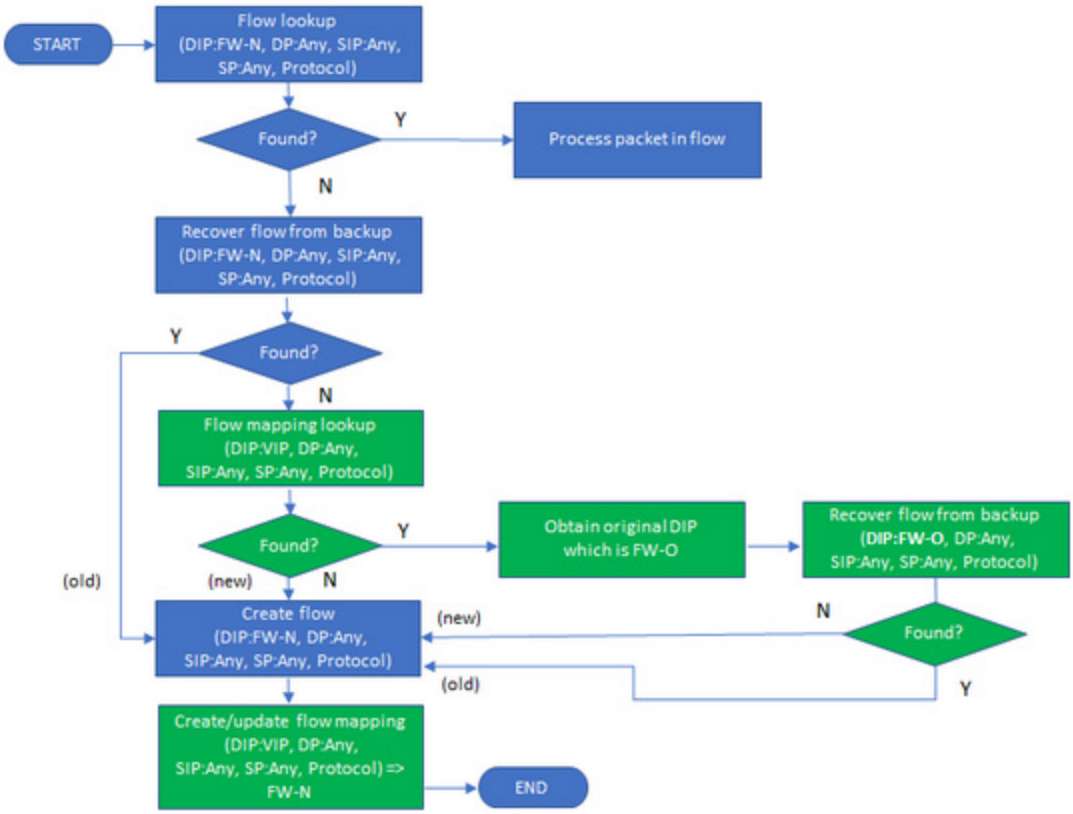

Figure 2

There are common issues shared by both public cloud and Kubernetes implementations. The Kubernetes service may be the common VIP address that is used to translate the destination IP address at the ingress point. This allows the security policy to be applied to the Kubernetes service object (or invariant IP address). This is similar to the public cloud where the VIP address is the internal LB to the application instances (e.g., part of a network security group).

The techniques described herein may involve mapping a flow to a common flow tuple regardless of the stateful service instance that originally processed the flow. This may achieve a consistent lookup criteria to enable stateful high availability for the flow. Even though a flow moves to another stateful service instance after an original service instance fails, a flow tuple is set to a value that can be used to look up the stored flow context and proceed with inspecting the packet statefully. This solution is designed specifically for stateful service instances (such as network security devices) that otherwise cannot scale with an external stateless load-balancer due to traffic asymmetry and potential loss of state/context.

The flow tuple remains the same regardless of which stateful service instance receives the flow for tracking and lookup purposes. For cloud native application networking, an LB will change the destination IP address to reach the backend servers or a service instance. This is the problem being addressed. When the destination IP address is changed to the receiving service instance, the change in the flow tuple prevents a stateful flow lookup for locating the stateful context of the existing flow.

When a stateful service instance fails, another stateful instance can retrieve a flow context/state using the consistent flow tuple. When the flow is statelessly load-balanced to another service instance, the destination IP address is changed. Thus, the destination IP address is rewritten to a common VIP address to derive a consistent flow tuple. A similar operation may be performed for Kubernetes in that the VIP address is the invariant IP address associated with the Kubernetes service label.

When a scaling event occurs, some flows may be statelessly directed to a new stateful service instance depending on the load balancing algorithm. This allows for processing of any flows that migrated by a stateful service instance through identification of the original flow and locations of its existing context/state using a consistent flow tuple. The backend target IP address is selected based on the backend service. This is typically an internal LB to web/application/database servers or Kubernetes service label associated with microservices in pods/containers.

The techniques described herein provide stateful availability when a cluster of stateful service instances are sandwiched between load balancers in the public cloud. These techniques may leverage threat defense and management software to provide an existing set of features for backward compatibility. Firewall clustering may be leveraged with enhanced Virtual Extensible Local Area Network (VXLAN) capability to enable a logical threat defense module formed by auto-scaling one or more threat defense Virtual Machines (VMs) in the cluster. Furthermore, an external LB and internal LB may be leveraged to forward traffic to the cluster. The VM scale sets may be leveraged to create and manage a group of identical, load balanced VMs used by the cluster. It will be appreciated that this solution is generally applicable to any public cloud provider that provides elastic load balancing and auto-scaling orchestration. In addition, these techniques may apply to auto-scaling a cluster of threat detection containers.

Furthermore, this scheme may apply to a private cloud as well as a public cloud. To enable private cloud deployment, the LB may be used in Layer 3 mode (e.g., a destination IP address may be sent to the backend server).

In summary, techniques are described herein to use stateful availability to maintain a session when auto-scaling a cluster or when a service instance fails. This has many advantages over current approaches in which public cloud and Kubernetes deployments can only provide stateless availability due to a number of limitations when load balancing a cluster of service instances.