March 2023

TECHNIQUES TO FACILITATE OBSERVABILITY-AWARE AND SELF-HEALING TRAFFIC ENGINEERING

Praveen Parthasarathy Iyengar

Maneesh Saxena

Pralhad Katti

Alhad Rajurwar

TECHNIQUES TO FACILITATE OBSERVABILITY-AWARE AND SELF-HEALING TRAFFIC ENGINEERING

AUTHORS:
Praveen Parthasarathy Iyengar
Maneesh Saxena
Pralhad Katti
Alhad Rajurwar

ABSTRACT

Many new applications/services/microservices are hosted in some form of a cloud network, such as a private, public, or hybrid cloud. This means the network connecting applications/services/microservices to clients is critical infrastructure that is needed to facilitate a good user-experience. However, some cloud network environments that utilize datacenter fabrics may have limited visibility with regard to observability and/or awareness of different network events (e.g., failures, service faults, potential performance optimizations, etc.). Provided herein are techniques through which Service-Level Agreement (SLA) decisions that may be made by third-party tools can be consumed in order to drive flow steering in a datacenter fabric utilizing local, hardware-assisted switch mechanisms. Such techniques may provide for the ability to replace/assist tool/controller interactions by offloading control plane changes for scenarios involving SLA violations, which may result in faster SLA resolution utilizing hardware-assisted switching.

DETAILED DESCRIPTION

Cloud networks (e.g., private clouds, public clouds, or hybrid clouds) can be utilized to host a variety of applications, services, microservices, and/or the like. Networks that interconnect such clouds to clients are a critical infrastructure that can impact user-experience. User-experience can be impacted by various reasons, such as application faults, intermediate validation service faults (e.g., firewall, Transmission Control Protocol (TCP) optimizer, etc.), network system faults, and/or the like.

There are many products or solutions that track Service-Level Agreement (SLA) adherence for networks, application infrastructure, application frameworks, etc. that can generate very impactful inferences with respect to reasons of failure. However, current

network systems are often incapable of assisting in temporary/permanent recovery to improve user-experience without a network controller. For example, network fabric systems are often not observable-aware.

In order to address such issues, techniques are presented herein through which SLA decisions that may be made by third-party tools can be consumed in order to drive flow steering in a datacenter fabric utilizing local, hardware-assisted switch mechanisms.

In particular, an SLA map can be created in which each node represents a single element in the network. The network element can be a physical switch, a virtual switch, a microservice, a set of microservices, or an element from the microservice architecture. Each of the nodes can be represented by varied number of vectors specific to the class of the node. A network node may have vectors such as switch uptime, port flap interval, bandwidth utilized, etc. In accordance with techniques herein, third-party observable metrics can be transposed in order to be represented in the SLA map.

Consider an example network topology, as shown below in Figure 1.

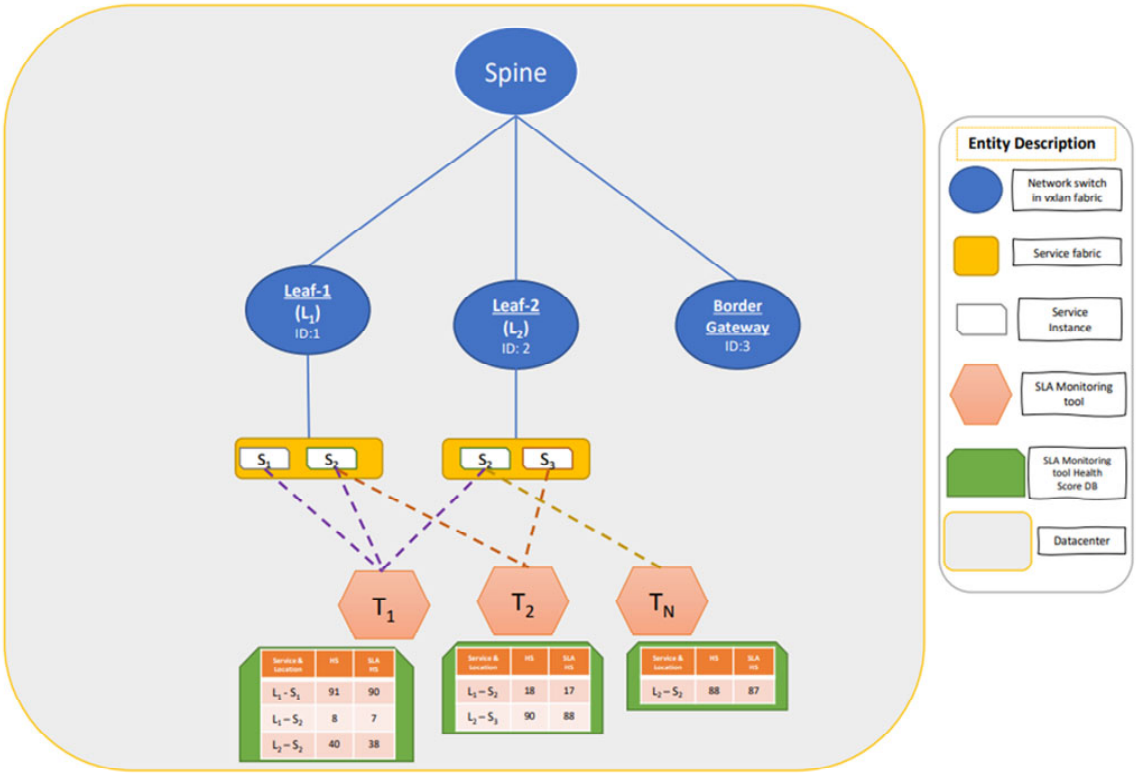

Figure 1: Example Network Topology

As illustrated in Figure 1, consider for the example network topology that there are two leaf nodes (L_1 and L_2) connected via a spine. Each of the leaf nodes is further connected

to two different service fabrics in which there are three services, labeled S_1, S_2, and S_3, in which S_2 is available in both service fabrics, whereas there is only one instance of S_1 and S_3. The services are monitored by multiple SLA monitoring tools, labeled, T_1, T_2, T_N in which T_1 monitors all services on leaf L_1 and S_2 on L_2, T_2 monitors S_2 on leaf L_1 and S_3 on leaf L_2, and T_N monitors S_3 on L_2.

A goal of the techniques of this proposal is to consume SLA decisions made by third-party tools in order to drive flow forwarding in the datacenter fabric. This is to replace/assist tool/controller interactions by offloading control plane changes for SLA violations for faster SLA resolution to a hardware-assisted switch local approach. Thus, this proposal provides techniques through which third-party SLA monitoring tools can utilize hardware assisted traffic forwarding modifications. Figure 2, below, illustrates three distinct actions or steps for a process flow through which techniques of this proposal may be utilized.

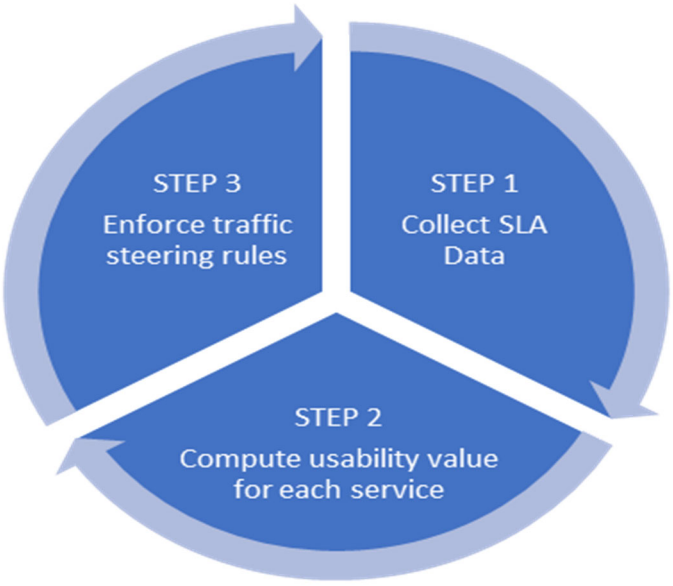

Figure 2: Proposed Process Flow

For the steps of the proposed process flow, it is assumed that each leaf is assigned a 30-bit identifier by the controller during fabric initialization. An existing leaf identifier may be used in some instances if it is unique for each leaf in the fabric. It is further assumed for the process flow that if a service instance is reachable from multiple leaf nodes that each leaf should have its own unique service instance configuration/representation.

Additionally, each leaf should be capable of maintaining a cache for leaf identifier and leaf information needed to forward traffic. Further, it is assumed that SLA monitoring tools are reachable via the management plane and that the solution has been defined for the full mesh network fabric.

Consider various operations that may be utilized for a first step of the process flow, as shown in Figure 2. For example, a mapping can be created between service instance(s) and SLA collectors on each leaf. Each service instance can have multiple SLA collectors and each service instance may have a unique identifier for fetching data from SLA collectors for the first step. The unique identifier may be different for each SLA collector. All service instances of same type should use the same service-identifier. If there are multiple instances of a given service on the same leaf, each instance should have a leaf-local unique instance identifier; a default instance identifier may be set to '0'. Figure 3, below, illustrates an example service configuration.

```
service Sa
    service-identifier 123 instance 3
     sla-collector sla-tool-1-x.sla identifier 123
    sla-collector sla-tool-2-x.sla identifier 345
    sla-collector sla-tool-3-x.sla identifier 123
```

Figure 3: Example Service Configuration

The data provided by each SLA monitoring tool can include a current health score for a given service, identified as 'C_{HS}', and a health score that is required to satisfy a user-defined SLA, identified as 'SLA_{HS}'.

Next, for a second step of the process flow, the fetched data can be summarized at each local leaf node. For example, a summarized score for each service can be calculated such that, for each SLA_M in the SLA collector list [1 . . . N] of service S_a, the (C_{HS}, SLA_{HS}) can be retrieved and a normalized value for the health score, labeled '$M(SLA_M)$', can be calculated as: $M(SLA_M) = C_{HS}/SLA_{HS}$. Next, a serviceability value for a service-leaf pair, labeled '$U(S_a)$', can be set to $U(S_a) = 1$ if all M scores are ≥ 1, otherwise $U(S_a) = 0$.

Thereafter for the second step, a 64-bit entry is created for each service-leaf pair, which is referred to herein as a service-serviceability tracker entry, as shown below in Figure 4. This data is sent across the network to enforce any traffic steering changes, as needed.

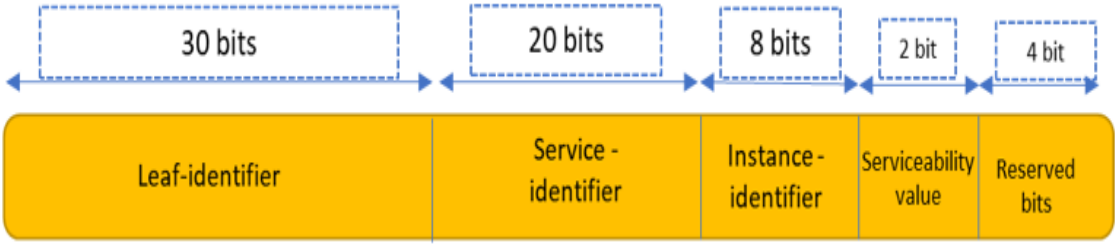

Figure 4: Service-Serviceability Tracker Format

For a third step of the process flow, the traffic steering rules can be enforced. For example, once each leaf summarizes the health score data, it will verify if the existing flows adhere to the serviceability information in order to ensure that every flow is mapping to a service that is serviceable.

If there is any flow that is steered to an unusable service, the corresponding entry can be removed from the hardware. Thus, when the next packet arrives for that flow, there will be no entry in the ternary content-addressable memory (TCAM) for the flow, which will trigger a TCAM miss. The logic of the hardware can register for this interrupt and install a new service to which to send the flow.

Such operations can result in two scenarios. In a first scenario for a given service, it may be possible to find the next usable service instance on the same local leaf, which means that the change is a localized change and there will be no need to inform other switches of the fabric of the change.

However, in a second scenario for a given service, the next usable service instance may only be found on another leaf in the fabric. This can be identified, for example, by comparing the leaf identifier information for the serviceability tracker data to the local leaf identifier. In such a scenario, the leaf is expected to perform two actions.

For a first action, as noted for the first scenario, the TCAM entry is to be removed in order to trigger a TCAM miss event such that, on interrupt, the flow can be redirected to the leaf that is hosting the next service instance. For a second action, the leaf is to

additionally inform the border gateway switch about the change. The information sent will include the flow information and serviceability tracker data used to update this action. Upon receiving the information, the border gateway will be able to identify the new local leaf for the service based on the leaf identifier information in the serviceability information and will reinstall the TCAM entry for this flow to point to the new leaf.

In summary, techniques of this proposal provide for creating a summarized double-bit value in order to represent serviceability of a service instance. Based on the location of a given switch in the fabric, the same serviceability tracker information can be consumed in different forms. Additionally, techniques of this proposal provide for the ability for third-party monitoring tools to interface with a closest connected leaf and to generate interrupts, which may facilitate improved SLA violation resolutions. Thus, the techniques of this proposal may provide for minimizing an SLA violation period and enforcing an optimal flow for traffic. Accordingly, the techniques of this proposal facilitate a datacenter fabric solution that provides traffic steering within a datacenter, as opposed to at edge/branch locations and, therefore, can help in saving bandwidth utilization in the fabric.