Technical Disclosure Commons

Defensive Publications Series

January 2020

SRV6/BITMASK DATAPLANE SIGNALED SEAMLESS FAAS/ SERVERLESS FUNCTION DEPLOYMENT

Nagendra Kumar Nainar

Carlos M. Pignataro

SRV6/BITMASK DATAPLANE SIGNALED SEAMLESS FAAS/SERVERLESS FUNCTION DEPLOYMENT

AUTHORS:

Nagendra Kumar Nainar
Carlos M. Pignataro

ABSTRACT

Presented herein are techniques through which a list of functions (FUNC) that are to be invoked in a system may be carried as part of a Segment Routing for Internet Protocol version 6 (SRv6) micro-Segment Identifier (uSID) or a bitmask in an interesting packet. A first packet may be used to invoke the functions and, once invoked, a more specific entry for the address can be instantiated in a forwarding table so as to avoid attempting to invoke the same functions for subsequent packets.

DETAILED DESCRIPTION

While many (if not most) of the latest applications are developed using a microservice architecture, the introduction of Function as a Service (FaaS) is making application development more scalable and compute efficient by allowing developers to build different functions (FUNCs) that can be instantiated on demand basis based on a trigger. Serverless computing allows for the implementation of service code as functions without compute resources always being allocated. Instead, functions are only instantiated when there is a trigger, such as a scheduled task, a HyperText Transfer Protocol (HTTP) request, a queue message, etc. While the concept of serverless computing may be compute efficient, some delay may be introduced in bringing up a service instance before executing a function.

Serverless computing may also be used for developing cloud native network functions (NFs). Figure 1, below, illustrates a simple example in which multiple virtualized NFs (vNFs) are deployed as functions that are invoked on a demand basis using a trigger, such as an interesting packet (e.g., a synchronization (SYN) message), that may be received by a Fission® Controller, etc.

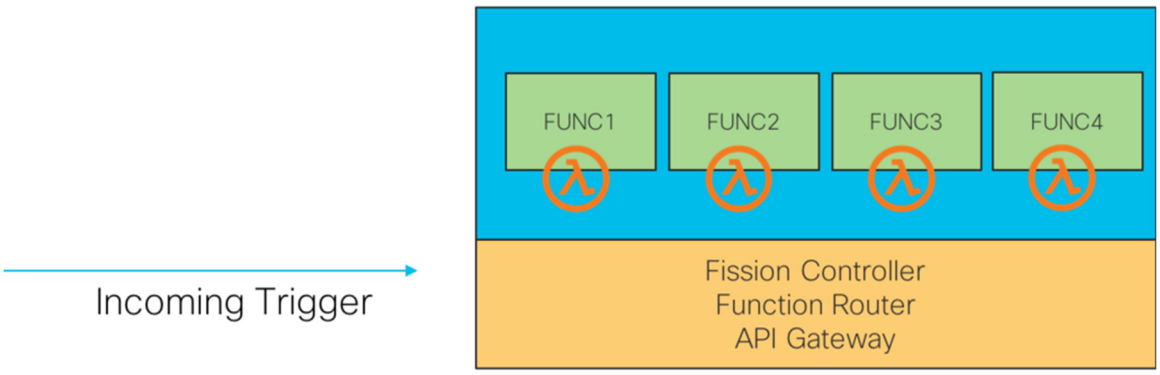

Figure 1

In the example illustrated in Figure 1, FUNCx may be function code of a vNF or a containerized NF (NF). Any combination of vNFs/cNFs may be used to apply different types of services. By default for this example, when there is interesting traffic, the controller will invoke the relevant FUNC in order to execute the relevant function code. Code execution normally takes a few hundreds of milliseconds (e.g., Fission® takes a minimum of 100 milliseconds (msec) using warm containers); thus, applying three services may take approximately 500+ msec.

This proposal provides two techniques that provide for dynamically invoking a dependent FUNC apriori using a Segment Routing (SR) micro-SID (uSID) based approach or a bitmask based approach.

SRv6 uSID based approach

In the SRv6 uSID based approach, each FUNC is assigned a unique micro-SID. For example, assume that the size of the uSID is 16-bits. A unique Prefix block (for example, FC00:FEED::/16) can be assigned for this purpose. A sample SRv6 SID comprising a stack of FUNC uSIDs is shown below in Figure 2.

Figure 2

As shown in the above example, FC01, FC02, FC03, and FC04, respectively, may each be a unique FUNC-SID (uSID) assigned for each FUNC. The FC00:FEED::/16 prefix can be used to identify that a stack of FUNC-SIDs is carried for functions that are to be invoked.

The dataplane can be programmed with fc00:feed::/16 with a forwarding semantic configured as below:

- Extract the FUNC-SIDs from the destination address field or a Segment Routing Header (SRH) and invoke the corresponding functions.
- Upon successfully invoking the functions, a more specific entry can be created to forward towards the functions.

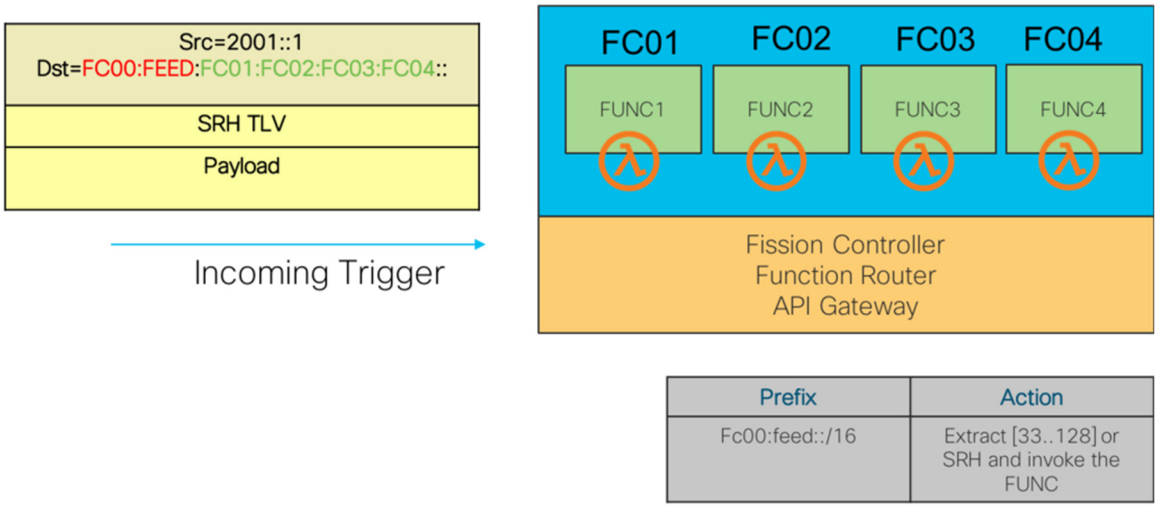

Figure 3

As shown in Figure 3, when a dataplane component receives the packet for a first time, it can match fc00:feed::/16 in the table that instructs the component to extract the FUNC-SID and invoke the functions. Upon invoking all the functions, the component can create a more specific state entry to forward to the relevant functions, as shown below in Figure 4.

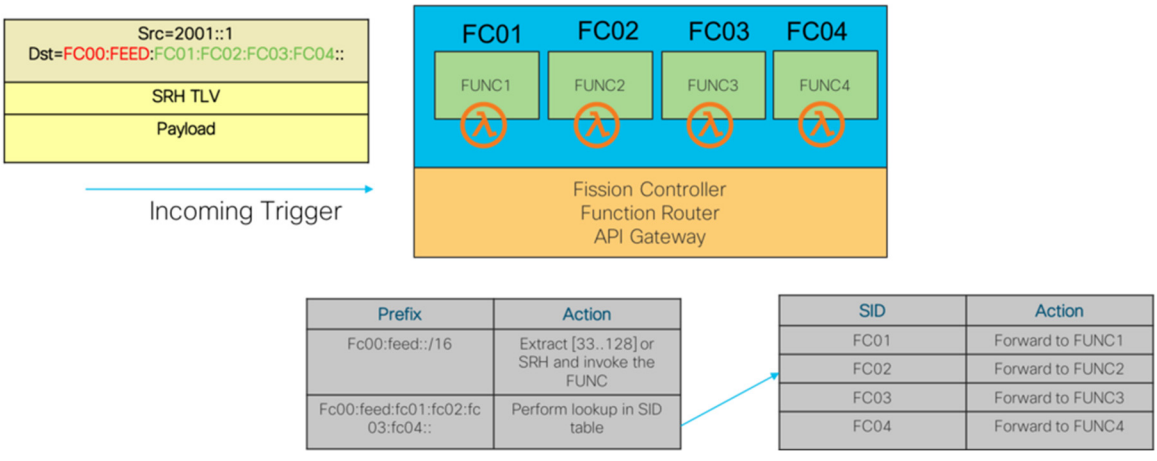

Figure 4

Any subsequent packets will match the more specific entry that points towards the SID table, which can be used to apply/execute towards the relevant functions. In some instances, a local timer may be utilized such that when the local timer expires without any active usage of a given function, the function can be stopped and the more specific entry can be removed from the table.

Bitmask based approach

The bitmask approach is similar to the SRv6 uSID based approach in that each FUNC can be assigned a unique bitmask. While a maximum of 64-unique bits can be assigned, an example is provided involving 4-bits for ease of understanding. For example, as shown in Figure 5, below, FUNC1 can be assigned a bitmask of '0001', FUNC2 can be assigned a bitmask of '0010', FUNC3 can be assigned a bitmask of '0100', and FUNC4 can be assigned a bitmask of '1000'.

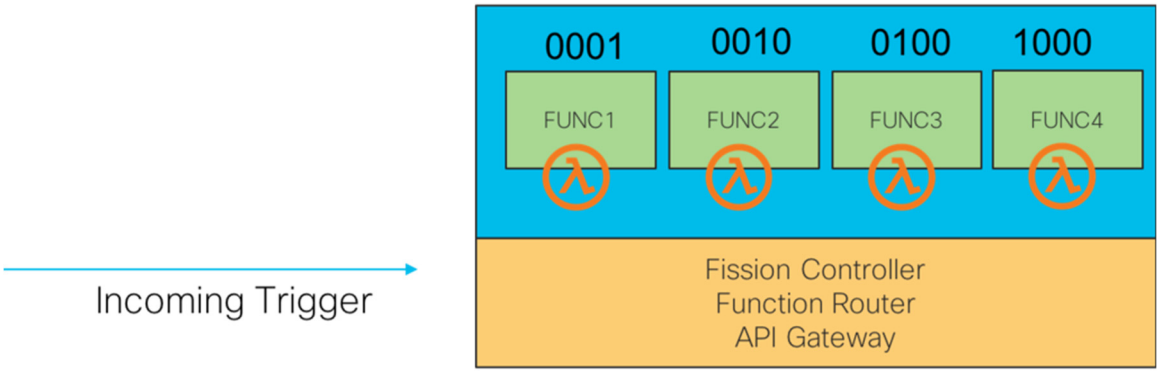

Figure 5

A unique prefix can be assigned as fc00:feed::/64 where the last 64-bits can be used to carry the bitmask, as shown below in Figure 6

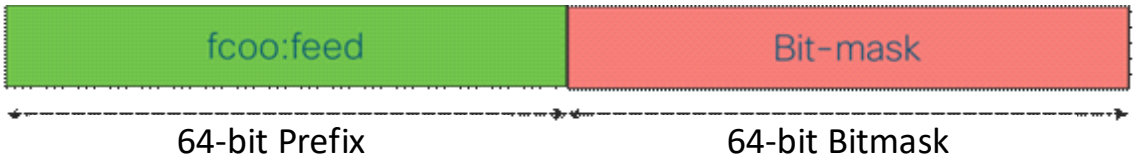

Figure 6

Upon receiving an interesting packet, the bitmask in the host portion can be used to invoke the relevant functions and a more specific entry can be created to forward the packet

to the functions. This approach may provide for the ability to carry more functions in the header as compared to the SRv6 uSID based approach.

In summary, this proposal provides two techniques through which a list of functions that are to be invoked in a system may be carried as part of a bitmask or a SRv6 uSID in an interesting packet. A first packet may be used to invoke the functions and, once invoked, a more specific entry for the address can be instantiated in a forwarding table so as to avoid attempting to invoke the same functions for subsequent packets.