Service Function Chaining (SFC) B. Rijsman
Internet Draft J. Moisand
Intended status: Informational Juniper Networks
Expires: August 2014 February 12, 2014

Metadata Considerations
draft-rijsman-sfc-metadata-considerations-00.txt

Status of this Memo

Copyright Notice

Abstract

This draft discusses the topic of metadata in the context of Service Function Chaining. It aims to define the problem space for metadata signaling, identify the key challenges, and guide the exploration and comparison of possible solutions.

Table of Contents

1. Introduction

This draft discusses the topic of metadata in the context of Service Function Chaining.

As described in [draft-quinn-sfc-arch] a Service Function Chain (or Service Chain for short) is a sequence of Service Functions such as Network Address Translation (NAT), Firewalls, Deep Packet Inspection (DPI), Intrusion Detection Systems (IDS), content caches, etc.

Service Functions process the traffic flows which traverse the Service Function Chain. In additional to the data which is in the processed traffic flow itself, the Service Functions may also benefit from additional contextual information about the traffic flows. This additional contextual information is referred to as metadata. One example of metadata is an Accounting-ID which is used for accounting and billing purposes.

This draft aims to define the problem space for metadata signaling, identify candidate approaches, describe the key challenges, and guide the exploration and comparison of possible solutions.

Section 2 defines what metadata is and what it can be used for, i.e. use cases for metadata.

Section 3 lists five different approaches for signaling metadata: in-band signaling (attaching metadata to each packet in a traffic flow), signaling metadata in application layer headers, congruent out-of-band metadata signaling, non-congruent out-of-band metadata signaling, and a hybrid approach combining in-band signaling with out-of-band signaling.

Section 4 discusses various challenges with metadata and describes the pros and cons of each of the five approaches in terms of dealing with these challenges.

Section 5 contains a summary and conclusion.

2. Metadata use cases

2.1. What is metadata?

Metadata is "data about data".

In the context of Service Function Chaining, metadata provides contextual information about the data packets which traverse a Service Function Chain.

The following sections provide concrete examples of what metadata can be used for.

2.2. Contextual information which is not locally available

Metadata can be used to transport contextual information which is available at one location in the network to another location in the network where that information is not readily available.

Edge routers often have detailed information about each traffic flow such as the subscriber identity (Subscriber-ID) and the associated policies (i.e. sets of rules which need to be applied to the traffic flow).

This information is available at the network edge as a result of the customer-facing interface on which the traffic arrives, or as a result of information in encapsulations or protocols which are stripped off at the edge of the network, or as a result of interactions with policy servers such as Authentication, Authorization, and Accounting (AAA) servers.

Deeper in the network this detailed information is either not available or difficult to obtain.

We illustrate this using two concrete examples:

- In Long Term Evolution (LTE) mobile networks, the Packet Data Network Gateway (PGW) sits at the edge of the IP network. The PGW decapsulates packets from the General Packet Radio Service Tunneling Protocol (GTP). The PGW uses the Diameter protocol [RFC6733] to interact with the Policy and Charging Rules Function (PCRF) server. As a result, the PGW knows the subscriber identity and policy for each traffic flow. This information is not easily available deeper in the network.

- In fixed broadband networks, the Broadband Network Gateway (BNG) sits at the edge of the IP network. The BNG decapsulates packets from a variety of encapsulations such as Point-to-Point over Ethernet (PPPoE). The BNG interacts with an Authentication, Authorization, and Accounting (AAA) server using the Remote Dial In User Service (RADIUS) protocol. As a result, the BNG know the subscriber identity and policy for each traffic flow. This information is not easily available deeper in the network.

These edge routers (such as PGWs or BNGs) may be used as the starting point for a Service Function Chain.

Metadata can be used to signal information from the place where it is readily available (e.g. the PGW or the BNG at the start of the Service Function Chain) to places deeper in the network where it is not readily available (e.g. the Service Functions in the Service Function Chain).

Examples of such metadata include:

- A Subscriber-ID, or more accurately, an Accounting-ID which maps traffic flows to a unique identifier used for accounting and billing purposes.

- A Service-Profile-ID and/or Service-Profile-Parameters which identify the service which the Service Functions must apply to the traffic flow. This is discussed in more detail in section 2.4.

2.3. Avoiding repeated execution of expensive operations

Certain types of information are computationally expensive to extract from the traffic flow.

For example, it is computationally expensive to perform Deep Packet Inspection (DPI) because the TCP stream may have to be reconstructed and sophisticated layer-7 pattern matching algorithms must be applied.

If multiple Service Functions need the same information, it is more efficient to perform the computationally expensive operation (e.g. DPI) only once and to share the result with other Service Function in Service Function Chain using metadata.

As a concrete example of a use case that can benefit from not performing the same expensive operation more than once, we consider some examples of Service Functions which could benefit from the availability of metadata (namely an Application-ID) provided by a DPI Service Function earlier in the Service Function Chain.

A caching Service Function may only want to attempt to cache YouTube video traffic but not Netflix video traffic.

A firewall Service Function for a human resources department may want to block Facebook games but allow other type of Facebook traffic.

A WAN optimization Service Function may want to optimize database synchronization traffic but not video conference streaming.

Each of these Service Functions needs to know what type of application layer traffic is carried in each traffic flow.

Rather than repeatedly doing DPI at each Service Function (at the cache, the firewall, and the WAN optimizer) we can do DPI once and associate the resulting Application-ID as metadata with the traffic flow.

2.4. Fine-grained policies

For coarse-grained policies, i.e. if the number of policies is small, it is feasible to create a separate Service Function Chain for each policy. The Service Classifier can apply the correct policy to each traffic flow by steering that traffic flow into the corresponding Service Function Chain.

As a concrete example, consider a scenario with two Service Functions: a firewall and a cache. Some traffic flows only visit the firewall, some traffic flows visit only the cache, and some traffic flows visit both the firewall and the cache. This can be implemented by creating three separate Service Function Chains:

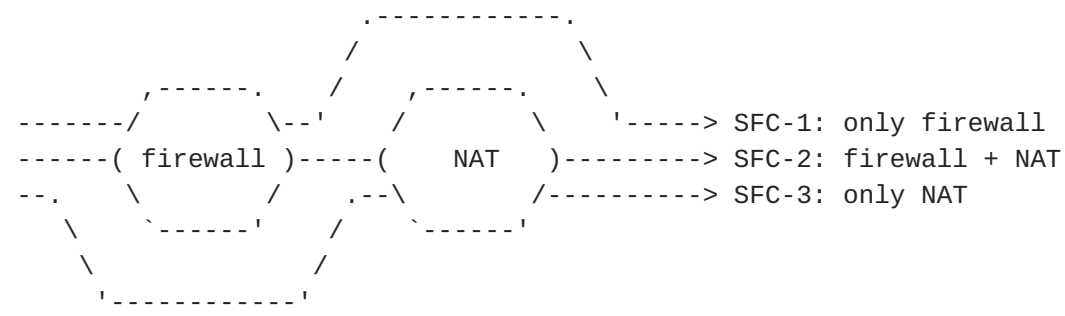

Figure 1: Combinations of Service Functions

As the number of Service Functions increases, the number of possible combinations an permutations increases exponentially. Hence, it may not be feasible or efficient to create separate Service Function Chains.

Furthermore, it may be necessary to apply different service policies to different traffic flows which visit the same sequence of Service Functions.

For example, the firewall may block certain websites for one traffic flow and other websites for a different traffic flow.

One way of implementing this is to create a separate Service Function Chain for each unique combination of service policies.

In the following example we have three Service Function Chains which all visit a firewall and a throttle service. The Service Function Chain implicitly determines which service policy is applied at the firewall and at the throttle service. For example, the third Service Function Chain (SFC-3) receives blocks Facebook at the firewall and is limited to 30 Mbps at the throttle service:

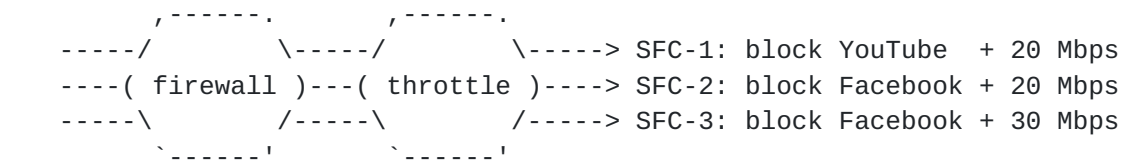

Figure 2: Combinations of service policies

Once again, as the number of service policies increases at each Service Function, the number of possible combinations of service policies increases exponentially. Hence, it may not be feasible or efficient to create separate Service Function Chains.

This problem is aggravated when the service policies become more "fine grained". For example, in the example in figure 2 above we only have two possible values for the bandwidth namely 20 Mbps or 30 Mbps. Consider what would happen if we had ten or more possible bandwidth values: the number of possible service policies combinations would suffer a combinatorial explosion.

In the extreme case, if each customer could have his own individual bandwidth limit, this would imply a Service Function Chain for each individual customer.

Instead of creating separate Service Function Chains, we can use metadata to signal the fine grained policy information. The metadata can identify which service policy needs to be applied at each Service Function, and the metadata can carry fine grained service policy parameters (such as the bandwidth in the above example) in the form of parameter values for a policy profile or policy template.

There is a trade-off between simplicity and flexibility. If the operator provides a small number of uniform network services, then the simple approach of using one Service Function Chain per network service suffices and no fine grained policy metadata is needed. On

the other hand, if the number of unique network services is large or variable (i.e. dependent on the subscriber) then the additional complexity of fine grained policy metadata is justified by the additional flexibility it provides.

3. Metadata signaling approaches

This section describes various approaches for signaling metadata.

3.1. In-band marking

The metadata can be transferred by attaching a metadata field to each packet which traverses the Service Function Chain.

In this draft we refer to this "in-band" marking because the metadata and the data are carried in the same communications channel, i.e. stored in the same packet.

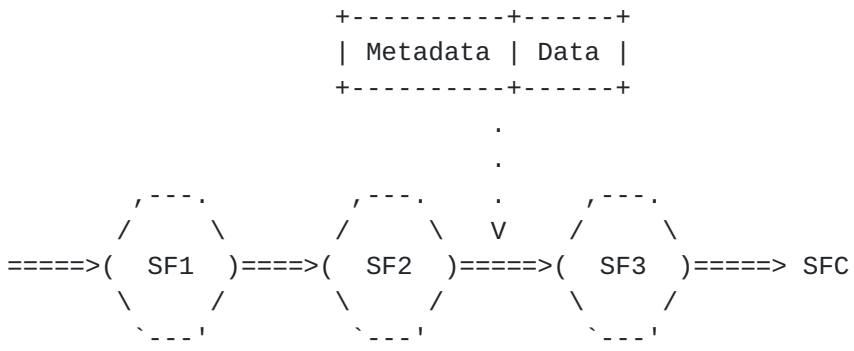

Figure 3: Metadata in each packet

The metadata may simply be attached to each data packet as shown above. Or, the extra field which is attached to each packet may be used to implement a more sophisticated protocol in which case it would be more appropriate to speak of in-band signaling instead of in-band marking. There is a fine line between in-band signaling and congruent out-of-band signaling which is discussed in section 3.3.

The figure above is purposely vague on where exactly in the data packet the metadata is carried. There are several options:

The field which carries the metadata can be a new field which is introduced specifically for the purpose of carrying metadata. The Network Service Header (NSH) described in [draft-quinn-sfc-nsh] is an example of this approach.

Alternatively, the field can be an existing field such as a new IPv4 option or a new IPv6 extension header. This has the advantage of minimizing the impact on existing routers, switches, hosts, and applications; they are able to receive and forward packets with options or extension headers which they do not understand.

Note that the presence of IPv4 options or IPv6 extension headers may cause the packets to be processed in the so-called "slow path" of some core routers instead of the fast path and hence impair forwarding performance.

3.2. Metadata in application layer headers

Another approach is to carry metadata in the headers of application-layer messages.

This approach works well the Hypertext Transfer Protocol (HTTP) [RFC2616] where it easy to introduce new header fields into messages.

Hypothetically, this approach can also be done with similar text-based protocols such as the Simple Mail Transfer Protocol (SMTP) [RFC5321] or the Real Time Streaming Protocol (RTSP) [RFC2326], although in practice it is mostly used with HTTP. This approach does not work well with binary application layer protocols.

Only Service Functions which process traffic at the application layer (e.g. parse HTTP) have access to this kind of metadata.

Routers and switches and Service Functions which process traffic at layer 1 through 4 do not have access to the metadata. This is not necessarily always a disadvantage - the expense of parsing the metadata is incurred only at the places which need it.

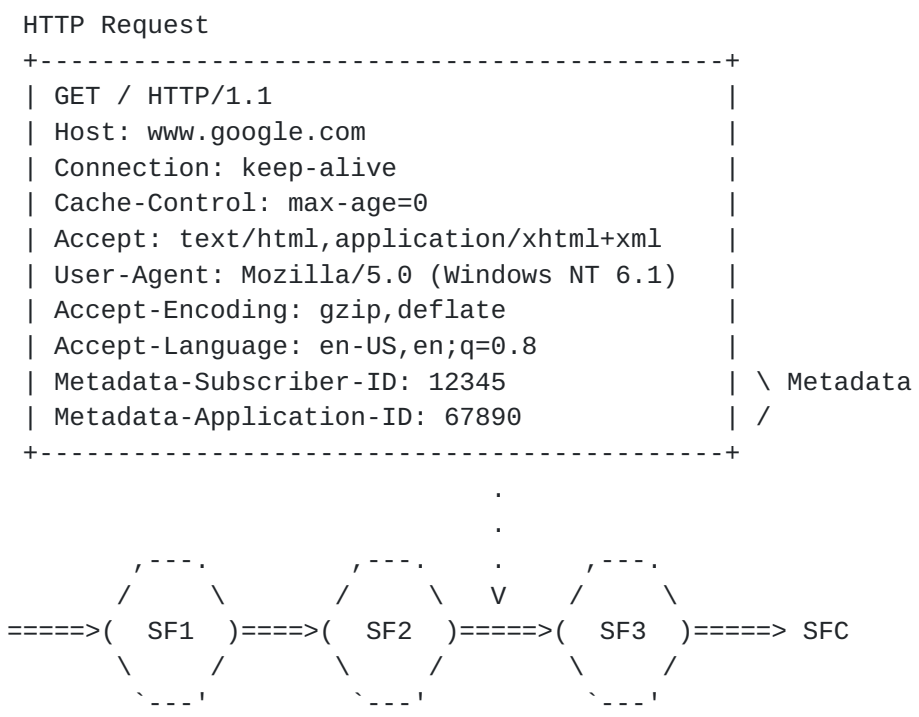

Figure 4: Metadata in HTTP headers

The above example uses hypothetical non-standard headers Metadata-Subscriber-ID and Metadata-Application-ID to carry the metadata. [RFC2774] defines an HTTP extension framework (which is not used in this example) for introducing new HTTP headers. The practice of prefixing non-standard headers with X- has been deprecated by [RFC6648].

3.3. Congruent out-of-band metadata signaling

Metadata can be signaled using a congruent out-of-band control plane protocol.

"Out-of-band signaling" means that the data plane and the control plane for signaling the metadata are carried in different communication channels (i.e. different packets). This is opposed to "in-band marking" which means that the data and metadata are carried in the same communication channel (i.e. in the same packet as discussed in section 3.1.).

"Congruent" means that the data plane protocol and the control plane protocol follow the same path through the network.

The classical example of a congruent out-of-band protocol is the File Transfer Protocol (FTP) [RFC959] which has separate data and control connections, but they follow the same path through the network.

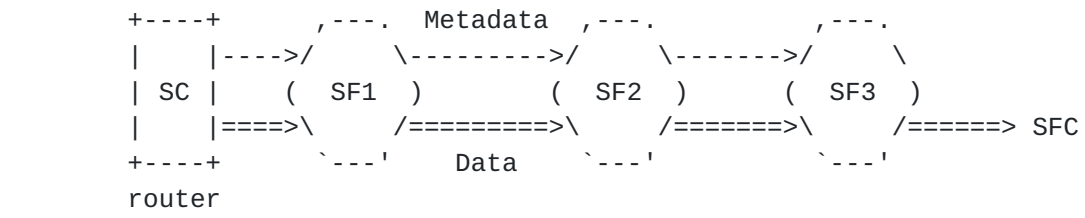

Figure 5: Congruent out-of-band metadata signaling

With congruent out-of-band control plane signaling, the router at the head of the service chain sends control plane messages to the first Service Node (SN) in the Service Function Chain to associate metadata with a particular traffic flow. The Service Node makes the metadata available to the Service Function to consume. The Service Function also has the opportunity to add or modify metadata. The Service Node then uses control plane signaling to propagate the new metadata for the traffic flow to the next Service Node. Etcetera.

New control plane protocols could be introduced for the congruent out-of-band metadata signaling, or existing protocols such as the Layer 2 Tunneling Protocol (L2TP) [RFC2661,RFC3931], or the Resource Reservation Protocol (RSVP) [RFC2205] could be extended to do metadata signaling.

3.4. Non-congruent out-of-band metadata signaling

Metadata can be signaled using a non-congruent out-of-band control plane protocol, separate from the data plane protocol which carries the packets through the Service Function Chain.

"Non-congruent" means that the data plane protocol and the control plane protocol follow different paths through the network.

A classic example of a non-congruent out-of-band protocol is an Interior Border Gateway Protocol (IBGP) [RFC4271] session to a route reflector: the BGP session and the data traffic can follow completely different paths through the network.

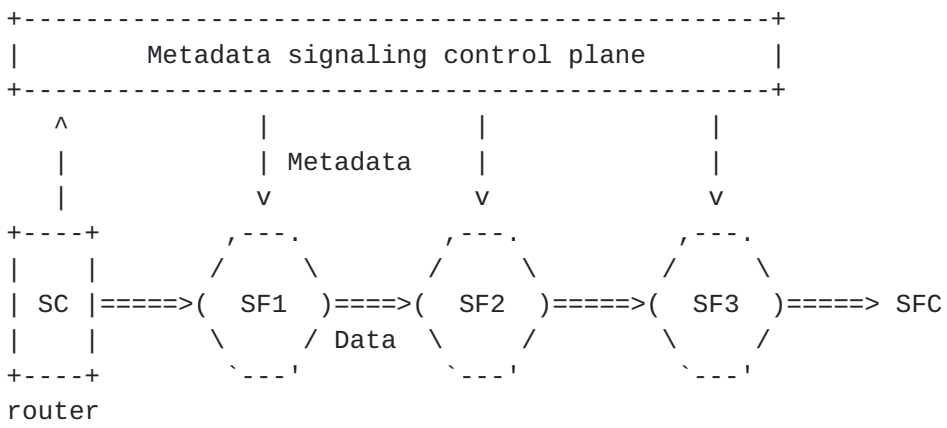

Figure 6: Non-congruent out-of-band metadata signaling

The control plane could consist of a request-response model where the Service Functions explicitly request the metadata when a traffic flow is first observed.

New control plane protocols could be introduced for the non-congruent out-of-band metadata signaling, or existing protocols such as Remote Authentication Dial In User Service (RADIUS) [RFC2865], Diameter [RFC6733], or the Session Initiation Protocol (SIP) [RFC3261] could be extended to do metadata signaling.

Alternatively, the control plane could consist of a publish-subscribe ("pub-sub") message bus. Any Service Function in the Service Function Chain or the router at the start of the Service Function Chain can publish metadata for individual traffic flows. Any Service Function in the Service Function chain can create a subscription to receive metadata for traffic flows. The subscription can include a filter to receive only specific type of metadata of interest to the Service Function.

One could use a pub-sub message bus such as ZeroMQ [zeromq.org] or a pub-sub database such as Redis [redis.io].

3.5. Hybrid in-band marking and out-of-band signaling

Metadata can be signaled using a hybrid approach combining in-band marking and out-of-band signaling.

Each data packet can carry a small fixed-length field which serves as a "correlator". An out-of-band signaling protocol can then be used to map the correlator value to the actual metadata value(s).

The correlator field is typically a form of session identifier; we will use the term Session-ID in this draft. All data packets with the same Session-ID belong to the same session.

The concept of a correlator field is very similar to the concept of a Forwarding Equivalence Class (FEC) which defines a group of packets which need to be treated in the same way.

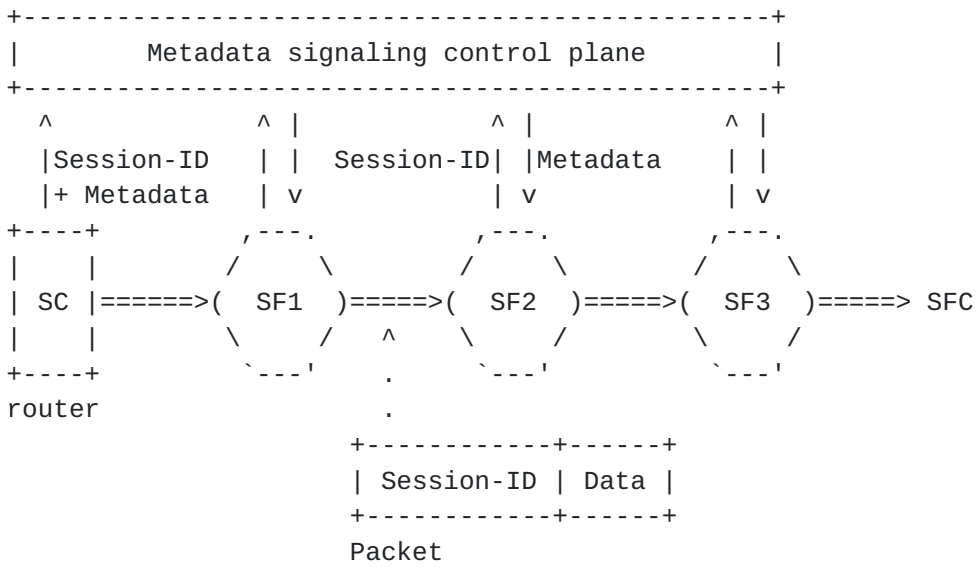

Figure 6: Hybrid in-band marking and out-of-band metadata signaling

The Layer 2 Tunneling Protocol (L2TP) [RFC2661,RFC3931] and the General Packet Radio System Tunneling Protocol (GTP) [GTP] are both examples of hybrid protocols. Each has a data plane protocol (GTP-U in the case of GTP) which carries a Session-ID. The Session-ID is used as a correlator to a separate control plane protocol (GTP-C in the case of GTP).

4. Metadata challenges

This section discusses the challenges associated with metadata.

4.1. Support for metadata-unaware Service Functions

There is already a large number of Service Functions on the market such as firewalls, load balancers, WAN optimizers, caches, DDoS mitigation, etc. These existing Service Functions are available in both physical form and virtual form.

Since metadata has not yet been standardized, the existing Service Functions generally don't understand how to extract and process metadata. They generally expect to receive and send normal IP over Ethernet packets, possibly with a VLAN tag. We refer to such Service Functions as being metadata-unaware.

If the IETF defines a new type of encapsulation header specifically for the purpose of carrying metadata, it will not be possible to inject packets with the new metadata header into existing metadata-unaware Service Functions. Such packets would be dropped because the new encapsulation is not recognized.

Thus, for backwards compatibility it will be required to strip any new metadata headers from the packet before it is injected into a metadata-unaware Service Function.

One possible implementation choice is to always strip the metadata from the packet before injecting it to any Service Function, both metadata-aware and metadata-unaware Service Functions. The metadata-aware Service Functions could choose to retrieve the metadata, for example by calling an API (e.g. a socket call) to retrieve the metadata.

The other approaches (i.e. other than introducing a new header for metadata) can handle metadata-unaware Service Functions more easily.

Existing Service Functions are able to ignore new HTTP headers. Many existing HTTP-aware Service Functions can be configured to recognize specific HTTP headers, even non-standard HTTP headers, and take specific actions based on the presence or value of specific HTTP headers.

Existing Service Functions are able to ignore new IPv4 options or new IPv6 extension headers. However, most existing Service Functions can NOT be configured to take action based on the presence or value of a specific option or extension header.

If an out-of-band control plane protocol is used to signal metadata, metadata-unaware Service Functions can ignore the metadata by simply not participating in the control plane protocol.

4.2. Preserving metadata through metadata-unaware Service Functions

As described in the previous section, there are ways to support a metadata-unaware Service Function in a Service Function Chain, at least in the sense that the metadata-unaware Service Function will not choke on the traffic flow. The metadata-unaware Service

Function will see what appears to be a completely normal IP over Ethernet traffic flow (possibly with a VLAN tag).

However, there is another problem. The metadata should be preserved when it passes through a metadata-unaware Service Function.

Consider, for example, a scenario where the metadata is stripped from the packet before the packet is injected into a metadata-unaware Service Function. For Virtual Service Functions running in a virtual machine, the metadata is typically stripped in the virtual switch (vSwitch) or virtual router (vRouter) in the hypervisor of the virtualized server. For Physical Service Functions running on hardware, the metadata is typically stripped in the proxy node (sometimes called gateway node) in front of the Physical Service Function.

Now imagine there is a subsequent Service Function later in the Service Function Chain which is metadata-aware. In this case we need to re-attach the metadata to the packet after it leaves the metadata-unaware Service Function and before it reaches the metadata-aware Service Function.

Clearly, the metadata-unaware Service Function cannot re-attach the metadata because it is, by definition, not aware of the existence of the metadata.

At first sight, it seems reasonable to expect the vSwitch or vRouter or proxy node which stripped the metadata to also re-attach the metadata. In practice, however, this is extremely difficult to implement. The algorithm for stripping and re-attaching the metadata would look something like this:

1. For every received packet P do the following

2. Strip the metadata from P

3. Store the stripped metadata

4. Inject the packet into metadata-unaware Service Function F

5. Service Function F receives the packet P

6. Service Function F processes packet P

7. Service Function F sends packet P

8. Retrieve the stored metadata for packet P

9. Re-reattach the metadata to packet P

10. Send packet P

This seems simple enough but there are two major problems.

The first problem is in steps 3 and 8. What is the key which we use to store and retrieve the stored metadata? For every given flow, there are going to be many packets P1, P2, ..., Pn which have exactly the same Ethernet source and destination address, the same IP source and destination address, the same protocol ID, the same TCP source and destination port etc. However, packets P1, P2, ..., Pn may have different metadata. Furthermore, certain Service Functions such as Network Address Translation (NAT) and HTTP Proxies change the source or destination addresses or ports of the packets.

The second problem is in steps 5, 6 and 7. There is an implicit assumption that Service Function F sends exactly one packet P for every received packet P. In reality, this is not a valid assumption. Service Function F may drop packets, re-order packets, duplicate packets, insert new packets, or even completely change the traffic flow including the addresses (e.g. a proxy).

A similar problem occurs for metadata which is stored in an existing header such as an IPv4 option or an IPv6 extension header. In this case it is not necessary to strip the metadata from the packet before it is injected into the metadata-unaware Service Function. However, the metadata is typically lost when it traverses the metadata-unaware Service Function. Existing Service Functions on the market are often implemented by opening a pair of sockets, reading packets from one socket, processing the packets, and sending the packets on the other socket. Typically implementations using socket APIs do not preserve IPv4 options or IPv6 extension headers.

How can we deal with this problem?

One possible approach is to simply accept that the metadata is lost once the traffic flow visits a metadata-unaware Service Function. Any subsequent Service Functions in the Service Function Chain do not have access to the metadata anymore.

Another possible approach is to introduce the restriction that if a Service Function Chain contains at least one metadata-aware Service Function, then all Service Functions in the Service Function Chain must be metadata-aware.

A third possible approach is to modify the Service Function implementation to retrieve and maintain the metadata. That is just a different way of saying that all Service Function should be metadata-aware which will take time.

Some of the other approaches (i.e. other than storing the metadata in the same packet as the data) can more easily preserve metadata as it passes through a metadata-unaware Service Function.

All HTTP headers, even unrecognized HTTP headers, are generally preserved when they pass through a Service Function.

The control plane based approaches can simply skip the metadata unaware service. For non-congruent out-of-band metadata signaling this is trivial. For congruent out-of-band metadata signaling this is also easy, but care must be taken that the control plane does not run in the Service Function itself but in the vSwitch or vRouter or proxy node 'below' the Service Function.

4.3. Metadata encoding

There is a tradeoff between flexibility and performance when making a decision on how to encode metadata.

We can use a fixed number of metadata fields or a variable number of metadata fields.

Each metadata field can have a fixed length format or a variable length format.

We can support a fixed set of metadata field types, or we can support an extensible framework where different vendors and users can introduce new metadata field types using some sort of allocation mechanism.

Clearly, supporting a variable number of metadata fields, each with a variable length, each with a different syntax, and with some registration mechanism to introduce new metadata types provides the most flexibility and extensibility.

Such flexibility and extensibility is perfectly feasible for some approaches, namely when we carry metadata in HTTP header, or when we use extensible out-of-band signaling protocols such as Diameter.

However, when the metadata is attached to each packet (either in a new header or in an existing field such as an IPv4 option or an IPv6

extension header), flexibility and extensibility come at such a high cost that it is not practical.

When metadata is attached to a packet it must still be possible to forward packets in the fast path of a router or switch. It is very difficult and expensive to deal with variable-length encoding and with fields with an unbounded length in hardware. At first sight, this is not a problem because the metadata can be stored behind the Ethernet header and the IP header. However, in practice it is a problem. The fast path must often look deeper into the packet for Access Control Lists (ACLs), Quality of Service (QoS), firewalling, tunnel decapsulation, etc.

4.4. Scalability and performance of the control plane

Scalability and performance of the control plane is a concern when we use out-of-band metadata signaling.

Typically there will be at least one control plane message exchange for every new traffic flow or for every new session which passes through a service chain to associate metadata with that traffic flow. The rate at which traffic flows or sessions appear and disappear can be very high: thousands per second or more. This requires a very high performance control plane. The number of simultaneously active traffic flows or sessions, each with its own state, can be very high: millions or more. This requires a very scalable control plane.

The control plane interaction also causes extra latency for the data plane traffic.

4.5. Synchronization between the control plane and the data plane

In the out-of-band signaling based approaches, the data and the metadata are carried over separate communication channels. The data is carried over a data plane channel, and the metadata is carried over a control plane channel.

The data and the metadata for a new flow do not arrive simultaneously at a Service Function. The data may arrive before the metadata, or the other way around.

As a result, the Service Function may have to buffer the data packets while waiting for the metadata to arrive. This requires stateful processing of traffic flows which is difficult and expensive to implement in hardware. Also, the memory for the additional buffers increases the cost.

Alternatively, the Service Function may simply drop the data while waiting for the corresponding metadata, relying on the application layer to retransmit the data. This increases latency and decreases end-user perceived quality.

The delay and buffering problem is aggravated for non-congruent out-of-band signaling (compared to congruent out-of-band signaling) because the control plane uses a different path for the control plane which is typically much slower than the data plane path because it uses slower interfaces and passes through intermediate control plane nodes.

4.6. Distributing metadata only to interested Service Functions

It is highly desirable that metadata is only signaled to those Service Functions which actually need it.

In other words, ideally a Service Function should only receive:

- Metadata for those traffic flows which actually visit that instance of the Service Function. For example, if a Service Function Chain contains multiple parallel scale-out instances of a firewall, each firewall should only receive metadata for the traffic flows which actually pass through that instance of the firewall.

  This is easy to achieve with in-band marking and with congruent out-of-band signaling. With non-congruent out-of-band signaling is also easy to achieve if a "pull" model is used, but it is non-trivial to achieve if a "publish" model is used.

- Metadata which is of interest to that particular Service Function. For example is a Service Function chain contains a sequence of a firewall and a cache, then the firewall should only achieve the metadata which of interest to the firewall and the cache should only receive the metadata which is of interest to the cache. There could of course also be metadata which is of interest to both the firewall and the cache.

  This is easy to achieve with non-congruent out-of-band signaling. It is more difficult to achieve with in-band marking and with congruent out-of-band signaling.

4.7. Associating data plane flows with control plane signaling

If we use out-of-band control plane signaling for metadata, there must be a way to associate a control plane message with a data plane flow.

There will be a control plane message which says "use metadata M for traffic flow F" or some variation thereof.

The question is: how is traffic flow F identified?

A seemingly obvious answer is to use the 5-tuple (source IP, destination IP, protocol, source port, destination port) of the flow or some subset of the 5-tuple.

This approach is thwarted by the fact that many (possibly most) service chains contain one or more services which modify the 5-tuple, such as for example Network Address Translation (NAT) or HTTP proxies.

One possible approach is to use the IPv6 flow label field. Unfortunately, there is not standard flow label field for IPv4. Plus, existing Service Functions are not likely to preserve the flow label as discussed in section 4.2.

Another possible approach is to use an approach which is a hybrid between the in-band marking approach and the out-of-band signaling approach as described in section 3.5.

4.8. Layering considerations

As described in [draft-quinn-sfc-nsh] Service Function Chains consists of one or more parallel Service Paths.

Each Service Path consists of a concatenation of Service Functions which are connected by Service Path Segments. These Service Functions may be Virtual Service Functions on Service Nodes. Or, they may be Physical Service Functions connected to proxies or gateways.

The Service Classifier at the head of the Service Function Chain perform load balancing by steering each traffic flow in the chosen Service Path.

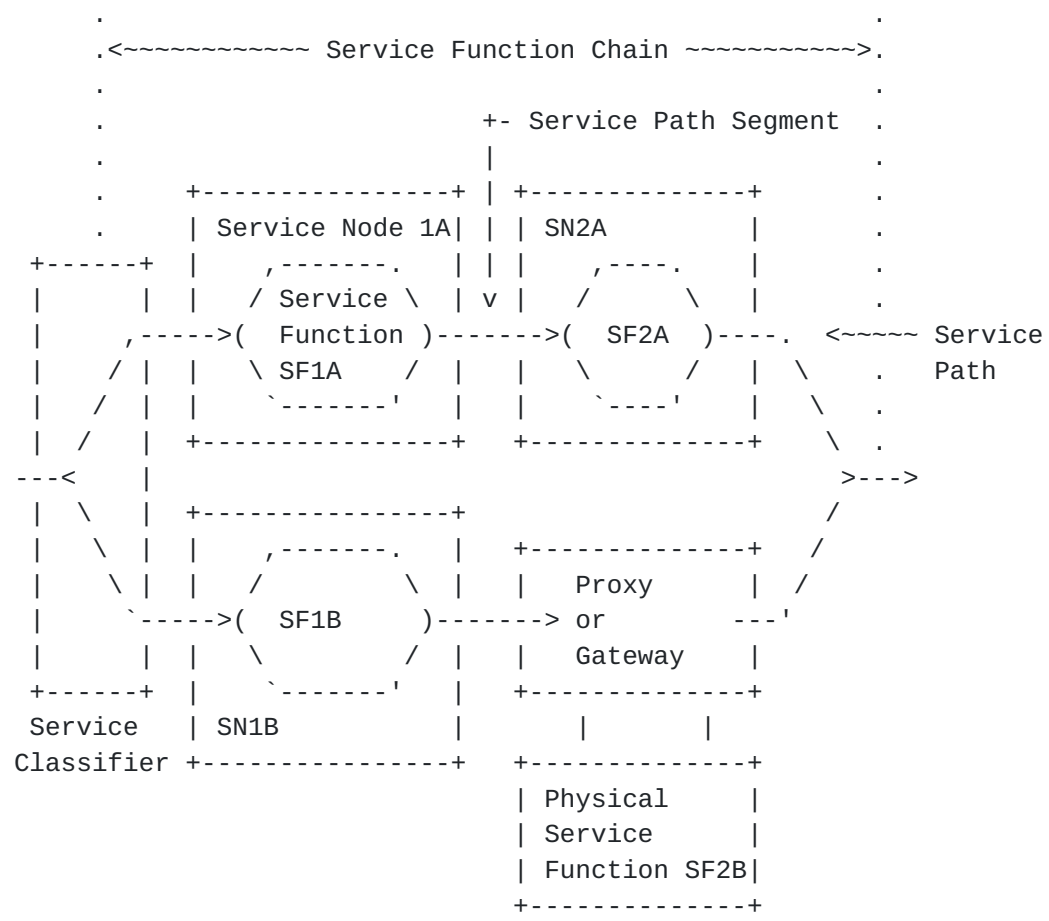

Figure 7: Service Function Chains, Paths, and Path Segments

It is useful to think of this architecture in terms of layers:

1. The service function layer is responsible for performing the service at each Service Function and for steering the packets into the right Service Path.

2. The service path layer is responsible for transporting data packets from one Service Function to the next Service Function in the Service Path.

3. The traditional network layer provided by the underlay network is responsible for forwarding packets from Service Node to Service Node.

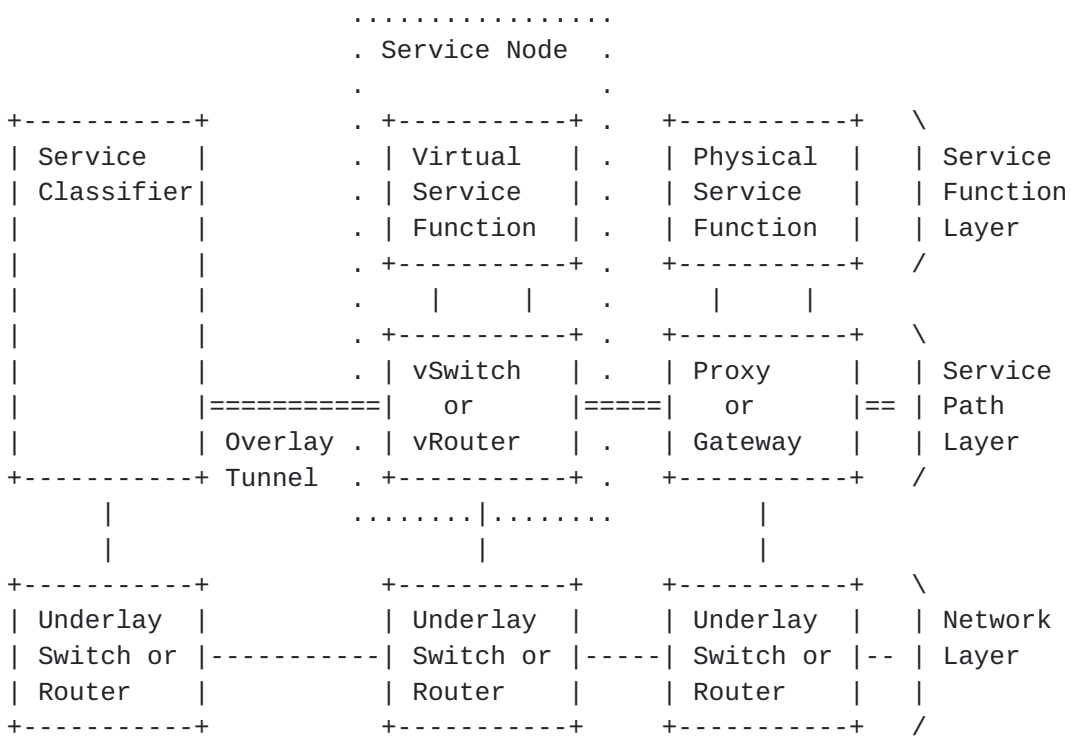

Figure 8: Layers

In the service path layer, each Service Path Segment is typically implemented using some sort of overlay tunnel.

The overlay tunnel can be implemented in various ways, including Virtual Local Area Networks (VLANs) [802.1Q], Virtual Extensible Local Area Networks (VXLANs) [draft-mahalingam-dutt-dcops-vxlan], Generic Route Encapsulation (GRE) [RFC2784] tunnels, or Label Switched Paths (LSPs) [RFC3031].

At the end of each Service Path segment there is some entity which decapsulates the packets from the overlay tunnel and dispatches the packets into the right Service Function. For Virtual Service Functions may be a virtual switch (vSwitch) or virtual router (vRouter). For Physical Service Functions this may be a proxy or gateway.

The tunnel encapsulation needs a field to identify the Service Path Segment. The vSwitch or vRouter or proxy or gateway uses this field to dispatch each packet into the correct Service Function. This field can be implemented in multiple ways, including a VLAN tag, a Virtual Network Identifier (VNID) in the VXLAN header, a Multi-Protocol Label Switching (MPLS) label, or the service path field in

the base service header of the Network Service Header [draft-quinn-sfc-nsh].

In addition to the field which identifies the Service Path as just discussed, there may be more fields in the packet which contain additional metadata such for example as the Session-ID or the Accounting-ID.

These additional metadata fields are only relevant in the service function layer. They are attached by the Service Classifier. They are read and acted upon by the Service Functions. The vSwitches, vRouters, proxies, or gateways never need to read or write this additional metadata.

Thus, it is important to make a distinction between fields which are used at the service path layer to identify the Service Path Segment, and additional fields which carry metadata which is imposed and interpreted at the service function layer. Combining both types of fields into a single header should probably be avoided from a layering point of view.

The need for such separation is reinforced by the existence of various ways to convey metadata that were discussed in chapter 3. It isn't necessarily obvious that packet marking is the best way to implement the service function layer.

4.9. Load balancing and symmetry

In a scaled-out service, i.e. when the Service Function Chain consists of more than one Service Path, load balancing is used to assign each traffic flow to exactly one of the available Service Paths.

Often the Service Functions in a Service Function chain are "stateful" and as a result they may need to see all packets in a flow in both directions. In that case, the forward and the reverse traffic flow must be assigned to the same Service Path, i.e. load balancing must be symmetric. (This is actually difficult to achieve when the different Service Paths terminate on different Service Nodes, but that discussion is beyond the scope of this draft.)

The metadata signaling protocol must ensure that the metadata for a given traffic node is signaled to the same Service Path (i.e. the same set of Service Nodes) which process the data for that traffic flow. Ideally, the metadata is not signaled to any other Service Nodes, i.e. not signaled to places where it is not needed.

This is easy to achieve when the metadata is attached to the data and it is also easy to achieve with congruent out-of-band signaling. It is more challenging to achieve for non-congruent out-of-band signaling, particularly in a pub-sub model.

4.10. High availability and geographic dispersion

Any kind of construct requiring high-rate out-of-band communication with a policy or signaling server is subject to all sorts of challenges if said server is suddenly no longer reachable (not necessarily a crash, but a simple connectivity loss). In other words, one has to factor in the geographical distribution of the problem.

In the non-congruent out-of-band signaling approach there may be logically centralized servers (e.g. RADIUS or Diameter servers) that communicate with the Service Nodes and the Service Classifiers.

The Service Nodes and Service Classifiers may be geographically dispersed. A single Service Function Path may visit Service Functions which are located in a central office, a distributed regional data/service center, and a more centralized data/service center.

Using a single (logical) server to control all geographic locations introduces challenges from a scaling and high availability perspective.

Using multiple (logical) servers improves scalability and reduced the size of failure domains, but introduces complexity because the servers must use federation protocols for coordination.

4.11. Multiple sources of metadata

In simple use cases it may be sufficient for all metadata to be produced and associated with the traffic flow once at the start of the service chain.

On more sophisticated use case it may be useful to allow Service Functions in the Service Function Chain to associate additional metadata with a traffic flow as the data traverses the service chain.

In the latter case, the signaling mechanism must support multiple sources of metadata.

4.12. Extensibility

The metadata mechanism must be extensible.

As new Service Functions are introduced, it must be possible for to introduce new types of metadata using some sort of registration process.

The extension mechanism must support backward compatibility: it must be possible to distribute a new type of metadata in a Service Chain which includes some Service Functions that do not understand the new type of metadata.

Extensibility is particularly important for the service function layer, more so than for the service path layer (see section 4.8. for layering).

5. Conclusion

In this draft we discussed several metadata signaling approaches:

- In-band marking
- Metadata in application layer headers
- Congruent out-of-band metadata signaling
- Non-congruent out-of-band metadata signaling
- Hybrid in-band marking and out-of-band signaling

We also discussed metadata signaling challenges:

- Support for metadata-unaware service function
- Preserving metadata through metadata-unaware service functions
- Metadata encoding
- Availability and performance of the control plane
- Synchronization between the control plane and the data plane
- Distributing metadata only to interested service functions

- Associating data plane flow with control plane signaling
- Layering considerations
- Load balancing and symmetry
- High availability and geographic dispersion
- Multiple sources of metadata
- Extensibility

We also discussed how each of the signaling approaches was affected by each of the challenges.

The problem of signaling metadata is complex. Whatever signaling approach is chosen multi-vendor interoperability is required, hence the need for standardization.

In general, it is desirable to solve simple problems with simple solutions and to make more complex solutions and mechanisms optional.

A productive path forward could be to divide and conquer: to clearly separate the problem of Service Function Path topology from the problem of metadata. In other words, to make a clear distinction between what we called the service path layer and the service function layer in section 4.8.

The Service Function Chaining working group could initially focus on the service path layer, i.e. the mechanism for steering packets through the right sequence of Service Functions in the Service Path.

Conveying contextual metadata could be viewed as optional, and not applicable to all service chains. Furthermore, the means to address the needs of the service function layer aren't necessarily the same as those required for the service path layer.

In this draft shy away from recommending one of the approaches as the best approach of all use cases. Different approaches may make sense for different use cases.

6. Security Considerations

The metadata signaling protocol must be secure in the sense that the authenticity (i.e. the validity of the content) and the authority (i.e. the source) must be guaranteed. Non-renumeration (i.e. encryption) of the metadata may or may not be a requirement.

7. IANA Considerations

None.

8. References

8.1. Normative References

None.

8.2. Informative References

[802.1Q] IEEE Std. 802.1Q-2011, Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks.

[draft-mahalingam-vxlan] M. Mahalingam, et al. "VXLAN: A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks.", draft-mahalingam-dutt-dcops-vxlan-08, February 3, 2014.

[draft-quinn-sfc-arch] P. Quinn, at al, "Service Function Chaining (SFC) Architecture", draft-quinn-sfc-arch-04, January 28, 2014.

[draft-quinn-sfc-nsh] P. Quinn, et al, "Network Service Header", draft-quinn-sfc-nsh-00, April 10, 2014.

[GTP] 3GPP TS 32.295 V6.1.0 (2005-06), 3rd Generation Partnership Project.

[in-band-control] Wikipedia, "In-band control", http://en.wikipedia.org/wiki/In-band_control

[out-of-band-control] Wikipedia, "Out-of-band control", http://en.wikipedia.org/wiki/Out-of-band_control

[redis.io] "Redis website", http://redis.io

[RFC959] J. Postal and J. Reynolds. "File Transfer Protocol (FTP)", RFC959, October 1985.

[RFC1661] W. Simpson, et al. "The Point-to-Point Protocol (PPP)", RFC1661, July 1994.

[RFC2205] R. Braden, et al. "Resource ReSerVation Protocol", RFC2205, September 1997.

[RFC2326] H. Schulzrinne, et el. "Real Time Streaming Protocol (RSTP)", RFC2326, April 1998.

[RFC2616] R. Fielding, et al. "Hypertext Transfer Protocol -- HTTP/1.1", RFC2616, June 1999.

[RFC2661] W. Townsley, et al. "Layer Two Tunneling Protocol L2TP", RFC2661, August 1999.

[RFC2774] H. Nielsen, et al. "An HTTP Extension Framework", RFC2774, February, 2000.

[RFC2784] D. Farinacci, et al. "Generic Route Encapsulation (GRE)", RFC2784, March 2000.

[RFC2865] C. Rigney, et al. "Remote Authentication Dial In User Service (RADIUS)", RFC2865, June 2000.

[RFC3031] E. Rosen, et al. "Multiprotocol Label Switching Architecture", RFC3031, January 2001.

[RFC3261] J. Rosenberg, et al. "SIP: Session Initiation Protocol", RFC3261, June 2002.

[RFC3931] J. Lau, et al. "Layer Two Tunneling Protocol - Version 3 (L2TPv3)", RFC3931, March 2005.

[RFC4271] Y. Rekhter, et al. "A Border Gateway Protocol 4 (BGP-4)", RFC4271, January 2006.

[RFC5321] J. Klensin. "Simple Mail Transfer Protocol", RFC5321, October 2008.

[RFC6648] P. Saint-Andre, et al. "Deprecating the "X-" Prefix and Similar Constructs in Application Protocols", RFC6648, June, 2012

[RFC6733] V. Fajardo, et al. "Diameter Base Protocol", RFC6733, October 2012.

[zeromq.org] "ZeroMQ website", http://zeromq.org

9. Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot.

The authors thank Ross Callon, Sankar Ramamoorthi, Gary Greenberg, Galina Pildush, Jaspal Kohli, Stuart Mackie, and James Connolly for their review and comments.

Authors' Addresses

Bruno Rijsman
Juniper Networks
brijsman@juniper.net

Jerome Moisand
Juniper Networks
jmoisand@juniper.net