INTERNET-DRAFT
Intended Status: Standard
Expires: November 2017

T. Herbert
Quantonium

May 12, 2017

Service token option for IPv6
draft-herbert-ipv6-service-token-option-00

Abstract

This document describes the service token option in IPv6. The service token option expresses to the network the services that should be applied to a packet. The service token allows network infrastructure to map packets to a service class for processing in the network. The contents of service tokens are not globally defined, their meaning and semantics are defined for individual networks. Applications request service tokens from their network provider for the services they wish to be applied. The service tokens are sent in either Destination options or a Hop-By-Hop options IPv6.

Table of Contents

1 Introduction

Modern provider networks, such as those being designed for the 5G mobile standard, offer a variety a services to applications for processing packets. Applications need to coordinate with the network get desired services applied to their packets. This coordination entails that the applications signal to the network in some manner what its characteristics are and what its requirements are for service. This document describes a method for an application to explicitly signal the network to request services.

1.1 Current solutions

In the current Internet, there is little coordination between hosts and the network to provide services based on characteristics of the application. Differentiated services provide some service classification, however it is lacking in richness of expression and pervasiveness in use. Some network operators have resorted to do Deep Packet Inspection (DPI) whereby HTTP is parsed to determine URL, content type, and other application level information the network is interested in. DPI is limited only to the application layer protocols that the device is programmed to parse and more importantly is being effectively obsoleted in the network due the pervasive use of TLS.

1.2 SPUD and PLUS solution

SPUD (Session Protocol Underneath Datagrams) and its successor PLUS (Path Layer UDP Substrate) proposed a UDP based protocol to allow applications to signal a rich set of characteristics and service requirements to the network.

SPUD has a number of drawbacks:

- SPUD is based on specific protocol over used over UDP. This requires applications to change to use a new protocol. In particular SPUD is incompatible with TCP which is the predominant transport protocol on the Internet.

- SPUD requires that intermediate nodes parse and process UDP payloads. Since UDP port numbers do not have global meaning [RFC7605] this introduces the possibility of misinterpretation and of silent data corruption if intermediate nodes modify UDP payloads. SPUD attempts to mitigate this issue with the use of magic numbers, however that can only ever be probabilistically correct.

- SPUD included connection tracking in the network. This problematic because:

- Not all communications have well defined connection semantics-- for instance a unidirectional data stream has no connection semantics at all.

- Connection tracking breaks multi-homing; it assumes that all packets of a connection in both directions are seen by the node doing connection tracking. Firewalls for instance require all packets for a flow to always go through the same device in both directions. This disallows flexibility and optimized traffic flow that a multi-home network device affords.

- The meta data information in SPUD would have global definition. This problematic because:

  - Application specific information could be leaked to unknown and untrusted parties.

  - Establishing a specification on what data should be conveyed in SPUD will be difficult. Different service providers may want different pieces of information, applications may also have different ideas about what information is safe to make visible.

1.3 Emerging use cases

In a typical client/server model of serving content, end host clients communicate with Internet services. The clients are typically user devices that are connected to the Internet through a provider network. In the case of mobile devices, such as smart phones, the devices are connected to the Internet through a carrier network. Content providers (web servers) tend to be more directly connected to the Internet, the largest of which can connect at exchange points.

Provider networks can be architected to provide different services and levels of services to their users based on characteristics of applications. For example, a mobile carrier network can provide different latency and throughput guarantees for different types of content. A network may offer different services for optimizing video: streaming an HD movie might need high throughput but not particularly low latency; a live video chat might have lower throughput demands but have low latency requirements.

The emerging 3GPP standard for 5G defines a set of mechanisms to provide a rich array of services for users. These mechanisms employ Network Function Virtulization (NFV), Service Function Chaining (SFC), and network slices that divide physical network resources into different virtualized slices to provide different services. To make

use of these services the applications running in UEs (User Equipment) will need to indicate desired services of the RAN (Radio Access Network). For instance, a video chat application may request bounded latency that is implemented by the network as a network slice; so packets sent by the application should be mapped to that network slice.

Note that an application service applies to both packet sent by UE and those sent from a peer towards the UE. For the latter case, the network needs to be able to map packets sent from hosts on the Internet to the services requested receiving application.

2 Architecture

The figure below illustrates an example network path between two hosts on the Internet. Note that each host connects to the Internet via a provider network and provider networks are connected in the Internet by transit networks.

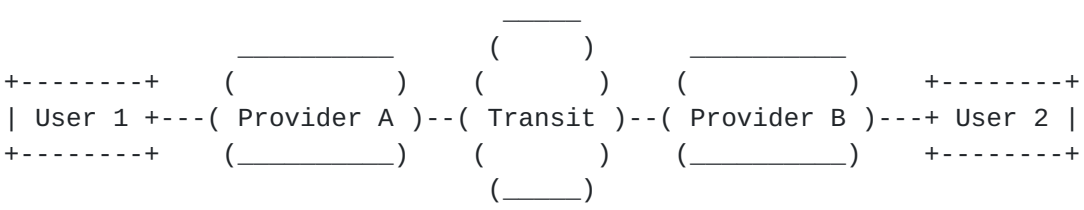


Figure 1

Within each provider network, services may be provided on behalf of the users of the network. In the example above, Provider 1 may provide services and service agreements for users in its network including User 1; and likewise Provider B can provide services to users in its network including User 2. Transit networks service all users and don't typically provide user specific services or service differentiation.

Services provided by different provider networks may be very different and dependent on the implementation of the network as well as the policies of the provider.

Based on this model, services and service differentiation can be considered local to each provider of the network. This document describes a mechanism whereby each user and application can request from its local provider the services to be applied to its traffic. The request is made to "service token provider". The contents of the request describe the service requirements that application desires. The service token provider responds with a "service token" that the application sets in its packets. When a packet is sent by the application with a service token, the token is interpreted in the

provider network to map the packet to the appropriate service. The token is only relevant to the provider network, to both the application and nodes outside of the provider network the token is an opaque value.

To facilitate service mapping in the reverse direction for a flow, that is packets sent from a peer host, peer hosts reflect the service token without modification or interpretation.

The use of service tokens may be symmetric for a connection so that each peer requests service. Therefore packets may contain two service tokens: one that is set by the sending host to signal its local provider network, and the other is the reflected service token that is a signal to the provider network of the peer endpoint.

Service tokens are scoped values, they only have meaning in the network for which they were defined. The destination option includes an autonomous system value with each service option to indicate which network interprets the value. The format, meaning, and interpretation of service tokens is network (autonomous system) specific. By mutual agreement, two networks may shared the policy and interpretations of service tokens. For instance, there could be an agreement between two autonomous systems to interpret each others service tokens or to use a common format.

2.1 Example packet flow

Referencing the diagram in figure 1, consider that User 1 is communicating with User 2 and wishes to have some service provided by its local network (Provider 1). The flow of events may be:

1. User 1 wishes to create a live video chat with User 2

2. User 1 makes a service token request from service token provider of Provider A that describes the video application and may include detailed characteristics such as resolution, frame rate, latency, etc.

3. Service token provider provides a token

4. Application sends packets with the service token set for the video chat

5. Provider A interprets the service token and applies the appropriate services to the packets

6. Packets traverse transit networks and Provider B network, the service token is ignored

7. Packet is received at User 2. The service token is saved in the context for the video session

8. User 2 sends video chat packets to User 1. The service token received from User 1 is reflected in these packets

9. Packets traverse Provider B network and transit networks

10. Provider A interprets the reflected service token and applies appropriate services to the packets

11. Packets are received at User 1

2.2 Requirements

The requirements for this solution are:

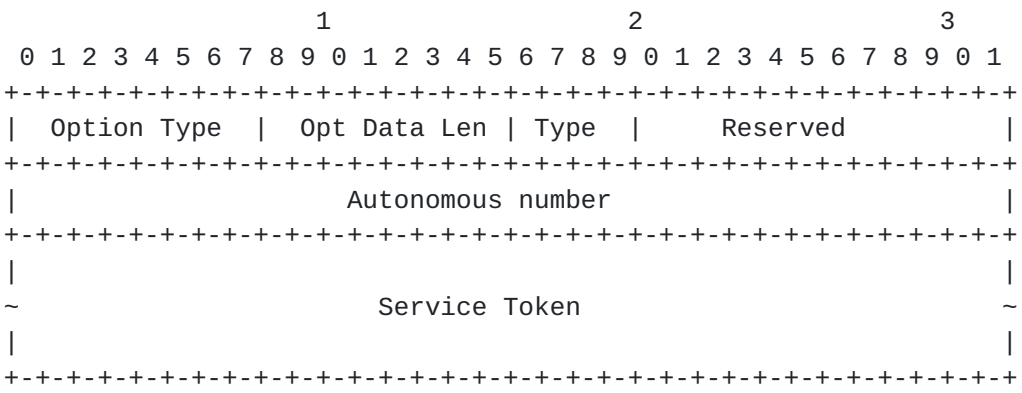
- Service tokens should be connectionless
- Service tokens should work in multi-homed environments
- Service tokens should indicate the network for which they are applicable
- Outside of the relevant network the service token should be opaque and no application specific information should be derivable from the token
- Service tokens should work with any transport protocol
- Service tokens should minimize the changes to an application. Their use should should be an "add-on" to the existing communications of an application
- Service tokens should prevent spoofing and other misuse that might result in illegitimate use of network services or denial of service attack
- Deep packet inspection is not needed
- Service tokens must allow services to be applied in the reverse path. In a client/server application it is often the packets in the reverse path that require the most service (for instance if a video is being streamed to a client).
- A fallback exists if packets with extension headers are dropped

3 Packet format

The service token is sent as Destination option or a Hop-By-Hop option.

3.1 Option format

The same option types and format are used for both a Destination and Hop-By-Hop option. The format of the option is:



Fields:

- ICMP Option type: 0x4F or 0x6F
- Opt Data Len: Length of the option data field. This is 6 + the length of the service token field
- Type: Indicates the type and action of the service token. This is one of:
  - 0x0: Service token from origin, don't reflect at receiver
  - 0x1: Service token from origin, reflect at receiver
  - 0x2: Reflected service token
  - 0x3-0xf: Reserved
- Autonomous number: AS for for the network in which the service token is valid. A value of zero indicates the locally attached network of the origin.

3.2 Option types

The service token may be sent as a Destination option or Hop-By-Hop option. The rationale is that the two methods exhibit different drop rates. For instance, [RFC7872] indicates that the drop rate to Alexa's Top 1M Sites for Destination options was 10.91%, whereas Hop-

By-Hop options have a drop rate of 39.03%

The are two option numbers requested for the service token option: 0x4F and 0x6F. The latter allows modification. This would be used in situations where the network nodes needed to modify the option with new information. If the option is modifiable it should be a Hop-By-Hop option.

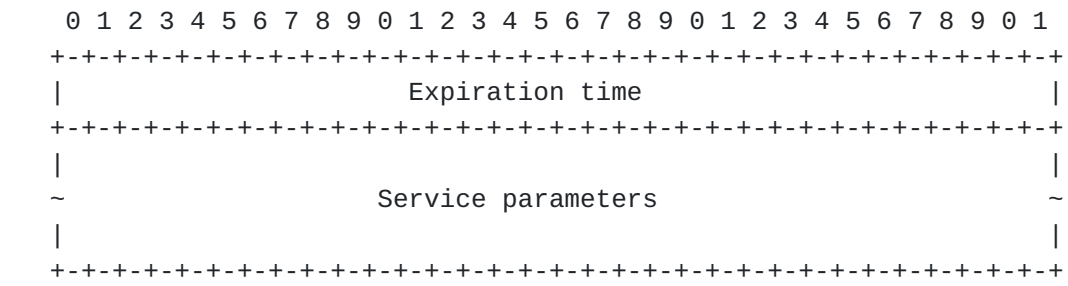
3.3 Service token

The service token contains service parameters that describe the desired services as well as additional fields that would be used to provide private and integrity.

The format of the service token is defined by the network in which the service token originates. The service token should be obfuscated or encrypted for privacy. It should also be resistant to spoofing when an attacker uses a service token seen on other flows to illegitimately have service applied to its packets.

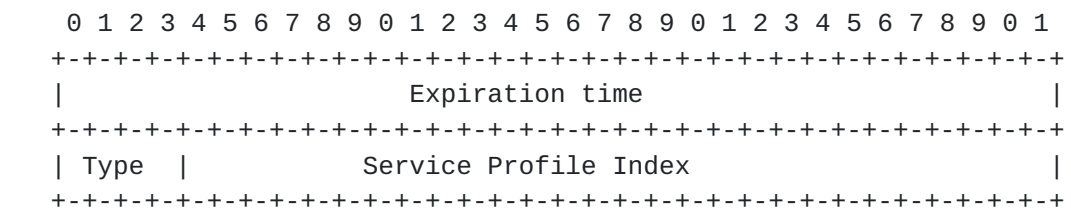
It is recommended that service tokens are encrypted and each token has an expiration time. For instance, a service token may be created by encrypting the token data with an expiration time and using the source address, destination address, and a shared key as the key for encryption.

For example, the security token for a network may have the format:



Where the expiration time is in a format that are understood by the provider network nodes which maintain synchronized time. The Service parameters are relevant to network nodes and describe the services to be applied. The service parameters could simply be a set of flags for services, an index to a service profile known by the network nodes, or possibly have more elaborate structure that could indicate numerical values for characteristics that have a range. The service parameters could also include a type field to allow a network to define different representations of service parameters.

A simple service token containing a service protocol index might be:



4 Operation

4.1 Origin application operation

An application that wishes to request services first requests a service token from a service token provider. The application request could be in the form of an XML structure with a canonical format (the definition is outside the scope of this document). The application makes a request to a service token provider for the local network. This could be done a web service using HTTP PUT/GET. Internally in the host the security token provider might be accessed through a library that interfaces to a service token provider daemon that in turn arbitrates requests between the applications and the network infrastructure.

When the service token provider returns a token, the application sets the token as Hop-By-Hop option or Destination option as indicated. This is typically done by setting socket option on a socket (in the case of TCP) or by indicating the option in the ancillary data when sending on a socket (in the case of UDP).

The service token provider should return an expiration time with the service token. An application can use the token until the expiration time, at which point it must request a new service token. The service token itself is opaque to the application and the application should no attempt to interpret any meaning from it.

4.2 Origin network processing

When a packet with a service token enters a network referred to by the autonomous system number it should be processed. The service token is decrypted if necessary and the expiration time is checked. If the service token is valid then the packet is mapped to be processed by the requested services. For instance, in a 5G network the packet may be forwarded on a network slice for the characteristics the application has requested (real-time video for instance).

Note that there are two points at which the provider needs to process the service token: when a local user sends a packet into the provider network, and when a packet from an external network enters the

provider's network with a reflected security token. Once the service token is processed and mapped to the network's mechanism it should not need further examination.

4.3 Peer processing

When an application receives a packet with a service token whose type is "from origin and needs to be reflected", it should save the service option in its flow context and reflect it on subsequent packets. When the application reflects the option is copies the whole option and only modifies the type to indicate a reflected option. The application continues to reflect the service token until a different one is received from the origin or a packet without a service token option is received for the flow.

4.4 Handling dropped extension headers

The downside of using IPv6 extension headers on the Internet is that they are unreliable. Some intermediate nodes will drop extension headers with rates described in [RFC7872].

4.4.1 Mitigations for dropped extension headers

There are some mitigating factors for this problem:

- A provider network that implements the service token protocol defined in this document would need to ensure that the service token option is usable within the network
- Transit networks are less likely to arbitrarily drop packets with extension headers
- Many content providers, especially the larger ones, may be directly connected to the Internet. For example, front end web servers may be co-located as exchange points.

4.4.2 Fallback for dropped extension headers

Since the possibility that extension headers are dropped cannot be eliminated, a fallback is included for use with service tokens.

When an application connects to a new destination for which it has no history about the viability of extension headers, it can perform a type of Happy Eyeballs probing. The concept is for the to send a number of packets with and without the service token. The application should observe whether packets with service tokens are being dropped.

There are a few possible outcomes of this process:

- A packet with a service token is dropped and an ICMP for extension headers [ICMPEH] processing limits is received. This is a signal that extension headers are not viable and should not be used for the flow.

- A packet with a service token is dropped and no ICMP error is received. This is a signal that extension headers may not be usable. If such drops are observed for all or a significant fraction of packets and there are no drops for packets that were sent without the service token, then extension headers should be considered not viable for the flow.

- Packets with service tokens are not being dropped, however service tokens are not being reflected. This is a signal that the peer application does not support reflection. Service tokens may be sent, however they are only useful in the outbound path.

- Packet with service tokens are not being dropped and service tokens are being reflected. Service tokens are useful in both direction.

5 Implementation considerations

5.1 Origin applications

Existing client applications can be modified to request service tokens and set them in packets. The kernel may need some small changes or configuration to enable an application to specify the option for its packets.

The interface to the service provider would likely be via a library API.

Using the BSD sockets interface, for a connected socket (TCP, SCTP, or connected UDP socket) the destination option can be set on the socket via the setsockopt system call. For an unconnected socket (UDP) the service token option can be set as ancillary data in the sendmsg system call.

Extension header probing, described in section 4.4.2, could be implemented in the kernel for a connection oriented transport protocol such a TCP. For connectionless protocols, probing could be handled by an application library.

5.2 Reflection

To perform reflection of the service option, a server must be updated. In the case of a connected socket (TCP, SCTP, or a connected UDP socket) this can be done as relatively minro change to the kernel

networking stack which would be transparent to application. For unconnected UDP, an application could receive the security token as part of the ancillary data in recvmsg system call, and then send the security option in a reply using ancillary data in sendmsg.

6 Security Considerations

Service token options may be visible to the Internet including untrusted and unknown networks in the path of sent packets. As such the token should be encrypted or obfuscated by the origin network.

7 IANA Considerations

IANA is requested to assigned the following Destination and Hop-By-Hop options:

Hex Value	Binary value act chg rest	Description	Reference
0x0F	00 0 01111	Service Token	This document
0x2F	00 0 01111	Modifiable Service Token	This document

IANA is requested to set up a registry for the Service Token option types. These types are 4 bit values. New values for control types 0x3-0xf are assigned via Standards Action [RFC5226].

Service Token option type	Description	Reference
0x0	Token from origin and don't reflect	This document
0x1	Token from origin and reflect	This document
0x2	Token not from origin and reflect	This document

8 References

8.1 Normative References

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

8.2 Informative References

[RFC7605] Touch, J., "Recommendations on Using Assigned Transport Port Numbers", BCP 165, RFC 7605, DOI 10.17487/RFC7605, August 2015, <http://www.rfc-editor.org/info/rfc7605>.

[RFC7872] Gont, F., Linkova, J., Chown, T., and W. Liu, "Observations on the Dropping of Packets with IPv6 Extension Headers in the Real World", RFC 7872, DOI 10.17487/RFC7872, June 2016, <http://www.rfc-editor.org/info/rfc7872>.

[ICMPEH] Herbert, T., "ICMPv6 errors for discarding packets due to processing limits", draft-herbert-6man-icmp-limits-00.txt

Author's Address

Tom Herbert
Quantonium
Santa Clara, CA
USA

Email: tom@herbertland.com