# Firewall and Service Tickets (FAST)

**Tom Herbert**

SiPanda
USA
tom@sipanda.io

## Abstract

Emerging network access architectures and technologies offer a rich array of network services that could greatly benefit end users. In practice, realizing these benefits has proven difficult. The problem is a lack of coordination between applications, the host OS, and the network infrastructure to provide end-to-end services. Firewall and Service Tickets (FAST) is a solution that facilitates coordination among the various players in communications. The basic idea is that applications signal the network for the services they want applied to packets. This signal is encoded in the form of a "ticket" that indicates the network services that the network applies to packets. Applications request tickets from a ticket agent in the network for the desired services, issued tickets are attached to packets, and tickets are processed by network elements to provide the requested services for each packet. FAST enables hosts and the network to *work together* to solve end user problems.

## Keywords

QoS, differentiated service, firewalls, Hop-by-Hop Options,

## Introduction

Firewall and Service Tickets (FAST) [1] is a facility to allow an application to signal to the network requests for admission and services for a packet. A *ticket* is data attached to a packet by the source host that is inspected and processed by intermediate nodes in a network. Tickets express a grant or right for packets to traverse a network or have services applied. FAST facilitates coordination among applications, hosts OSes, and network nodes for the purposes of providing rich and fine grained network services for the benefit of users. Heretofore, hosts and networks haven't really worked together to solve users problems, and current solutions for differentiated services and QoS tend to be limiting and restrictive.

The core idea of FAST is that hosts signal the network for the services to be applied on a per packet basis. Signals are information attached to packets that contain requests for service. In FAST, signals are encoded in *tickets*. Tickets are data attached to packets in Hop-by-Hop options. A ticket encapsulates the granted services in a concise form. An application requests tickets for admission or services from a ticket agent in their local network. The agent issues tickets to the application which in turn attaches these to its packets. In the forwarding path, intermediate network nodes interpret tickets and apply requested services.

## Alternative approaches

This section considers some current techniques and proposals for signaling the network for services.

Stateful firewalls and proxies are the predominantly deployed techniques to control access to a network and map packets to services. They have caused a number of problems:

- They require parsing over transport layer headers .
- They are limited to work only with a handful of protocols.
- They break multi-homing and multi-path.
- They break end-to-end security. For instance, NAT breaks the TCP authentication option.
- They are single points of failure and can be bottlenecks.
- Application characteristics need to be inferred from packets which can be imprecise or incorrect.

PLUS (Path Layer UDP Substrate) [2] proposed a UDP based protocol to allow applications to explicitly signal a rich set of characteristics and service requirements to the network. PLUS had a number of drawbacks:

- It requires UDP, and wouldn't work with TCP.
- Intermediate nodes parse payloads based on matching port numbers to applications risking misinterpretation.
- PLUS included stateful flow tracking in the network which leads to problems similar to those of stateful firewalls.
- PLUS could leak sensitive application information.

Segment routing [3] is a recently defined technique that proposes using an IPv6 routing header to source route packets through a network. This allows "network programming" where packets can visit various nodes towards the destination, each of which may apply some Network Function Virtualization (NFV). Segment routing as a form of network signaling has several drawbacks:

- Segment routing is intended to be confined to a segment routing domain, not for use over the Internet.
- The protocol is verbose. Each SID is sixteen bytes which adds up to considerable overhead.
- The segment list is plain text that sensitive internal information may be leaked.
- Segment routing only conveys routing for the forward path of the packet and not routing for the return path.

Both IPv4 and IPv6 have specified a field in the IP header for signaling quality of service (QoS) to the network. In IPv4 this was referred to as the Type of Service (TOS), and in IPv6 it is called Traffic Class. These fields have been overloaded in time to hold differentiated services (diff-serv) values. Differentiated services provides an IP layer means to classify and manage traffic, however it is lacking in richness of expression and a ubiquitous interface that allows applications to request service with any granularity. Diff-serv is useful in closed networks where all parties can be trusted, but a general Internet diff-serv lacks security and uniformity in how it's being set to be useful.

Some network devices perform Deep Packet Inspection (DPI) into the application data to classify packets to determine whether to admit packets or what services to apply. For instance, HTTP is commonly parsed to determine URL, content type, and other application related information. DPI is only effective with the application layer protocols that a device is programmed to parse. More importantly, application level DPI is being effectively obsoleted in the network due the pervasive use of Transport Layer Security (TLS).

## Architecture of FAST

FAST allows network providers to offer custom network services to their users. In particular, FAST does not endeavor to create a global infrastructure across the Internet to provide or manage network services. This is motivated by the common *dumbbell* topology of end to end communications over the Internet. In the dumbbell topology, illustrated in Figure 1, two communicating end hosts connect to the Internet via local provider networks and provider networks connect to transit networks to communicate across the Internet.

Within each provider network of the dumbbell topology, network services may be provided on behalf of the users in the local network. Referring to Figure 1, Provider A may provide services and service agreements for users in its network including User 1 and User 2; and likewise, Provider B can provide services to users in its network including User 3. Transit networks don't typically provide user specific services or service differentiation, that is transit networks may be considered the "open Internet".

In FAST, each provider network can issue tickets to local hosts in its network. The network that issued a ticket is called the *origin network* for the ticket, and an *origin ticket* is one that was issued by the network processing a ticket. Tickets are scoped so that only the network nodes in the origin network of a ticket interpret it and apply requested services.

In Figure 1, User 1 and User 2 reside in the same provider network; each can request tickets for network services to be provided in communication between the two users. User 1 and User 3 are in different provider networks. User 1 and User 3 can each request tickets from their local network to be applied in the *forward* path. When User 1 sends packets to User 3, tickets can be used for services while packets are in Provider A's network; and likewise when User 3 sends packets to User 1, tickets can be used for services while packets transit Provider B's network.

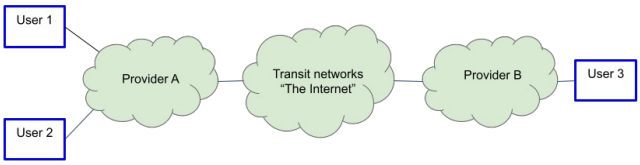

Figure 1. Example of a dumbbell network topology

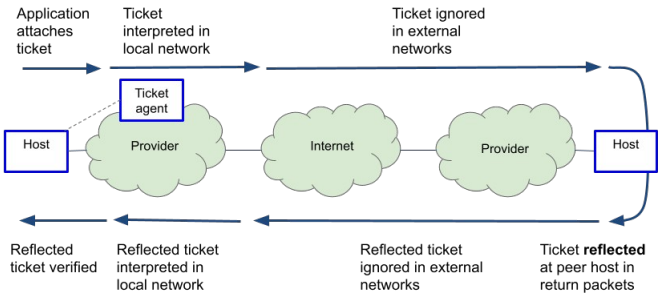

Figure 2. End to end ticket flow with reflection

In order to apply services in the *return path*, tickets may be *reflected* (Figure 2). For instance, when User 3 receives a packet with a an origin ticket sent by User 1, it can reflect the ticket and send it back to User 1; when the reflected ticket enters Provider A's network it can be interpreted and services applied to the packet for the rest of its journey to User 1. Ticket reflection is symmetric so that User 1 could reflect origin tickets sent by User 3 and services are applied to packets in the return path to User 3 as packets traverse Provider B's network. The end to end flow of a ticket with reflection is illustrated in Figure 2.

### Ticket properties

A ticket is scoped such that only particular on-path nodes in its origin network process and act on the ticket. The scope can be enforced by encrypting the ticket so that only authorized network nodes are able to decode it. Encryption serves as a security mechanism to limit the exposure of ticket data and to minimize the plain text in packets. Encryption, in combination with ticket authentication, prevents forgery and modification, hides details about requested services, hides information about applications and users, and enforces non-transferable tickets.

Tickets may include an expiration time such that they are only useful for some period of time. For instance, if a ticket is attached to the packets of a flow for the purpose of requesting network services, an associated expiration time would allow the infrastructure to limit the use of the ticket for a certain period of time and prevent unlimited reuse of the ticket.

An important property of tickets is that they are stateless inside the network; this facilitates multi-homing where routers in different paths for a flow would be able to decode and process host to network signals in packets associated with a flow. While tickets do not directly convey connection state, they may still be associated with a transport layer flow. For instance, a host may request tickets from a ticket agent to attach to packets of a particular flow. When an on-path element processes the ticket, it applies the services without regard to transport layer state.

Host to network signals are inherently uni-directional. In order for a source host to affect services on the return path of a flow, "signal" reflection" may be employed. The idea is that a signal can be sent with a "reflect" attribute. At a peer host, the signal can be reflected in reply packets to affect services for packets in the return path.

Tickets are sent in IPv6 Hop-by-Hop Options. The benefit of Hop-by-Hop Options is that tickets can be used with any transport protocol. There are some known drawbacks with Hop-by-Hop Options, migrations for these are discussed below.

A ticket should be obfuscated or encrypted for privacy so that only the local network can interpret it. It should be resistant to spoofing so that an attacker cannot illegitimately get service by applying a ticket seen on other flows.

### Example use: Network services in 5G

5G, the mobile standard being developed by the 3rd Generation Public Partnership (3GPP) [4], provides a good example of applying FAST to enable use of network services.

A key feature introduced by 5G is network slicing [5]. A network slice is an overlay virtualized network run over a physical network with its own operational characteristics. In combination with Network Function Virtualization (NFV) [6], network slices provide the foundation for a rich set of network services for low latency, high throughput, optimized mobile routing, etc. To fully apply network services to packets, network nodes need to deduce the service characteristics of an application based on the packets observed and apply appropriate services. This process is called *service mapping.* Service mapping happens today, however the techniques used are ad hoc, imprecise, and inferred.

In FAST, applications explicitly request services to be applied to the application's packets. In a 5G network, this would entail that applications running in UEs (User Equipment) indicate desired services to be provided by the RAN (Radio Access Network) and core network. An example of using FAST in a 5G network is illustrated in Figure 3. Figure 4 illustrates the processing flow of of tickets in a 5G network.

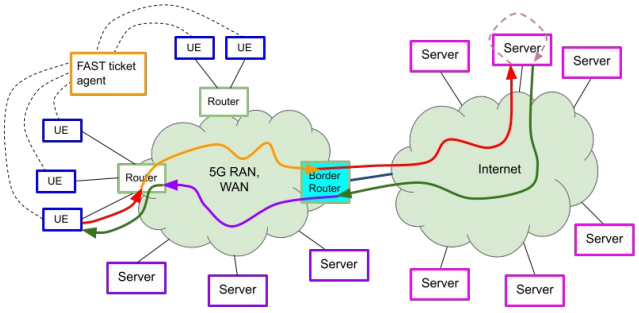


Figure 3. Example FAST path processing and topology in a 5G network

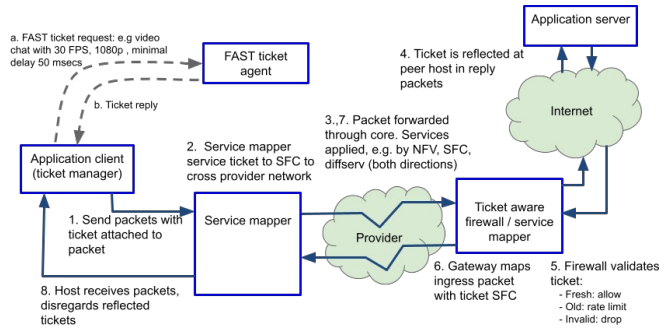


Figure 4. Example of using FAST in a 5G network for optimizing video chat

Referring to the Figure 3, suppose a user starts a video chat application that connects to a server on the Internet. The video chat application might request a ticket from the local ticket agent for network services for the video chat. The request might be for a service class like "video chat service", or could specify service characteristics such as expected latency, jitter requirements, or video frame rate. The issued ticket is attached to packets for the video chat and services are applied while the packet is in the local network (the orange arrow of the path). In this example, the first hop router of the UE may route packets over a network slice that provides the requested services. When packets exit the provider network into the Internet, services are no longer applied but the ticket is still attached to packets (the red arrow in Figure 2).

At the server, packets are received and the attached ticket is saved in the context for the connection. When packets are sent back to the client, the server reflects the ticket by setting it in packets. Packets traverse the Internet without services being applied (the green arrow in Figure 2) When the packet enters the RAN network, the ingress router processes the reflected ticket and routes the packet over a network slice for the services. The packet traverses the provider network with the services applied (the purple arrow in Figure 2)

### Protocol and operation

This section describes the protocol for FAST and ticket operations.

#### Hop-by-Hop option format

Tickets are encoded in IPv6 Hop-by-Hop options [7] as illustrated in Figure 5.

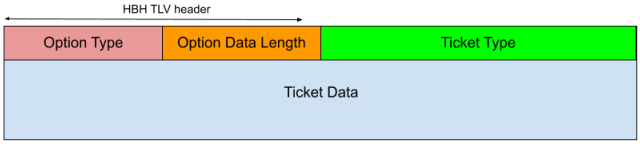


Figure 5. Format of a ticket in a Hop-by-Hop option

The fields of the Hop-by-Hop option containing a ticket are:

- Option Type: Type of Hop-by-Hop option. There are two possible types for FAST ticket options: an unmodifiable and a modifiable variant.
- Opt Data Len: Length of the option data field. The option data is comprised of the Pr, Ticket Type, and Ticket Data fields.
- Pr: Indicates the origin and reflection properties of the ticket. Possible values are:
  - 0x0: Origin ticket not reflected. Don't reflect at the destination host.
  - 0x1: Origin ticket to be reflected. Ticket is requested to be reflected by the destination host.
  - 0x2: Reflected ticket. The ticket was reflected by a destination host and is being returned to the origin source host.
  - 0x3: Reserved
- Ticket Type: The type and format of the ticket. This value is used by nodes in the origin network to interpret the rest of the ticket data. Values for this field are specific to the network that issues the ticket. The type is an IANA managed number space. A type of 0 indicates a "null" ticket that isn't to be processed by receivers.
- Ticket Data: Contains the ticket data that describes the services applied. The format and semantics of the data are determined by the Ticket Type.

### Ticket Data

It is expected that tickets are encrypted and each ticket has an expiration time (Figure 6). For instance, a ticket may be created by encrypting the ticket data with an expiration time and using the source address, destination address, and a shared key as the key for encryption. The operative part of the ticket that describes the service may have different types of data. For instance, a set of flags could be used, a list of service values, or a profile index into a table that describes a set of services. A ticket with an expiration time and service profile index might have the format shown in Figure 7.

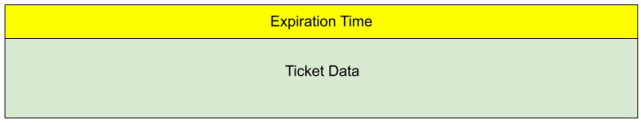


Figure 6. Format of a ticket with an expiration time. Ticket Data is variable length with a format determined by the ticket type

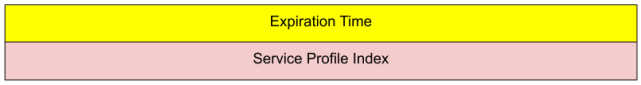


Figure 7. Format of a ticket with an expiration time with a service profile index. The service profile index could be a 32 bit number used to index a service parameters table

## Operation

Existing client applications can be modified to request tickets and set them in packets. It is also possible for the OS to set FAST tickets on behalf of an application that can't be changed or recompiled. The kernel may need some small changes or configuration to enable an application to specify the FAST Hop-by-Hop option for its packets (see section below). In BSD sockets this can be done by a *setsockopt* system call or in ancillary data of the *sendmsg* system call.

An application that wishes to use network services first requests tickets from a ticket agent. The request could be in the form of an XML structure with canonical elements. A request could be sent via a web service using RESTful APIs [8]. Internally in the host, the ticket agent might be accessed through a library that interfaces to a ticket daemon that in turn arbitrates requests between applications and a ticket agent in the network.

Service mappers need to parse and process tickets in the fast data-path. This entails an implementation that does efficient IPv6 header processing. Tickets need to be parsed, validated or decrypted, looked up, and interpreted quickly.

To perform ticket reflection, servers must be updated. In the case of a connected socket (TCP, SCTP, or a connected UDP socket) this is a relatively minor change to the kernel networking stack which would be transparent to applications. For unconnected UDP, an application could use ancillary data in *recvmsg* and *sendmsg* to receive and reflect tickets.

Tickets facilitate fine grained policies and "per use charging" of services. There are three points at which policy and charging can be applied: 1) at ticket request time, 2) when a client sends a packet with a ticket, and 3) when a network ingress node receives a packet with a reflected ticket

At ticket request time, policy can be applied to determine if the network can or should provide the services being requested. Ticket requests are authenticated and the requestor's identity is known. A database can be maintained that holds the per user policies, resource limits, and current resource utilization as input to a policy decision for issuing a ticket.

Each time a ticket is seen on the network it can be accounted for. The two cases, when a ticket is sent from a client or a ticket is reflected by a server, should be accounted for separately since the tickets sent directly from the client are a bit more trustworthy. A rogue peer, for instance, could attempt a narrow Denial of Service (DOS) attack by flooding the flow with fake packets using the same ticket.

Per use accounting is done at the service mappers. Each occurrence bumps a counter. Aggregated counts are periodically sent to a centralized accounting system that correlates the use of tickets across the service mappers. Based on the resulting data, users can be charged precisely for the services they actually used.

## Hop-by-Hop Options drops

RFC2460 [9] required that all intermediate nodes in a path process Hop-by-Hop Options. Some routers deferred processing of Hop-by-Hop Options to the software slow path, others ignored them, and still others elected to summarily drop all packets containing Hop-by-Hop Options. A related issue was that the number of Hop-by-Hop Options in a packet was only limited by the MTU for the packet. The lack of limits, combined with the requirement that nodes must skip over unknown options (when two high order bits in the option type aren't set), creates an opportunity for DOS attacks by sending long lists of unknown Hop-by-Hop options. These issues have severely impeded the deployment of Hop-by-Hop Options on the Internet ([10], [11).

There is ongoing work to fix, or at least mitigate, the deployability problems of Hop-by-Hop options:

- RFC8200 [7] specifies that intermediate nodes MAY ignore Hop-by-Hop options. There is no concept of a Hop-by-Hop option that must be processed by all nodes, the current assumption in defining any option is that it may be processed by only some nodes in the path, or even none at all. Allowing nodes to ignore options they're not interested in, instead of just dropping the packets, preserves the usability of Hop-by-Hop across the whole path.
- Internet Draft ietf-6man-hbh-processing [12] modifies the processing of Hop-by-Hop options described in [RFC8200] to make processing of the IPv6 Hop-by-Hop Options header practical. In particular, this clarifies the expectation that Hop-by-Hop Options should not be processed in the slow path and that new Hop-by-Hop options are designed to always be processed in the fast path.
- Internet Draft ietf-6man-eh-limits [13] specifies that intermediate nodes that process Hop-by-Hop Options may set and apply configurable limits on Hop-by-Hop Options processing. For instance, one limit is for the number of options that are processed; if the limit is exceeded then options processing is terminated and the packet is forwarded without any ill-effects. The use of limits is optional and while specific default limits are recommended, there are no specific "hard" limits that must be enforced.
- Internet Draft herbert-eh-inflight-removal [14] describes a protocol to remove Hop-by-Hop Options headers from packets in flight. This could be applied in host to network signaling by arranging that the last router that processes a signal in a Hop-by-Hop option removes the Hop-by-Hop Options header from the packet. Removing the Hop-by-Hop Options headers increases the probability that a packet won't be dropped without substantial loss of functionality.

## Linux kernel support

The Linux kernel needs some changes to support extension headers and Hop-by-Hop options for FAST. A draft patch set was posted to Netdev mailing list [15].

The functionality of these patches includes:

1. Allow modules to register support for Hop-by-Hop and Destination options. This is useful for development and deployment of new options.
2. Allow non-privileged users to set Hop-by-Hop and Destination options for their packets or connections. This is especially useful for options like Path MTU and IOAM options where the information in the options is both sourced and consumed by the application.
3. In conjunction with #2, validation of the options being set by an application is done. The validation for non-privileged users is purposely strict, but even in the case of a privileged user, validation is useful to disallow allow application from sending ill-formed packets (for instance now a TLV could be created with a length exceeding the bound of the extension header)
4. Consolidate various TLV mechanisms. Segment routing should be able to use the same TLV parsing function, as should UDP options when they come into the kernel.
5. Use a TLV type lookup table to make option lookup on receive is O(1) instead of list scan (O(N)).
6. Allow setting specific Hop-by-Hop and Destination options on a socket. This would also allow some options to be set by application and some might be set by kernel.
7. Allow options processing to be done in the context of a socket. This will be useful for FAST and Path MTU (PMTU) options.
8. Allow experimental IPv6 options in the same way that experimental TCP options are allowed.
9. Support a robust means of extension header insertion. Extension header insertion is a controversial mechanism that some router vendors are insisting upon. The way they are currently doing it breaks the stack (particularly ICMP and the way networks are debugged). With proper support we can at least mitigate the effects of the problems being created by extension header insertion.
10. Support IPv4 extension headers. This allows Hop-by-Hop Options and other extension headers to be used with IPv4 with the same semantics of their use in IPv6.

## Conclusion

Firewall And Service Tickets is a facility that allows hosts to signal the network for requesting services and admission of packets. FAST promotes the end-to-end principle, net neutrality, and clean protocol layering. It also leverages forward looking features of the IPv6 protocol. The ultimate goal is to spur innovation in network services to benefit users.

## Acknowledgments

The author would like to thank the Netdev Society board, Netdev community, and IETF for facilitating this work.

### References

1. Herbert, T., "Firewall and Service Tickets", Work in Progress, Internet-Draft, draft-herbert-fast-07, 7 October 2023 https://datatracker.ietf.org/doc/html/draft-herbert-fast-07.

2. Tramell, B. and M. Kuehlewind, "Path Layer UDP Substrate Specification", December 2016, https://datatracker.ietf.org/doc/draft-trammell-plus-spec/00/.

3. Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <https://www.rfc-editor.org/info/rfc8402>.

4. "5G; System Architecture for the 5G System", September 2018, https://www.etsi.org/deliver/etsi_ts/123500_123599/12351/15.03.00_60/ts_123501v150300p.pdf.

5. Wikipedia, "5G network slicing", https://en.wikipedia.org/wiki/5G_network_slicing.

6. Wikipedia, "Network function virtualization", https://en.wikipedia.org/wiki/Network_function_virtualization.

7. Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, https://www.rfc-editor.org/info/rfc2460.

8. RedHat, "What is a REST API", May 2020, https://www.redhat.com/en/topics/api/what-is-a-rest-api.

9. Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, https://www.rfc-editor.org/info/rfc2460.

10. Gont, F., Hilliard, N., Doering, G., Kumari, W., Huston, G., and W. Liu, "Operational Implications of IPv6 Packets with Extension Headers", RFC 9098, DOI 10.17487/RFC9098, September 2021, https://www.rfc-editor.org/info/rfc9098.

11. Huston, G., "IPv6 Extension headers revisited", October 2022, https://blog.apnic.net/2022/10/13/ipv6-extension-headers-revisited/.12. Hinden, R. M. and G. Fairhurst, "IPv6 Hop-by-Hop Option Processing Procedures", Work in Progress, Internet-Draft, draft-ietf-6man-hbh-processing-12, 21 November 2023, https://datatracker.ietf.org/doc/html/draft-ietf-6man-hbh-processing-12.

13. Herbert, T., "Limits on Sending and Processing IPv6 Extension Headers", Work in Progress, Internet-Draft, draft-ietf-6man-eh-limits-10, 23 November 2023, https://datatracker.ietf.org/api/v1/doc/document/draft-ietf-6man-eh-limits/>.

14. Herbert, T., "Infight Removal of IPv6 Hop-by-Hop and Routing Headers", Work in Progress, Internet-Draft, draft-herbert-eh-inflight-removal-01, 2 October 2023, https://datatracker.ietf.org/doc/html/draft-herbert-eh-inflight-removal-01.

15. Herbert, T., "ipv6: Extension header infrastructure", https://lkml.kernel.org/netdev/1570139884-20183-1-git-send-email-tom@herbertland.com/.