Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: January 16, 2014

T. Eckert, Ed.
R. Penno
A. Choukir
C. Eckel
Cisco Systems, Inc.

July 15, 2013

A Framework for Signaling Flow Characteristics between Applications and the Network

draft-eckert-intarea-flow-metadata-framework-01

Abstract

This document provides a framework for communicating information elements (a.k.a. metadata) in a consistent manner between applications and the network to provide better visibility of application flows, thereby enabling differentiated treatment of those flows. These information elements can be conveyed using various signaling protocols, including PCP, RSVP, and STUN.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document provides a framework for communicating information elements (a.k.a. metadata) in a consistent manner between applications and the network to provide better visibility of application flows, thereby enabling differentiated treatment of those flows. These information elements can be conveyed using various signaling protocols, including PCP, RSVP, and STUN.

The framework is built around the definition of four key components:

1. A set of application independent information elements (IEs)

2. An encoding of these IEs that is independent of the signaling protocol used as transport

3. Usages of these IEs to support various transactional semantics

4. A mapping of one of more to these usages to an initial set of signaling protocols, including PCP, RSVP, and STUN

This document defines an initial set of IEs, a set of encoding rules, and initial usage model. The actual encoding is defined in [I-D.choukir-tsvwg-flow-metadata-encoding]. Additional documents define the mapping to specific signaling protocols (e.g. RSVP [I-D.zamfir-tsvwg-flow-metadata-rsvp], STUN [I-D.martinsen-mmusic-malice], and PCP [I-D.wing-pcp-flowdata])

2. Background

This section provides background on the motivation for the framework.

Identification and treatment of application flows are critical for the successful deployment and operation of applications based on a wide range of signaling protocols. Historically, this functionality has been accomplished to the extent possible using heuristics, which inspect and infer flow characteristics.

Heuristics may be based on port ranges, IP subnetting, or deep packet inspection (DPI), e.g. application level gateway (ALG). Port based solutions suffer from port overloading and inconsistent port usage. IP subnetting solutions are error prone and result in network management hassle. DPI is computationally expensive and becomes a challenge with the wider adoption of encrypted signaling and secured traffic. An additional drawback of DPI is that the resulting insights are not available, or need to be recomputed, at network nodes further down the application flow path.

The proposed solution allows applications to explicitly signal their flow characteristics to the network. It also provides network nodes with visibility of the application flow characteristics and enables them to contribute to the flow description. The resulting flow description may be communicated as feedback from the network to applications.

2.1. Deep packet inspection

2.1.1. Benefits

Deep Packet Inspection (DPI) and other traffic observation methods (such as performance monitoring) are successfully being used for two type of workflows:

1. Provide network operators with visibility into traffic for troubleshooting, capacity planning, accounting and billing and other off network workflows. This is done by exporting observed traffic analysis via protocol such as IPFIX and SNMP.

2. Provide differentiated network services for the traffic according to network operator defined rule sets, including policing and shaping of traffic, providing admission control, impacting routing, permitting passage of traffic (e.g. firewall functions), etc.

Note: For the context of this document, we consider that DPI starts as early into packets as using ACLs with UDP/TCP port numbers to classify traffic.

2.1.2. Limitation

These two workflows, visibility and differentiated network services, are critical in many networks. However, their reliance on inspection and observation limits the ability to enable these workflows more widely.

- Simple observation based classification, especially ones relying on TCP/UDP, ports often result in incorrect results due to port overloading (i.e. ports used by applications other than those claiming the port with IANA).

- More and more traffic is encrypted, rendering deep packet inspection impossible or much more complex (e.g. needing to share encryption keys with network equipment).

- Observation generally requires inspecting the control and signaling traffic of applications. This traffic may flow through a different network path than the actual application data traffic. Impacting the traffic behavior is ineffective in those scenarios.

- o Observation of control, signaling and data traffic with DPI will in general result in less insight into the applications intent than if the application was explicitly signaling its intent to the network.

- o Without explicit desire by the application to signal its intent to the network, it will also not consider to explicitly provide authentication to the network. DPI mechanism have a more difficult job in analyzing application traffic when authentication mechanisms are in use (if they even can)

- o Without explicit involvement of the application, network services leveraging DPI traffic classification impact the application behavior by impacting its traffic, but cannot provide explicit feedback to the application in the form of signaling.

2.2. Explicit signaling methods

There are a variety of existing and evolving signaling options that can provide explicit application to network signaling and serve the visibility and differentiated network services workflows where DPI is currently being used. It seems clear that there will be no single one-protocol-fits-all solution. Every protocol is currently defined in its own silo, creating duplicate or inconsistent information models. This results in duplicate work, more operational complexity and an inability to easily convert information between protocols to easily leverage the best protocol option for each specific use case. Examples of existing signaling options include the following:

- o RSVP is the original on path signaling protocol standardized by the IETF. It operates on path out-of-band and could support any transport protocol traffic (it currently supports TCP and UDP). Its original goal was to provide admission control. Arguably, its success was impacted by its reliance on router-alert because this often leads to RSVP packets being filtered by intervening networks. To date, more lightweight signaling workflows utilizing RSVP have not been standardized within the IETF.

- o NSIS (next Steps in Signaling) is the next iteration of RSVP-like signaling defined by the IETF. Because it focused on the same fundamental workflow as RSVP admission control as its main driver, and because it did not provide significant enough use-case benefits over RSVP, it has seen even less adoption than RSVP.

- o STUN is an on path, in-band signaling protocol that could easily be extended to provide signaling to on path network devices because it provides an easily inspected packet signature, at least for transport protocols such as UDP and SCTP. Through its extensions TURN and ICE, it is becoming quite popular in application signaling driven by the initial use-case of automatically opening up firewall pinholes and determining the best local and remote addresses for peer-to-peer connectivity (ICE).

- o PCP is a protocol designed to support use cases similar to UPnP firewall traversal. It also can easily be extended to provide more generic application to network signaling for traffic flows. Unlike the prior protocols, it is not meant to be used on path end-to-end but rather independently on one "edge" of a traffic flow. It is therefore an attractive alternative (albeit with challenges under path redundancy) because it allows the introduction of application to network signaling without relying on the remote peer. This is especially useful in multi-domain communications.

- o In addition to these, depending on the devices where it is performed, different degrees of DPI may be used to achieve explicit signaling. For example, inspection of HTTP connections is often viable in high-touch network devices. Such inspection may provide explicit signaling if the application purposely keeps or inserts information elements that are meant to be signaled to the network in the clear, or knowingly uses an encryption scheme shared with the network.

Rather than encourage independent, protocol specific solutions to this problem, this document provides a protocol and application independent framework that can be applied in a consistent fashion across the various protocols.

3. Proposed framework

3.1. Overview

The proposed framework includes the following elements:

3.1.1. Common, application independent, IPFIX registered, information elements

An application media flow may be expressed as a set of information elements that are defined and registered like observation-based IPFIX attributes. We propose leveraging IPFIX as the information model (not necessarily as the transport signaling) for the following reasons:

- o As outlined above, export of traffic information is one of the two big workflows. IPFIX is arguably the most flexible, extensible and best defined option for this. Leveraging the same information model for flow characteristics facilitates export of this information via IPFIX.

- IPFIX allows for IETF/IANA standardized information elements, but also for unambiguous vendor-defined attributes by including the so-called PEN (Private Enterprise Number) into the information element type. Note that IPFIX has ongoing work to better disseminate vendor specific registration of attributes. The framework defined here expects to be able to leverage the output of that work.

3.1.2. Cross-protocol information element encoding rules

The majority of the protocols listed previously (RSVP, NSIS, STUN/ ICE, PCP) require (or favor) compact binary encoding of information elements. This is natively supported by the information element registration of IPFIX.

The IPFIX registry defines each information element’s data-type, and there is a native binary network encoding for each of these types. At a minimum, every protocol leveraging common information elements would need to use an encoding that identifies the information element’s PEN and IE-ID, and that leverages network standard binary encoding of the value including the length of the value. Including the length of the value into the encoding is required for extensibility because otherwise new information elements could not be introduced without first having all network devices know the data-type, and therefore the length, of the information element. Leveraging network standard binary encoding is equally important to permit network elements to propagate information elements from one protocol to another protocol without understanding the information elements data-type.

In protocols that are not constrained to binary encoding, it is nevertheless highly desirable to include the equivalent information and therefore permit propagation between binary and non-binary transport of information elements without having to understand all information elements.

3.1.3. Anticipated Usage Models

The signaling of information elements may be from application to the network or from network to application. When signaled within a given protocol, the information elements may be interpreted independently of that protocol, or it may be used in combination with the given protocol.

3.1.3.1. Informational

The most simplistic usage model is one in which applications signal information elements describing their anticipated or existing flows into the network along the path of those flows without expecting or requiring anything back from the network. Network elements along the flow path may or may not do something with this information.

This "informational" usage model enables network elements along the path to support the workflows traditionally performed via DPI mechanisms, as described previously.

3.1.3.2. Advisory

This usage model extends the "informational" usage in that the application expects or requests some information back from the network. With this usage, the same information elements apply and may be communicated by the application into the network, but the application indicates its interest in receiving some feedback.

Default values are defined for each information element to unambiguously support cases in which an application does not have a valid value to communicate with the network; rather, it wants the network to provide a value back to it in response. In essence, this allows an application to ask a question and receive an answer from the network. Of course, a network element may provide similar feedback for cases in which an application communicated a non-default value as well. Network elements may also provide unsolicited advisory feedback.

In all cases, applications are not guaranteed to receive an answer or any specific service from the network. In the event an answer is provided, that answer is similarly not a guarantee of any specific service or treatment by the network. It is to be interpreted as advisory only.

As mentioned previously, the same information elements are used in the signaling from the application to the network as well as from the network to the application. The underlying transport protocol used to carry the information elements is expected to provide the necessary request/response semantics or some other mechanism by which the communication in both directions can be tied together.

3.1.3.3. Service Request

This usage model extends the "advisory" usage to operate as an explicit service request. Unlike the advisory usage, information elements signalled by the application are interpreted by network elements within the context of a service request, and information elements signalled by the network back to the application are interpreted within the context of a response to that request.

As with the advisory usage, the same information elements are used in the signaling from the application to the network as well as from the network to the application. The underlying transport protocol used to carry the information elements is expected to provide the necessary service request/response semantics.

3.1.4. Considerations for signaling of common information elements

3.1.4.1. Proxy originated information

The goal of this framework is to enable applications to explicitly signal common information elements about their traffic flows and optionally receive common information elements from the network as feedback. Nevertheless, it is clear that broad adoption of such technology is improved by enabling the use of proxies. The proxies can provide or amend the flow description information in the absence of Flow Metadata support by the application itself.

3.1.4.2. Authentication

Common information elements should provide for cryptographic authentication by the sender. In general the authentication provides some form of identification of the sender and proves that the common information elements covered by the authentication were originated from, or approved by, that identity.

3.1.4.3. Common encoding

A companion document [I-D.choukir-tsvwg-flow-metadata-encoding] covers recommended encoding rules that take the following aspects into account:

- Compact binary encoding rules

- Signaling for both sent and received traffic flows

- Signaling of standard and vendor specific information elements

- Minimizes protocol specific definition required to add informational or advisory common information elements into existing transactions

- Signaling of feedback from the network

- Identification of originator to support proxies and facilitate mitigation between common information elements from different originators

- Signaling of authenticators

3.1.4.4. Usage Model to Protocol integration

There is a range of options for how this framework is integrated with a particular transport protocol. We describe two examples we consider useful:

3.1.4.4.1. Common transport informative integration

1. A transport protocol signaling method is defined to carry the common encoded information elements at least in signaling from application to network.

2. If the transport by itself does not already have a mechanism to indicate a purely informative protocol transaction, then a protocol specific indication for this is added.

In result, this integration achieves two option:

1. Informative common information elements can be sent from application to network by using the protocol’s method to indicate the purely informational protocol transaction. This option effectively leverages the protocol as transport for additional informative attribute based services without impacting the services and transactions of the protocol otherwise.

2. Informative common information elements can be sent alongside an existing protocol transaction. In this case they may either be ships in the night (triggering informative attribute based services), or they may additionally be used by the policy rules of the protocol transaction itself which could be advisory or service request. All feedback of the transaction would still rely on protocol specific information element (common information elements only used from host to network).

This integration is for example defined in [I-D.wing-pcp-flowdata], [I-D.zamfir-tsvwg-flow-metadata-rsvp], and [I-D.martinsen-mmusic-malice].

3.1.4.4.2. Common transport advisory integration

In addition to the common transport informative integration, the transport encoding is extended to carry the common transport information element in feedback messages from the network to the host /application. The method to indicate informative only transaction, when sending to the network is used to indicate advisory only transaction when signaling from the network.

This option primarily enables informative and advisory usage models, but it can equally interact with pre-existing service-request options of the transport protocol and impact advisory feedback or the service request itself based on that interaction.

3.2. Proposed common information elements

The section defines an initial set of common information elements. These information elements are intended to be added to the set of IANA standardized information elements either by this or associated documents. Additional documents are expected to define additional attributes that can use either IANA or other vendor-PEN.

All information element definition must include the following:

1. Default value to be provided by an application when it does not have an informative value to provide to the network, but is interested in receiving an advisory value of the attribute from the network. If no advisory feedback is requested, and no informative value is known, the attribute may simply not be sent.

2. Conflict resolution in the presence of different values for the same information element (e.g. two peers signaling information elements for both the upstream and downstream direction of a flow include different values for the information element)

3.2.1. Bandwidth Attributes

3.2.1.1. Maximum Bandwidth

This attribute is used to convey the maximum sustained bandwidth for the flow. It is a 64 bit value and is specified in bits per second.

Default Value: 0

Conflict Resolution: Minimum for the set of non-default values

3.2.1.2. Minimum Bandwidth

This attribute is used to convey the minimum sustained bandwidth for the flow. It is a 64 bit value and is specified in bits per second. Not sending the Minimum Bandwidth is equivalent to sending the same value as for Maximum Bandwidth.

Default Value: 0

Conflict Resolution: Minimum of the set of non-default values

3.2.1.3. Bandwidth Pool

This attribute is used to convey that the traffic dynamically shares bandwidth with other traffic using the same Bandwidth Pool. Variable length GUID (Global Unique ID) of at least 48 bits. The Maximum Bandwidth used by the pool is the largest Maximum Bandwidth indicated by any member, the Minimum Bandwidth of the Pool is the largest Minimum Bandwidth indicated by any member.

3.2.2. Traffic Class Attributes

3.2.2.1. RFC4594-DSCP

This attribute is used to convey the DSCP value appropriate for the flow. It is an 8 bit value. Values signaled are assumed to be in compliance with [RFC4594] or backward compatible extensions thereof. Other values are undefined.

Default Value: 0xff

Conflict Resolution: tbd

3.2.2.2. Traffic Class Label (TCL)

The data type of this information element is a string. It carries the Traffic Class Label defined in [I-D.ietf-mmusic-traffic-class-for-sdp]. Depending on the outcome of that drafts standardization, the version carried as an information element may be slightly expanded over the its definition for SDP. The TCL is a structured string of the form:

```
<category>.<application>(.adjective)(.adjective)
```

category and application provide a base categorization of the traffic class that attempts to provide a simplified and extensible, framework for the traffic class definitions in [RFC4594]. These base classifications can be refined with zero or more adjectives. Examples of a TCL is "conversational.video.avconf".

Default Value: Empty string

Conflict Resolution: tbd

3.2.3. Acceptable Path Attributes

3.2.3.1. Delay Tolerance

This attribute is used to convey the delay tolerance of an application with respect to the associated flow. When provided by a network element, it indicates the delay tolerance expected of the application with respect to the associated flow. It is a 16 bit field defined in terms of milliseconds.

Default Value: 0

Conflict Resolution: For application to network, the minimum of the set of non-default values. For network to application, the maximum of the set of non-default values.

3.2.3.2. Loss Tolerance

This attribute is used to convey the loss tolerance of an application with respect to the associated flow. When provided by a network element, it indicates the loss tolerance expected of the application with respect to the associated flow. It is a 16 bit field defined in terms of hundredths of a percent of dropped packets (e.g. 5 == 0.05%, 150 == 1.50%, etc.)

Default Value: 0

Conflict Resolution: For application to network, the minimum of the set of non-default values. For network to application, the maximum of the set of non-default values.

3.2.4. Application Identification

Application identification is clearly one of the more difficult classification goals. The proposals included here are as of yet not widely vetted:

3.2.4.1. RFC 6759 style application identification

[RFC6759] defines the IPFIX IE-IDs that permit both IANA and vendor specific application identification. Though defined for observation (a.k.a.: DPI), it could also be used with explicit signaling from applications.

Applications that use one of the protocols for which there is an IANA port allocation could explicitly indicate this port via the IANA-L4 engine-id in their application to network signaling. This would identify the application even if the application is not using the IANA assigned port for it. This covers cases in which applications use ports other than registered, such as HTTP servers running on other than 80, or when ports get mapped due to PAT.

To avoid collision with DPI exported IANA-L4 classification, it is necessary to assign a new engine-id for application-self assigned IANA-L4 classification (e.g. new engine-id for IANA-L4-SELF-ASSIGNED). If an application vendor has a PEN, the application can use a PANA-L7-PEN classification with the PEN of the originating application vendor. Likewise, if applications are in general made available via "market" type reseller mechanism (common in mobile device applications), then the application vendor could request an application identification from the market owner and leverage the market owners PEN.

3.2.4.2. URL style application identification

One problem with [RFC6759] style application identification especially non-IANA registered ones is the complexity in making all network elements learn the semantic of the numeric encoding of e.g. the PANA-L7-PEN information element in signaling protocols that only use the numeric encoding of information elements. The second problem may be to determine what PEN to use, because not every developer of an application may be a company that has a PEN or otherwise would intend to apply for one. Application identification via a URL encoded string information element is a way to overcome both issues. Today, almost all applications have some DNS domain associated with them through which they are being marketed or that belongs to the company developing the application. Therefore, one simple form of self assigned application identification is a new IPFIX information element: UrlAppId. The value of this information element is an abbreviated URL of the following form:

```
<fqdn> / <app-name> /[ <version> | <other-details> ]
```

The idea is that the owner of <fqdn> (fully qualified domain name) is assigning an <app-name>, and by signaling both <domain-name> and <app-name>, this information element provides a self-identifying, unambiguous application identification.

Example:

example.com/network-lemmings/sdn-edition

A game publishing house or application market operator with the domain name example.com is initially allocating the UrlAppId example.com/network-lemmings to that application. After 35 years, a new variant of the game is released, the SDN edition, and the app-developer decides that it would best like to distinguish this application variant by the above UrlAppId example.com/network-lemmings/sdn-edition.

In general, different traffic flows within a single application should best not be distinguished via the UrlAppId, but instead rely on attributes more specifically targeted for that purpose (such as the TrafficClassLabel). If there is no adequate better attribute defined, application developers may choose to use the other-details section of the UrlAppId to distinguish flows within the same application.

Formally, the only requirement against the UrlAppId is that the fqdn part is a DNS domain owned by the assigner, and that the rest of the string after the first / is as self explanatory as possible.

It should be noted that in the context of DPI, classification of web-based application traffic is very often performed by URL inspection of HTTP traffic. This proposed intent based information element leverages that model and makes it usable where it can not be currently used with just DPI: encrypted HTTP, non-HTTP applications, HTTP applications with non-descriptive URLs, etc.

4. Acknowledgements

The authors would like to thank Dan Wing, Anca Zamfir, Paul Jones, and Tirumaleswar Reddy for their valuable contributions to this document.

5. Informative References

[I-D.choukir-tsvwg-flow-metadata-encoding]
Eckert, T., Zamfir, A., Choukir, A., and C. Eckel, "Protocol Independent Encoding for Signaling Flow Characteristics", draft-choukir-tsvwg-flow-metadata-encoding-00 (work in progress), July 2013.

[I-D.ietf-mmusic-traffic-class-for-sdp]
Polk, J., Dhesikan, S., and P. Jones, "The Session Description Protocol (SDP) ’trafficclass’ Attribute",

draft-ietf-mmusic-traffic-class-for-sdp-04 (work in progress), July 2013.

[I-D.martinsen-mmusic-malice]
Penno, R., Martinsen, P., Wing, D., and A. Zamfir, "Meta-data Attribute signaLling with ICE", draft-martinsen-mmusic-malice-00 (work in progress), July 2013.

[I-D.wing-pcp-flowdata]
Wing, D., Penno, R., and T. Reddy, "PCP Flowdata Option", draft-wing-pcp-flowdata-00 (work in progress), July 2013.

[I-D.zamfir-tsvwg-flow-metadata-rsvp]
Eckert, T., Zamfir, A., and A. Choukir, "Flow Metadata Signaling with RSVP", draft-zamfir-tsvwg-flow-metadata-rsvp-00 (work in progress), July 2013.

[RFC4594] Babiarz, J., Chan, K., and F. Baker, "Configuration Guidelines for DiffServ Service Classes", RFC 4594, August 2006.

[RFC6759] Claise, B., Aitken, P., and N. Ben-Dvora, "Cisco Systems Export of Application Information in IP Flow Information Export (IPFIX)", RFC 6759, November 2012.

Authors’ Addresses

Toerless Eckert (editor)
Cisco Systems, Inc.
San Jose
US

Email: eckert@cisco.com

Reinaldo Penno
Cisco Systems, Inc.
170 West Tasman Drive
San Jose 95134
USA

Email: repenno@cisco.com

Amine Choukir
Cisco Systems, Inc.
Lausanne
CH

Email: amchouki@cisco.com

Charles Eckel
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134
US

Email: eckelcu@cisco.com