In-situ OAM (IOAM)

IPPM

July 18th 2018

Change set

- draft-ietf-ippm-ioam-data – Security consideration, Layering
- Encapsulation:
  - VXLAN GPE: draft-brockners-ippm-ioam-vxlan-gpe
  - GENEVE: draft-brockners-ippm-ioam-geneve
  - GRE: draft-weis-ippm-ioam-gre
  - IPv6: draft-ioametal-ippm-6man-ioam-ipv6-options (New)
- Export
  - draft-spiegel-ippm-ioam-rawexport
- FYI: SFC wg adoption
  - NSH encapsulation - draft-ietf-sfc-ioam-nsh
  - Proof of transit - draft-ietf-sfc-proof-of-transit

Changes to draft-ietf-ippm-ioam-data

1. Layering:
   - If several encapsulation protocols (e.g., in case of tunneling) are stacked on top of each other, IOAM data-records could be present at every layer. The behavior follows the ships-in-the- night model**, i.e. IOAM data in one layer is independent from IOAM data in another layer. Layering allows operators to instrument the protocol layer they want to measure. The different layers could, but do not have to share the same IOAM encapsulation and decapsulation.**
   - Encapsulation drafts include RFC7605 based considerations:
     Ensure that transit devices don't change IOAM data in environments where UDP port numbers aren't controlled.
2. Security Considerations
   - Details on next slide

Security Considerations

- Successful attack on IOAM may allow:
  - Compromise detection of failures / anomalies:
    - False positive
    - False negative
  - Reconnaissance
  - DoS:
    - Resource consumption
    - Packet length
- Management plane attacks
- Proof of transit (PoT) can be used for path verification

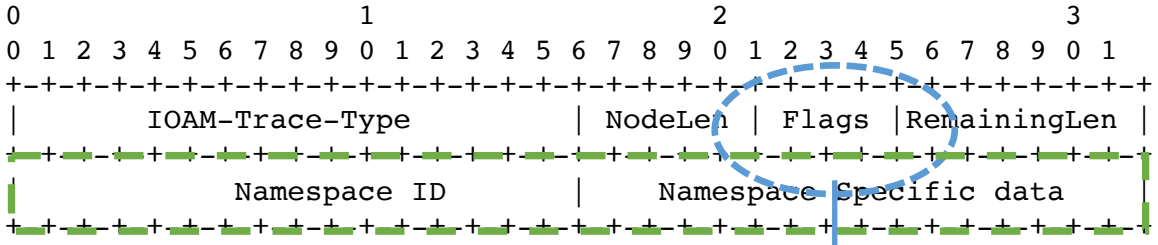
For Discussion:
On the need for a Namespace Identifier

- “IOAM is a network domain focused feature, with "network domain" being a set of network devices or entities within a single administration.” (see draft-ietf-ippm-ioam-data-03, section 3)
- IOAM data fields are defined in their own namespace. Data fields can have a domain specific interpretation, e.g. node id, interface id, queue depth (measured in memory buffers), app-data, opaque state snapshot.
- A deployment could require several different interpretations of the data fields (i.e. namespaces), consider e.g. layered IOAM deployment
- IOAM currently does not offer to carry an identifier for the namespace (to identify the interpretation context/domain) in the packet.

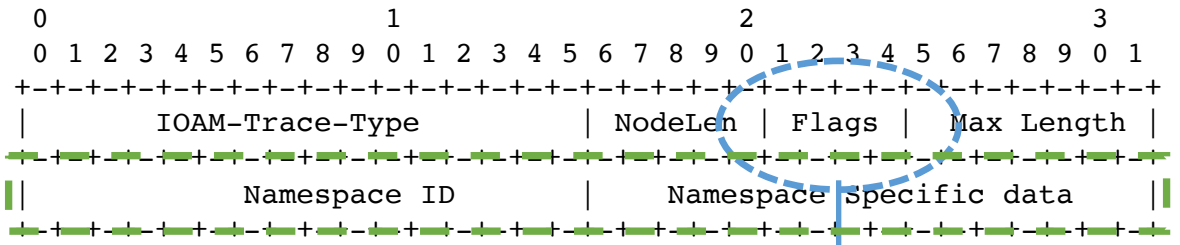
Namespace-ID proposal

- Extend base header by 4-bytes, adding namespace specific information
- Flag (Namespace bit) to identify whether the namespace ID is populated

Pre-Allocated trace option header:



Incremental trace option header:



Flags 4-bit field. New flag bit to extend the header for namespace specific data.

1. **Bit 0** "Overflow (O-bit) (most significant bit).
2. **Bit 1** "Loopback" (L-bit).
3. ***Bit 2*** *Namespace-ID (N-bit). Namespace-ID bit is used to indicate that the 4-octet of a namespace identifier and data is populated.*
4. **Bit 3** Reserved: Must be zero

Leveraging Namespace-ID for added flexibility

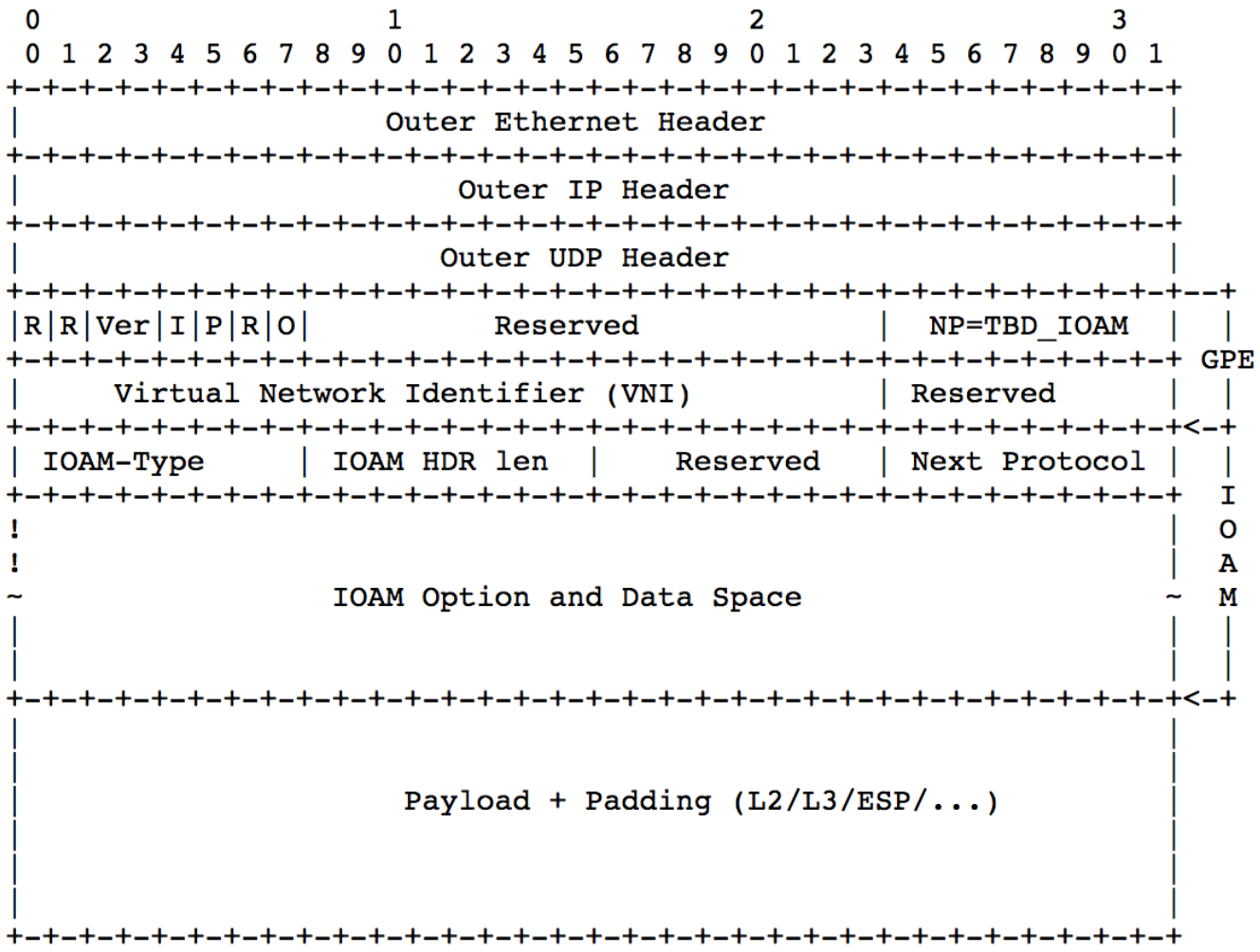
- IOAM trace options fields in draft-ietf-ippm-ioam-data represent the main set of industry use-cases, but not all
- Several cases have been made for specific enhancements, though they haven't found wider consensus so far. Examples:
  - draft-kumar-ifa-00: Indicate per-node which data fields are filled/supported (which let IFA to define new fields rather than reuse IOAM data fields)
  - draft-song-ippm-ioam-scalability-00: Additional trace types for specific use-cases (e.g. record inter-packet gap, etc.)
- Current approach to domain specific data fields and data interpretation is to use either "app-data" (short/wide) or "opaque state snapshot"
  - App-data is fixed free format, syntax/semantic is undefined
  - Opaque state snapshot is length/schema-id free format, syntax/semantic is undefined
- Namespace concept allows to specify how fields like app-data, opaque-state snapshot are interpreted: Examples:
  - For draft-kumar-ifa-00: Interpret app-data as a bit-field to express which fields a node updated
  - For draft-song-ippm-ioam-scalability-00: Interpret app-data / opaque-state snapshot as per the newly identified use-cases

Consider renaming "app-data" field to "namespace-specific-data".

IOAM Encapsulation

IOAM in VXLAN-GPE

draft-brockners-ippm-ioam-vxlan-gpe

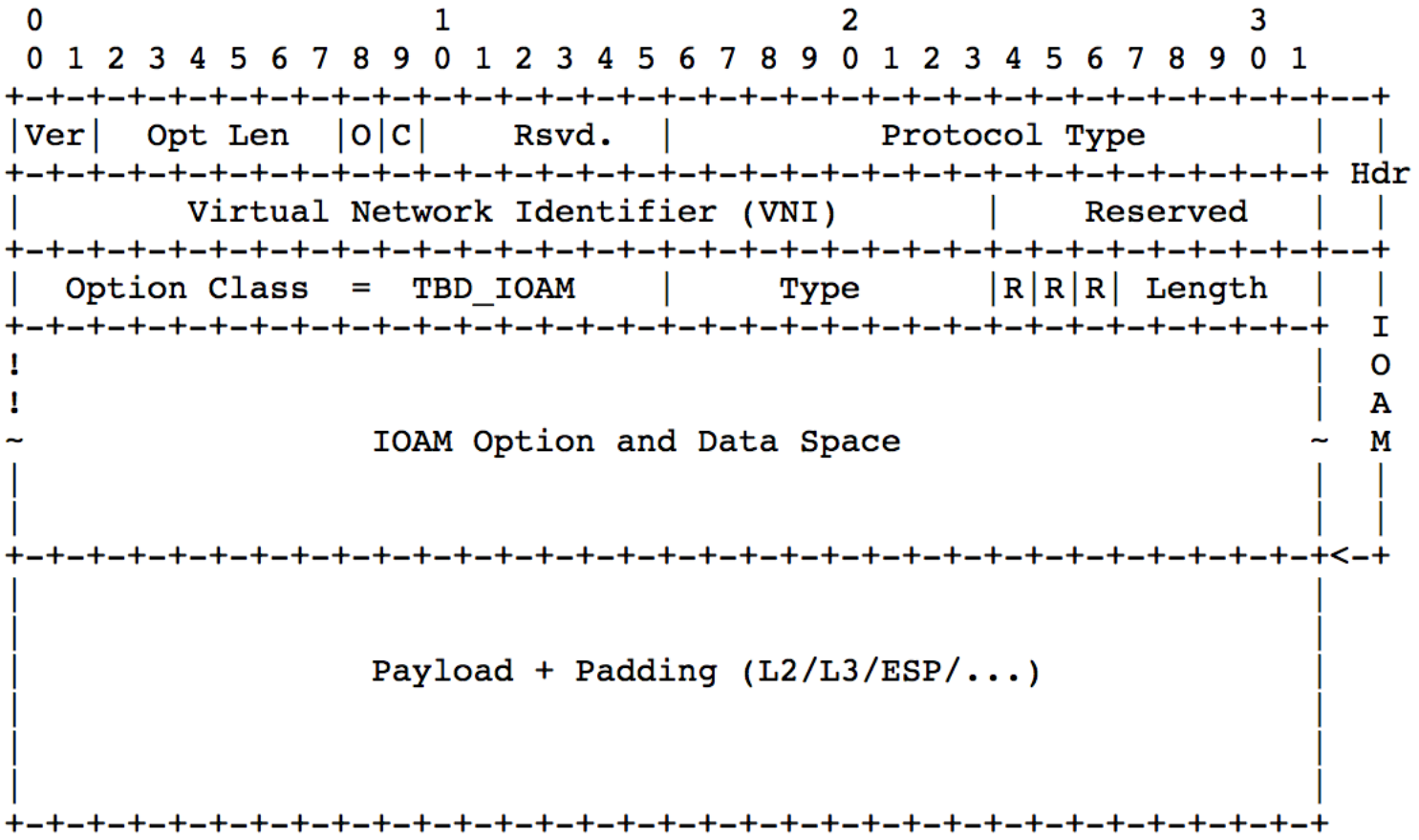


→ VXLAN GPE header

→ IOAM options

IOAM in Geneve

draft-brockners-ippm-ioam-geneve



→ Geneve header

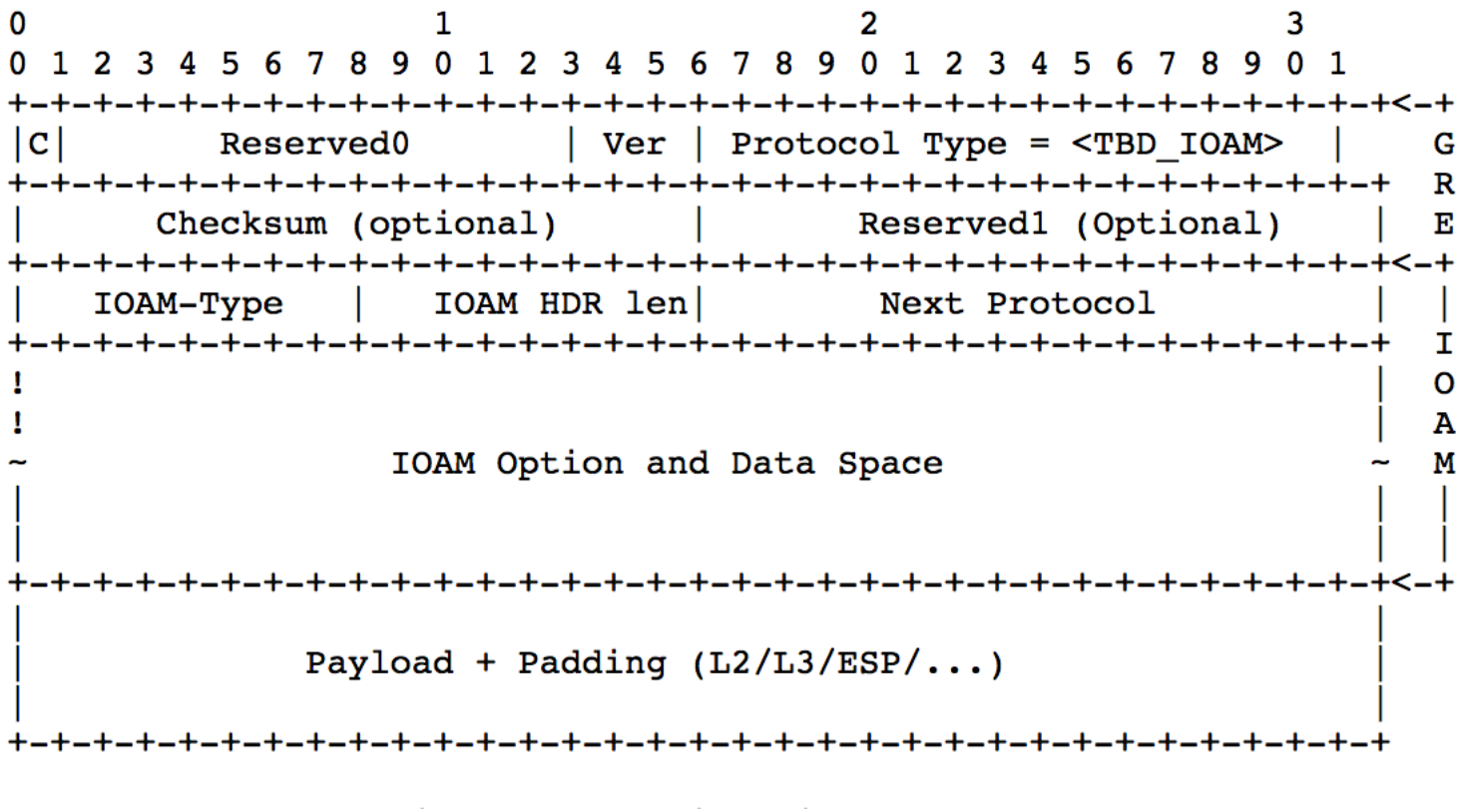
→ Geneve option class = IOAM,

Type as defined in:

draft-ietf-ippm-ioam-data

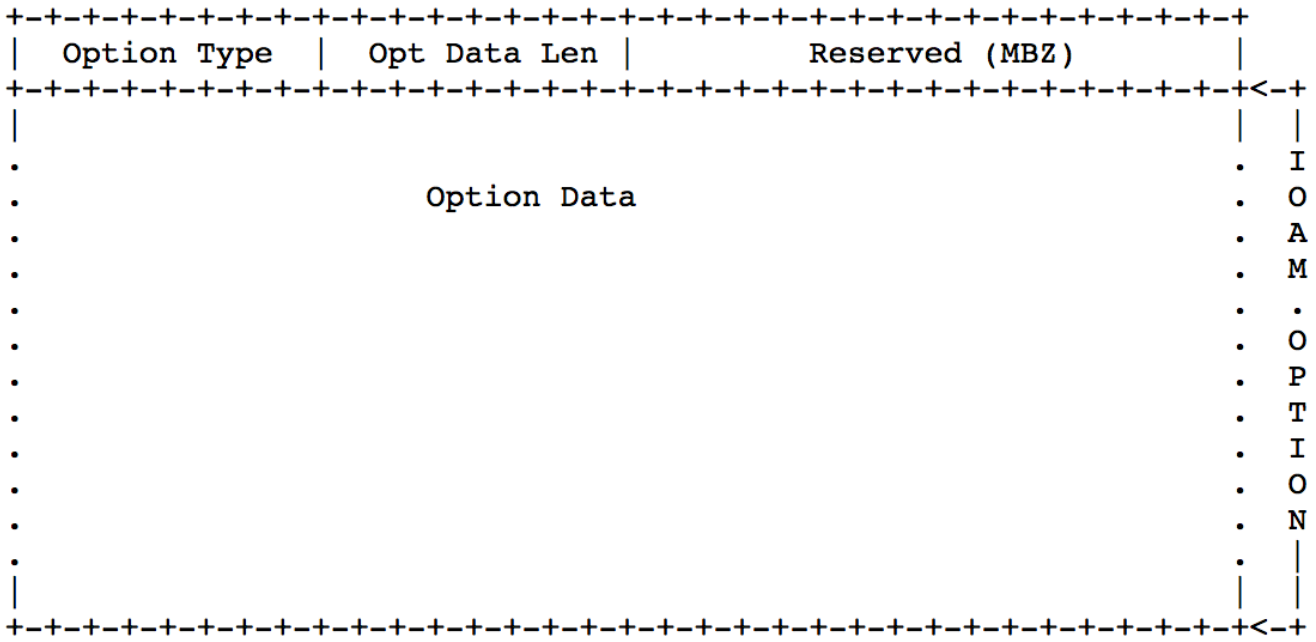
IOAM in GRE

[draft-weis-ippm-ioam-gre]



IOAM in IPv6

draft-ioametal-ippm-6man-ioam-ipv6-options



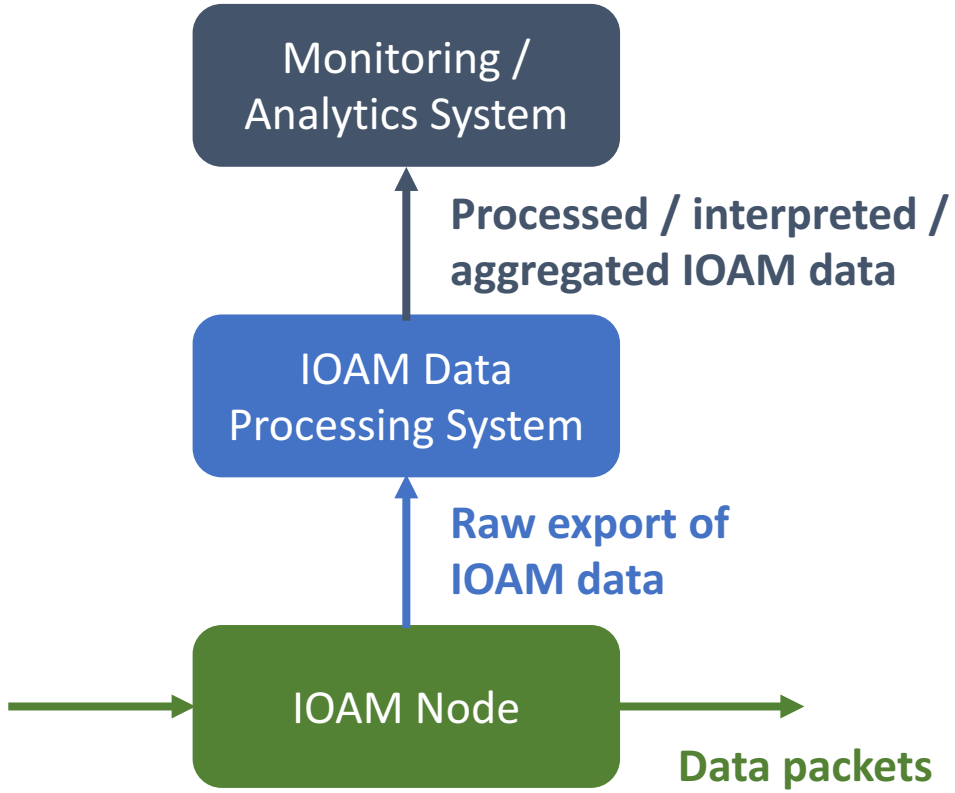
- IPv6 Hop-by-Hop options
  - Pre-allocated Tracing Option
  - Incremental Tracing Option
  - Proof of Transit Option
- IPv6 Destination Option
  - Edge to Edge Option

IOAM Raw Data Export with IPFIX

draft-spiegel-ippm-ioam-rawexport-00

Raw Export of IOAM Data

- A mode of operation where a node exports the IOAM data as it is received in the packet
- The exporting node neither interprets, aggregates nor reformats the IOAM data before it is exported
  - Off-load from the node which performs data plane operations
  - Implementable at billions of packets per second



Using IPFIX for Raw Export of IOAM

- IPFIX is defined as a generic export protocol that can export any Information Element(s) as long as they are described in the information model
- IPFIX protocol is well suited for and is defined as the protocol for exporting packet samples in [RFC5476]
  - IPFIX/PSAMP already define many of the information elements needed for exporting raw sections of packets, needed for deriving context
- Many network devices may be quite constrained in the IPFIX formats that they can support for IOAM export at high speeds
  - Small number of information elements supported
  - Alignment along 4 octet boundaries
    - Alignment constraint varies depending on the encoding used for packet snippets:
      - Fixed length : 4n
      - Variable length < 255 : 4n + 3
      - Variable length < 65536 : 4n + 1
  - Possibly a limited number of fixed templates

Key IPFIX IEs Leveraged for IOAM Raw Data Export

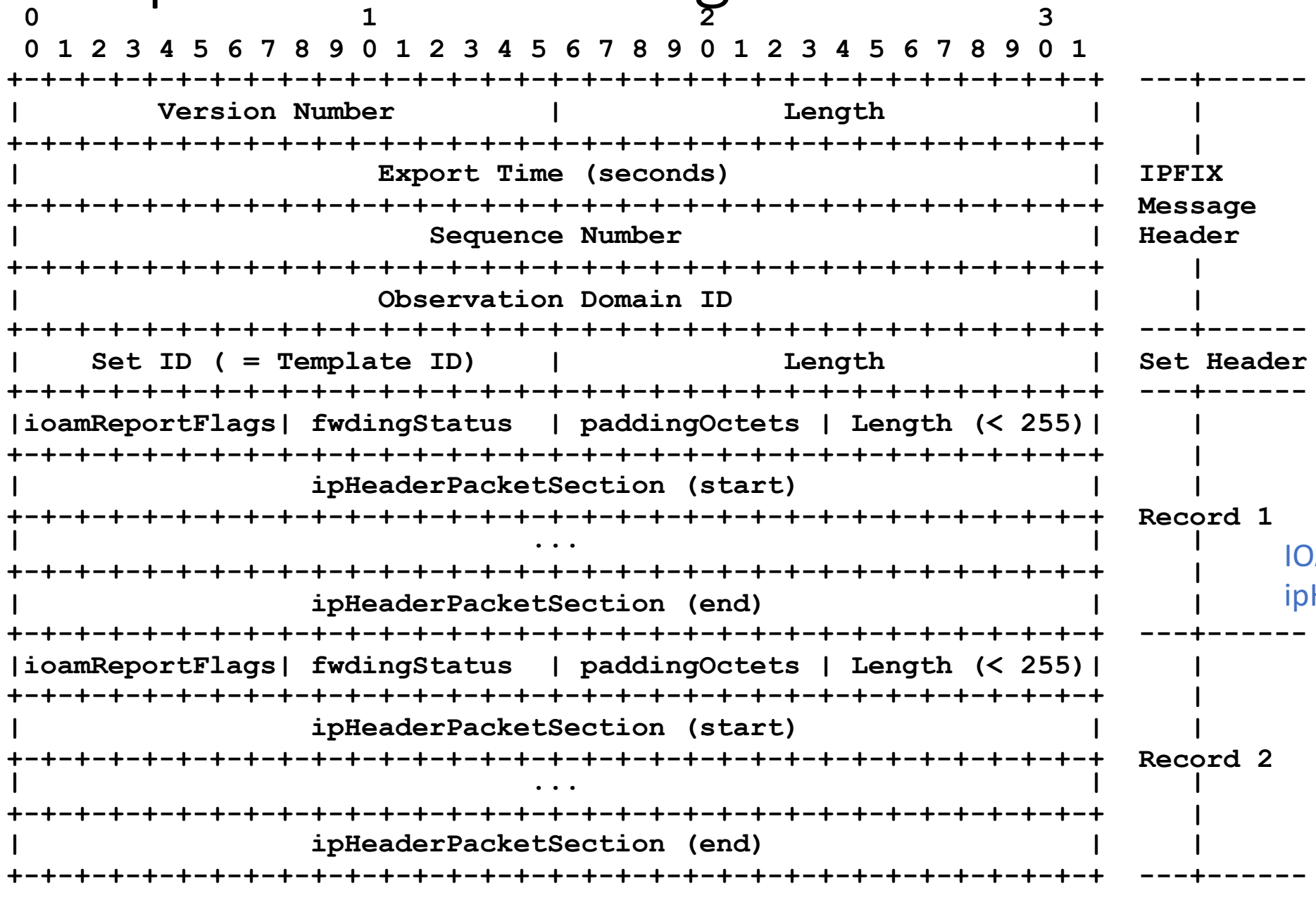
313	ipHeaderPacketSection	octetArray	default	This Information Element carries a series of n octets from the IP header of a sampled packet … With sufficient length, this element also reports octets from the IP payload. …
315	dataLinkFrameSection	octetArray	default	This Information Element carries n octets from the data link frame of a selected frame … Further Information Elements, i.e., dataLinkFrameType and dataLinkFrameSize, are needed to specify the data link type and the size of the data link frame of this Information Element. …
408	dataLinkFrameType	unsigned16	flags	This Information Element specifies the type of the selected data link frame. The following data link types are defined here: - 0x01 IEEE802.3 ETHERNET [IEEE802.3] - 0x02 IEEE802.11 MAC Frame format [IEEE802.11]
410	sectionExportedOctets	unsigned16	quantity	This Information Element specifies the observed length of the packet section … when padding is used. …
89	forwardingStatus	unsigned8	identifier	This Information Element describes the forwarding status of the flow and any attached reasons. …

Proposed new IPFIX Information Elements

ioamReportFlags	unsigned8	flags	This IE describes properties associated with an IOAM report. bit description ---+--- 0 Dropped Assn – Dropped packet of interest 1 Congested Queue Association – Indicates the presence of congestion on a monitored queue 2 Tracked Flow Association – Matched a flow of interest
ioamEncapsulationType	unsigned8	identifier	0 None: [I-D.ietf-ippm-ioam-data] 1 GRE: [I-D.weis-ippm-ioam-gre] 2 VXLAN-GPE: [I-D.brockners-ippm-ioam-vxlan-gpe] 3 GENEVE: [I-D.brockners-ippm-ioam-geneve] 4 NSH: [I-D.brockners-sfc-ioam-nsh]
ioamPreallocatedTraceData	octetArray	default	This IE carries n octets of IOAM Preallocated Trace data. *
ioamIncrementalTraceData	octetArray	default	This IE carries n octets of IOAM Incremental Trace data. *
ioamE2EData	octetArray	default	This IE carries n octets of IOAM E2E data. *
ioamPOTData	octetArray	default	This IE carries n octets of IOAM POT data. *

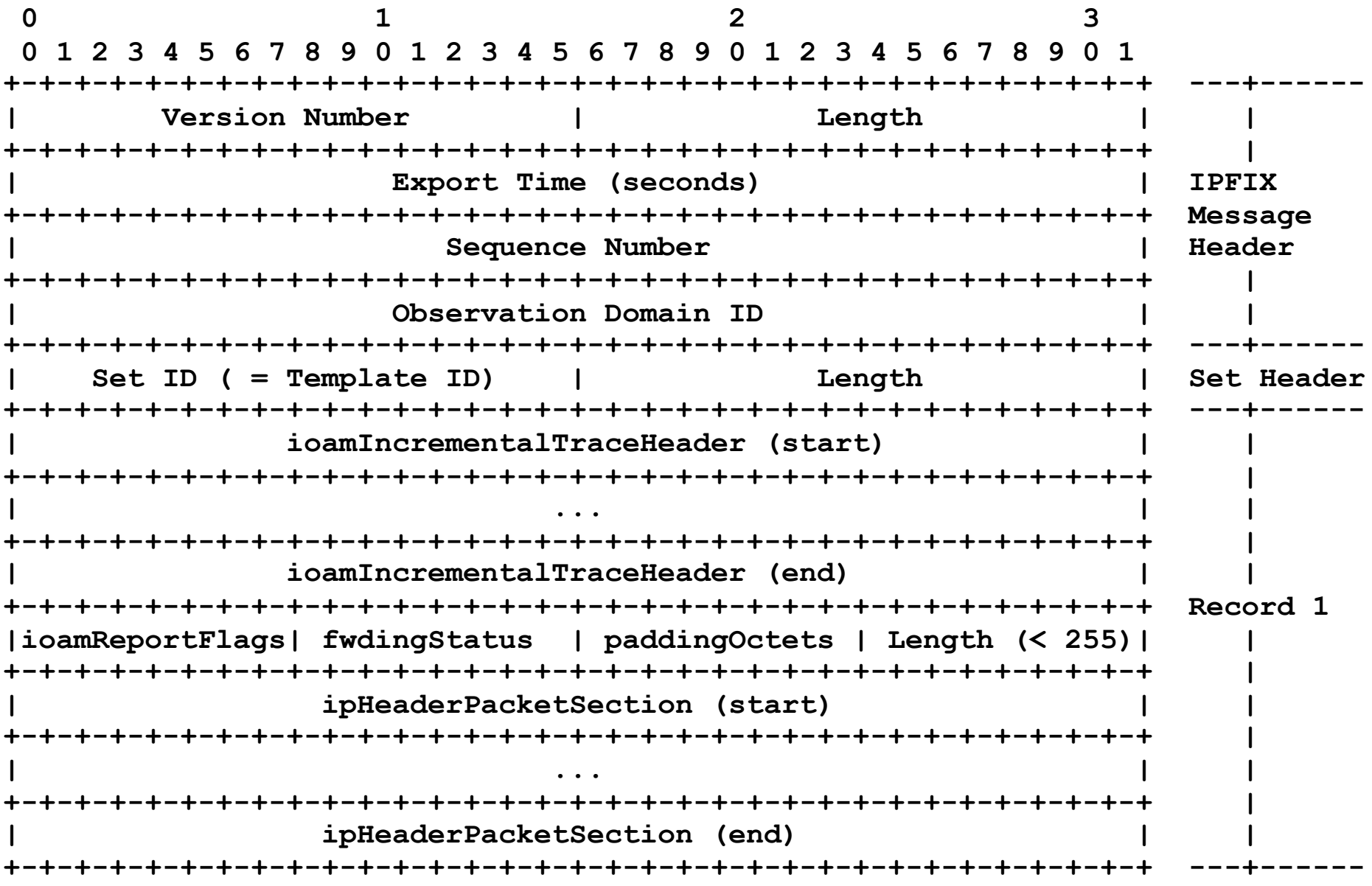
* When ioamEncapsulationType is present and has a value other than "None", and with sufficient length, this element may also report octets from subsequent headers and payload

Example: Variable Length < 255 IP Packet



IOAM data is part of ipHeaderPacketSection

Example: Variable Length IP Packet w/ Outer Incremental Trace



Potential Updates to Existing IPFIX IEs?

- The *"descriptions"* of some existing IPFIX information elements impose requirements
- There are opportunities to improve efficiency by relaxing these requirements
- *313 ipHeaderPacketSection* and *315 dataLinkFrameSection*
  - "*When the sectionExportedOctets field corresponding to this Information Element does not exist, this Information Element SHOULD have a variable length and MUST NOT be padded*"
  - When *ipHeaderPacketSection* is fixed size, why can't one omit *sectionExportedOctets* and still use padding?
    - Use total length / payload length in the IP header to determine where padding begins
  - In order to align multiple records within a set, when:
    - the length of the IP packet is small and is not a multiple of 4 octets
    - there is a variable length instance of *ipHeaderPacketSection* without *sectionExportedOctets*

    Could the *Length* be set to a multiple of 4 octets, inserting 0 to 3 octets of padding?
- *315 dataLinkFrameSection*
  - "*Further Information Elements, i.e., dataLinkFrameType and dataLinkFrameSize, are needed to specify the data link type and the size of the data link frame of this Information Element*"
  - Why can't there be a default value of *Ethernet II* when *dataLinkFrameSection* is present but *dataLinkFrameType* is omitted?
  - Why does *dataLinkFrameSize* need to be present?

Next Steps

Next Steps

- draft-ietf-ippm-ioam-data – Has been stable. WGLC?
- Encapsulation drafts adoption in IPPM:
  - VXLAN GPE: draft-brockners-ippm-ioam-vxlan-gpe
  - GENEVE: draft-brockners-ippm-ioam-geneve
  - GRE: draft-weis-ippm-ioam-gre
- Export draft review
- IPv6 draft review

Thank you

Backup Slides

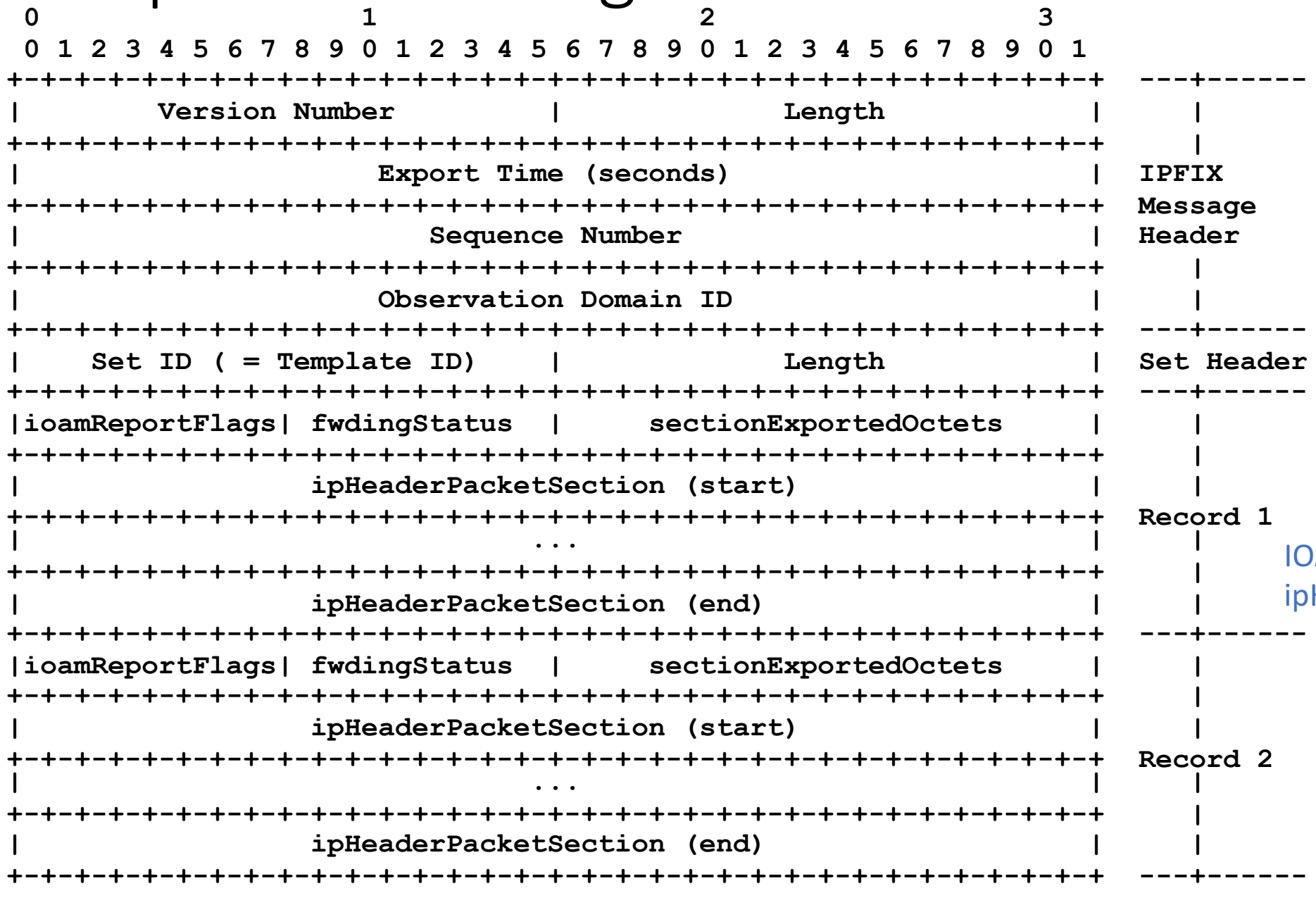
Abstract and Requirements

- Abstract:
  - This document discusses how In-situ Operations, Administration, and Maintenance (IOAM) information can be exported in raw, i.e. uninterpreted, format from network devices to systems, such as monitoring or analytics systems using IPFIX
- Requirements
  - Export all IOAM information contained in a packet
  - Export a specific IOAM data type - Incremental Trace type, Preallocated Trace type, Proof of Transit type, Edge to Edge type
  - Support coalescing of the IOAM data from multiple packets into a single raw export packet.
  - Support export of additional parts of the packet, other than the IOAM data as part of the raw export. This could be parts of the packet header and/or parts of the packet payload. This additional information provides context to the IOAM data (e.g. to be used for flow identification) and is to enable the IOAM data processing system to perform further analysis on the received data.
  - Report the reason why IOAM data was exported. The "reason for export" is to complement the IOAM data retrieved from the packet. For example, if a packet was dropped by a node due to congestion, it could be helpful to export the IOAM data of this dropped packet along with an indication that the packet that the IOAM data belongs to was dropped due to congestion.

Scope

- This document discusses raw export of IOAM data using IPFIX
- Considered out of scope for this document:
  - Protocols other than IPFIX for raw export of IOAM data
  - Interpretation or aggregation of IOAM data prior to exporting
  - Configuration of network devices so that they can determine when to generate IOAM reports, and what information to include in those reports
  - Events that trigger generation of IOAM reports
  - Selection of particular destinations within distributed telemetry monitoring sy
  - Export format for flow statistics or processed/interpreted/aggregated IOAM datastems, to which IOAM reports will be sent

Example: Fixed Length IP Packet



IOAM data is part of ipHeaderPacketSection

Example: Variable Length < 255 Ethernet Packet