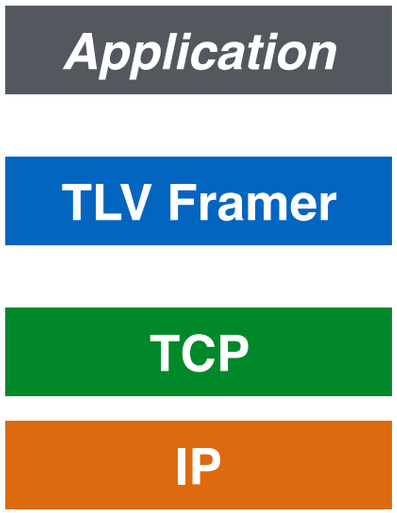
Adding Framers To TAPS

Tommy Pauly
TAPS
IETF 104, March 2019, Prague

Existing Text on Framers

Architecture *(draft-ietf-taps-arch-03)*

Framer: A Framer is a **data translation layer** that can be added to a Connection to define how **application-level Messages** are transmitted over a transport protocol. This is particularly relevant for protocols that otherwise present unstructured streams, such as TCP.



Existing Text on Framers

API *(draft-ietf-taps-interface-03)*

7.7. Sender-side Framing

```
Preconnection.FrameWith(Framer)

OctetArray := Framer.Frame(messageData)
```

8.4. Receiver-side De-framing over Stream Protocols

```
Preconnection.DeframeWith(Deframer)

{messageData} := Deframer.Deframe(OctetStream)
```

Defining Framers

Properties

A **`Framer`** can be inserted as an element of a **`Protocol Stack`**

Any protocol that operates on a single flow of data to package application Messages with a direct transformation can be expressed as a **`Framer`**.

Defining Framers

Properties

Simplest form of protocol, not a full transport

	Framer Protocol	Transport Protocol
Encapsulation	✓	✓
Prologue / Handshake	✓	✓
Data Encoding	✓	✓
Flow Control	✗	✓
Congestion Control	✗	✓
Reliability	✗	✓
Multiplexing	✗	✓
Multipath	✗	✓

Defining Framers

Examples

Framer Protocols	Transport Protocols
• TLVs (e.g. RADIUS) • Delimeters (e.g. SIP) • WebSocket • TLS records • ESP • HTTP messages	• TCP • SCTP • QUIC • HTTP/2 Connections

Proposed Changes

Modifications to Framer API

Remove the ability to use framers and de-framers asymmetrically

Define a Framer object that implements both inbound and outbound processing

Allow over any transport protocol, not just streams

Allow composition of multiple framers, and custom framers below built-in protocols

Implementing Framers

Functions a framer implements

```
@required
 handleOutput(Message)


 handleInput()


@optional
 start() -> bool(ready)
 stop()
 wakeup()
```

Functions a framer calls

```
parseOutput(range)->data
writeOutput(data)

parseInput(range)->data
deliverInput(Message)


ready()
fail(error)
scheduleWakeup(time)
```

Implementing Framers

TLV Example

```
func handleOutput(Message) {
  // Build a header
  outputData = concat(header, Message.data)
  writeOutput(outputData)
}
```

```
func handleInput() {
  headerData = parseInput(1..sizeof(Header))
  // Parse length in header
  advanceInput(sizeof(Header))
  bodyData = parseInput(1..length)
  deliverInput(Message(bodyData))
}
```

Annotating Metadata

Framer-specific Messages

Framers may need to communicate protocol-specific data to the application

- Headers for HTTP messages
- Op-Codes and Close-Codes for WebSocket
- Type for basic TLVs

Advanced Framers

Protocol features like STARTTLS can run below TLS and perform a handshake

Control messages, like PING keepalives, can be implemented on timers in a **`Framer`**

Messages that come in fragments (for protocols like WebSocket) can be delivered by the **`Framer`** as partial Messages

Next Steps

Get consensus on a framer model

Create PRs for API & Implementation

Anything else?