Post Sockets

An Abstract Programming Interface for the Transport Layer

draft-trammell-taps-post-sockets-00

Brian Trammell, Colin Perkins,
Tommy Pauly, Mirja Kühlewind

TAPS
IETF 98, March 2017, Chicago

Applications deal in objects of arbitrary size

Networks of the future are **explicitly multi-path**

Transports must **guarantee security properties**

Message reception is **inherently asynchronous**

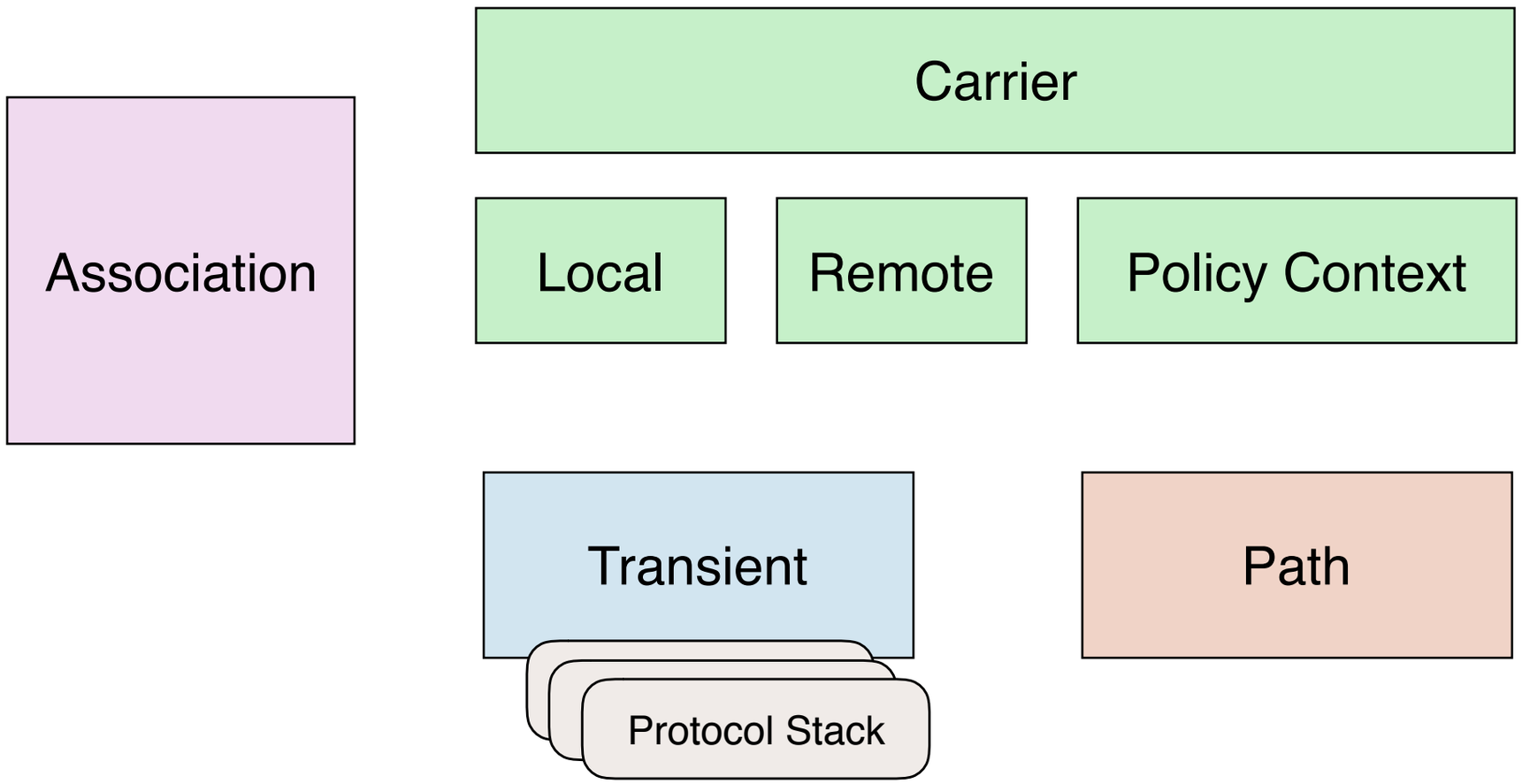
Connection Lifetime Objects

Data I/O API

Connection Patterns

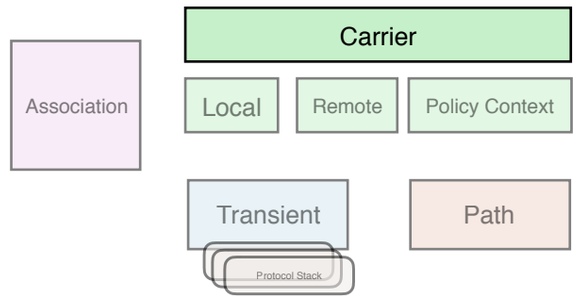
Connection Lifetime Objects

Overview



Connection Lifetime Objects

Carrier



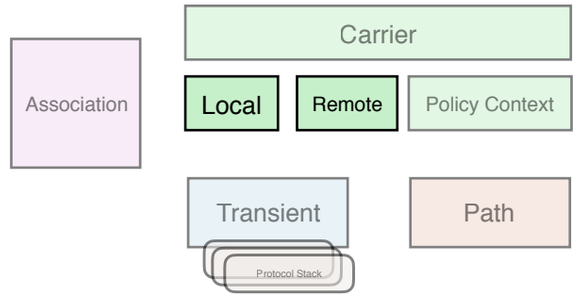
A “Message Carrier” is the primary way to interact with a networking connection, by sending and receiving messages

```
newCarrier = initiate(local, remote, policyContext)
```

This object corresponds to a bidirectional flow of messages the client can interact with, not necessarily a single transport connection

Connection Lifetime Objects

Local & Remote



Local and Remote objects represent the endpoints that messages can be sent to and received from

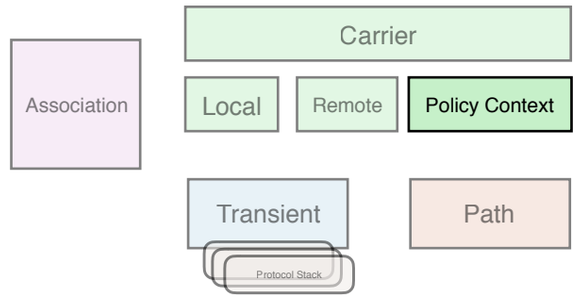
They contain information on where to reach them and any necessary credentials or metadata

```
Local : Port + [Address] + [Certificates]

Remote : Address + Port / Hostname + Port / URL
```

Remotes can be resolved from one form into another, which happens internally to the connection setup

Connection Lifetime Objects

Policy Context

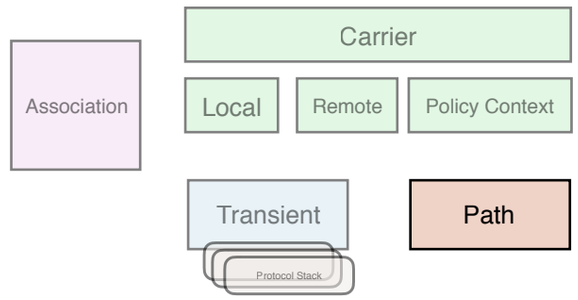


The Policy Context contains parameters describing the client preferences around:

- Interface or local address preferences & prohibitions
- Supported protocols
- Protocol-specific options
- Client metadata

Connection Lifetime Objects

Path



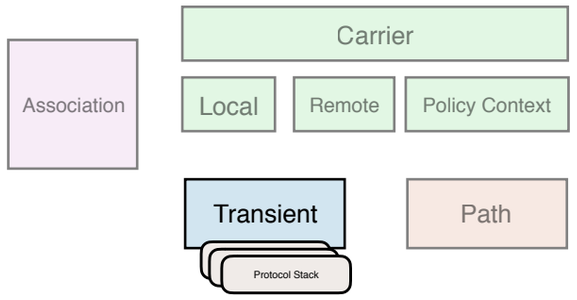
A Path represents the view of a single route through a network to be used for sending and receiving messages

Derived from a policy manager by evaluating the Local, Remote, and Policy Context

Contains MTU, quality properties, and information about how to use the outbound links

Connection Lifetime Objects

Transient



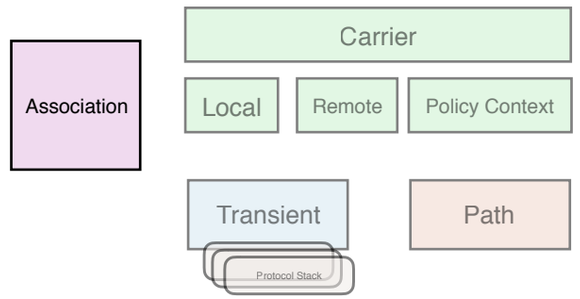
A Transient is an underlying protocol connection that provides the transport for a Carrier's messages. It uses a Path, and is associated with exactly one Carrier.

There may be multiple Transients under a Carrier. They can be raced across resolved Remotes, Paths, and Protocol Stack options. An active Carrier has at least one viable Transient

The Transient is the top level of a protocol stack (that may contain various application- and transport-level protocols). These protocol stacks may share instances across Carriers for multiplexing protocols.

Connection Lifetime Objects

Association



An association holds the long-term state for carriers that share Local/Remote/Policy Context

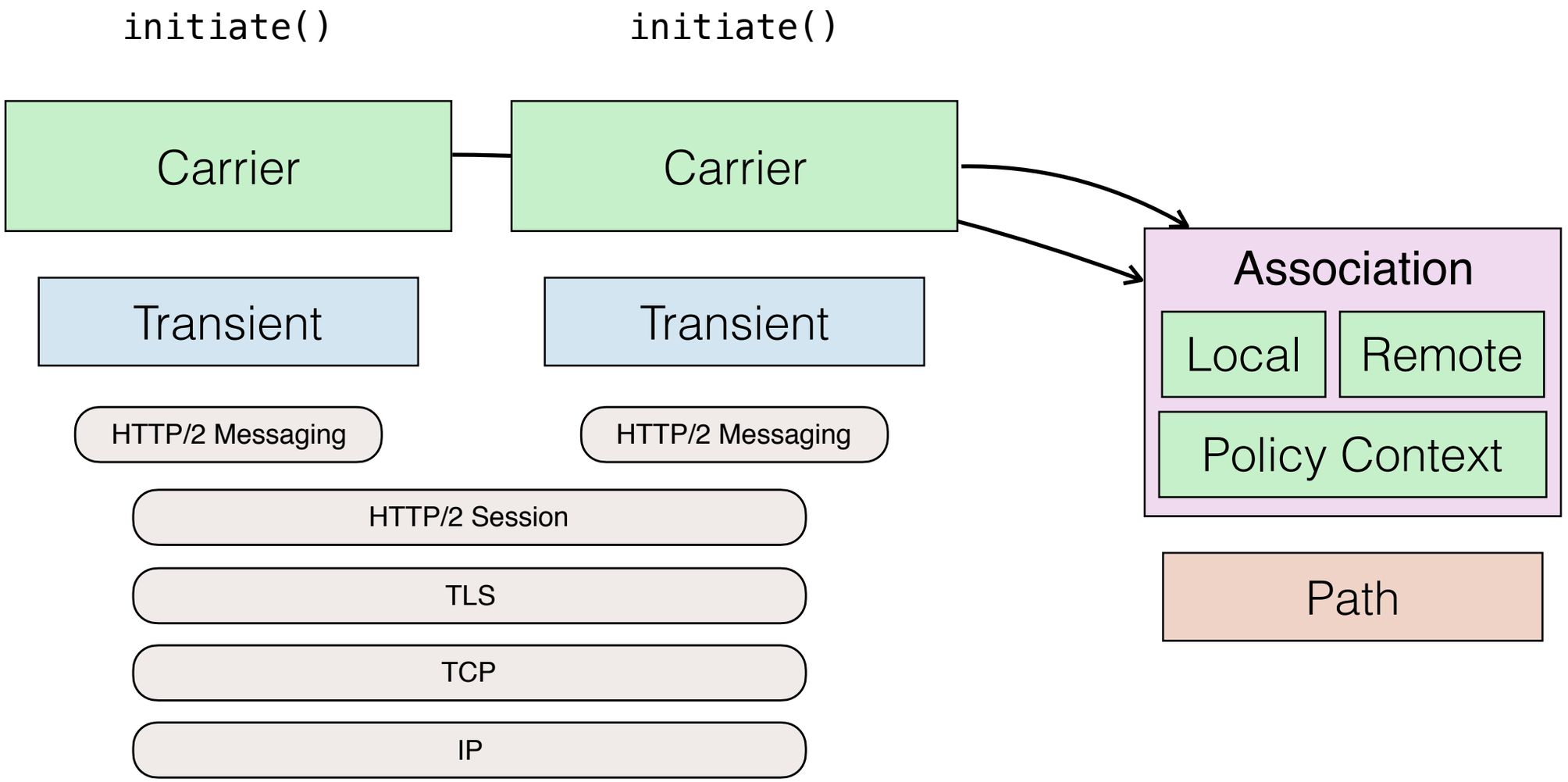
- Cryptographic session resumption
- Previous state about transients and paths

An Association is automatically assigned to each new Carrier. The Association may be exposed to clients, but clients are not required to be aware of it. Preferences for which Carriers share an Association can be set in Policy Context.

Connection Lifetime Objects

Association + Carriers

Multiple carriers in the same association may share elements of their protocol stack



Connection Lifetime Objects

Data I/O API

Connection Patterns

Messages

A message is an atomic unit of information to be communicated across a connection.

Both reading and writing are asynchronous

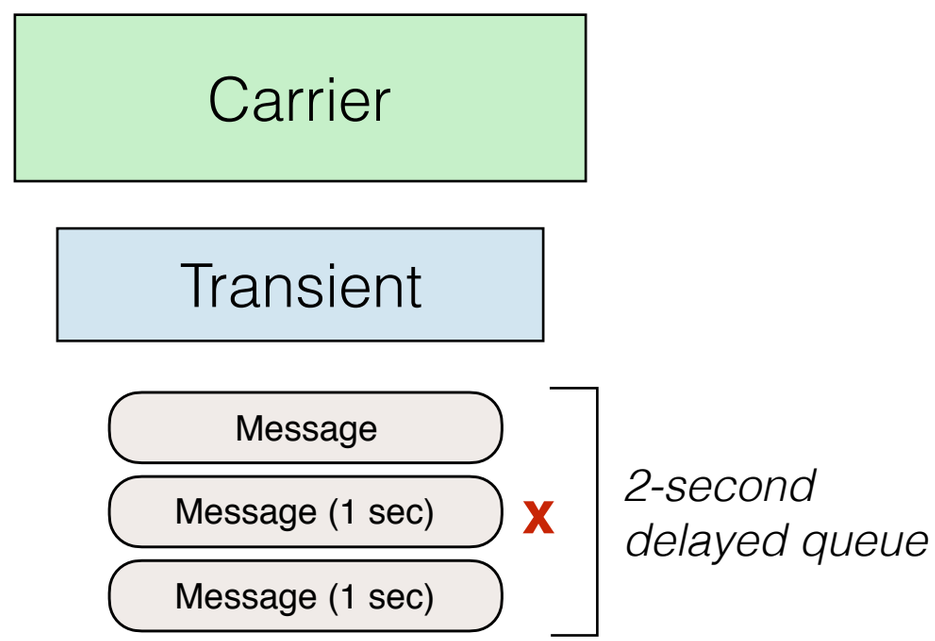
```
carrier.sendMessage(outMessage, sentCallback(),
ackedCallback(), expiredCallback())

carrier.ready(receiveCallback(inMessage, error))
```

Messages

Lifetime & Partial Reliability

Outbound messages can have lifetimes after which they expire if they have not been sent

Messages without lifetimes are sent reliably

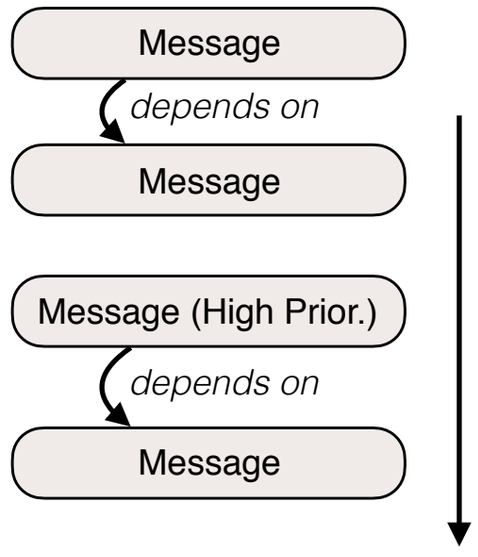


Messages

Priority & Dependence

Outbound messages can have priority, to yield to other messages with more priority

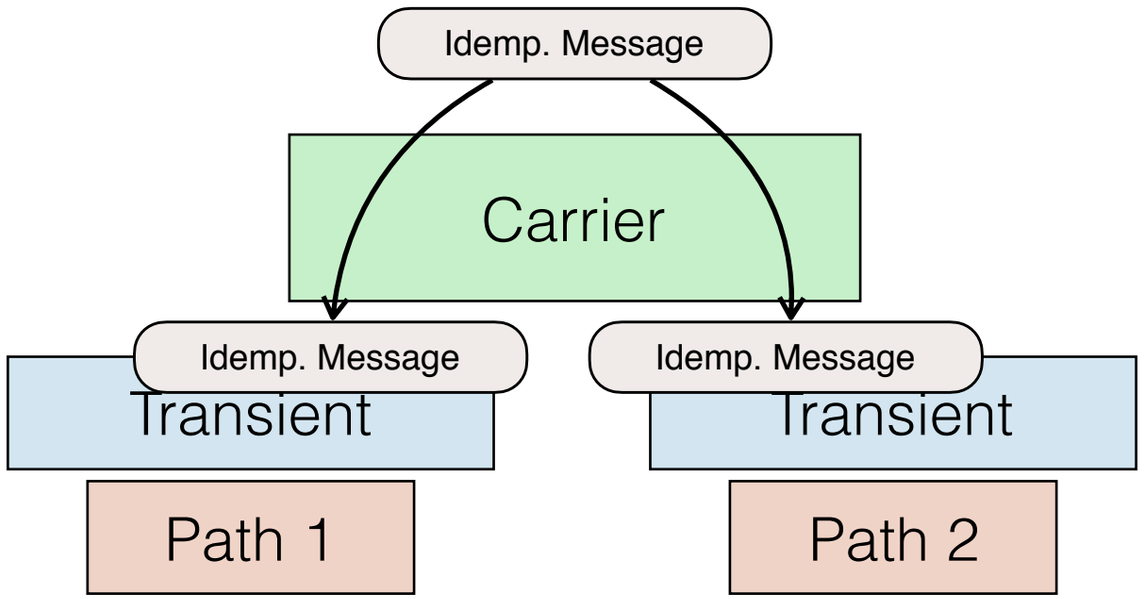
Messages can also specify antecedents which must be sent first



Messages
Idempotence

In order to support Fast-Open/0-RTT protocols, outbound messages may be marked as idempotent

Idempotent messages may be replayed across Transient connection attempts



Messages

Outbound messages can always support lifetime and ordering properties, even over raw TCP streams

The transient will schedule and send messages based on the requirements. If the protocol stack supports messages, it can decide how to parse them, otherwise their content data will be sent as datagrams or on a bytestream

Reading of messages will correspond to how the protocol stack parses incoming data

Connection Lifetime Objects

Data I/O API

Connection Patterns

Patterns

Not all networking apps use the network in the same way!

- Message Carrier (initiator)
  - Forking Message Carriers
  - Stream Carrier
- Listener
- Source
- Sink
- Responder

Patterns

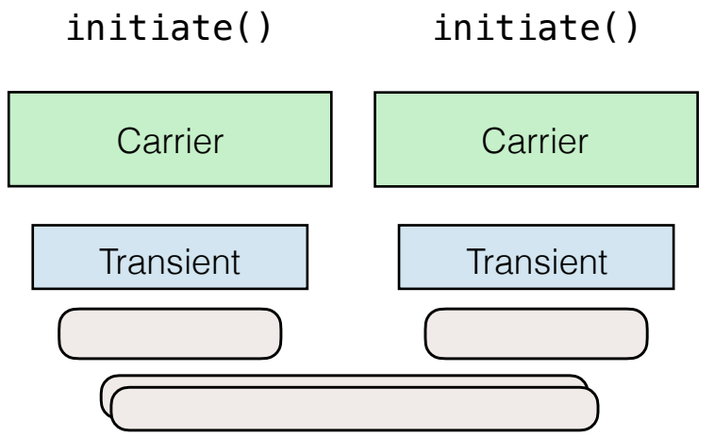
Forking Carrier

In order to explicitly have a multiplexed or multi-stream set of Carriers, an original Carrier may be “forked”

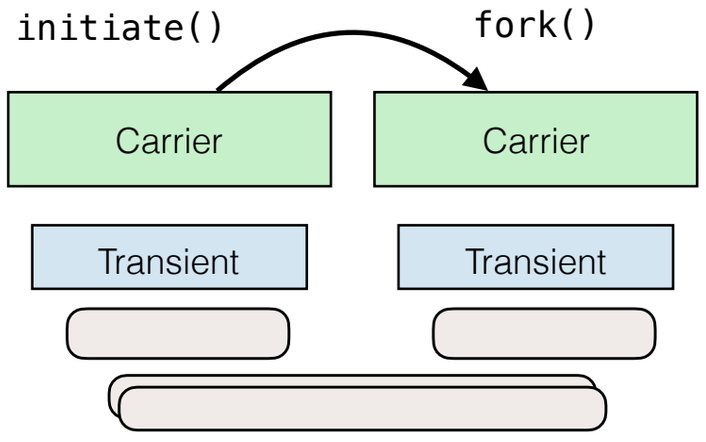
```
secondCarrier = fork(existingCarrier, policyCtxt)
```

Multiplexing may be implicit as well (matching endpoints and policy context, and a protocol that support multiplexing)

Implicit multiplexing



Explicit multiplexing



Patterns

Stream Carrier

A Message Carrier can be irrevocably turned into a Stream Carrier

Allows legacy clients to maintain an abstraction of a byte-stream

Streams must always be reliable and ordered

Patterns

Listener

Listeners are created with only a Local and Policy Context (no Remote), and vends Carriers

```
newListener = listen(local, policyContext, delegate)

    listenerDelegate.accept(newCarrier)
```

Patterns

Source & Sink

Sources are send-only carriers that allow sending multicast messages. They cannot be forked.

Sinks are receive-only carriers that allow receiving

Patterns

Responder

Responders allow a Carrier to receive messages from multiple sources, and send specific replies to these messages

This is a common pattern for servers in client-server interactions, such as responding to DNS queries or HTTP requests

Next Steps

- Experiment with more implementations
- Receive wider review of the API model
- Adopt within TAPS?

- …enter a bright new future of networking transport APIs!