

Post Sockets:

Towards an Evolvable Network Transport Interface

Brian Trammell¹, Colin Perkins², and *Mirja Kühlewind*¹
ETH Zürich¹ and University of Glasgow²



measurement and architecture for a middleboxed internet

measurement

architecture

experimentation



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 688421. The opinions expressed and arguments employed reflect only the authors' view. The European Commission is not responsible for any use that may be made of that information.



Supported by the Swiss State Secretariat for Education, Research and Innovation under contract number 15.0268. The opinions expressed and arguments employed herein do not necessarily reflect the official views of the Swiss Government.



SOCK_STREAM: yesterday's interface



socket

stream

„You can have any ~~color~~ you want, as long as it's ~~black~~.“

— Henry Ford




SOCK_STREAM: yesterday's interface, today

- Synchronous (**we got used to it**)
- Unicast (**nobody cares**, multicast is too hard)
- ~~No framing support~~ (**nobody cares**, apps do this anyway)
- ~~Single-stream~~ (**just open multiple sockets**)
- ~~Single-path~~ (MPTCP **hides this from you**)
- ~~No path abstraction~~ (**nobody cares**, middleboxes don't exist)
- ~~No security~~ (TLS solves all our problems, **right?**)

Simplicity wins: it makes the network look like a file!



SOCK_SEQPACKET: tomorrow's interface, yesterday

- Synchronous (with async event notification!)
 - Unicast or multicast!
 - Framing support!
 - Single- or multiple-stream!
 - Multipath! (for failover)
 - No security
 - No path abstraction
- 
- Bound to Stream Control Transmission Protocol (SCTP),
not extremely deployable in the open Internet today.



Motivations and Goals

- A **transport-** and **platform-independent** API
 - for **present** and **future** transport protocols.
- Support **dynamic selection** of transport protocol stacks
 - like Happy Eyeballs, but happier.

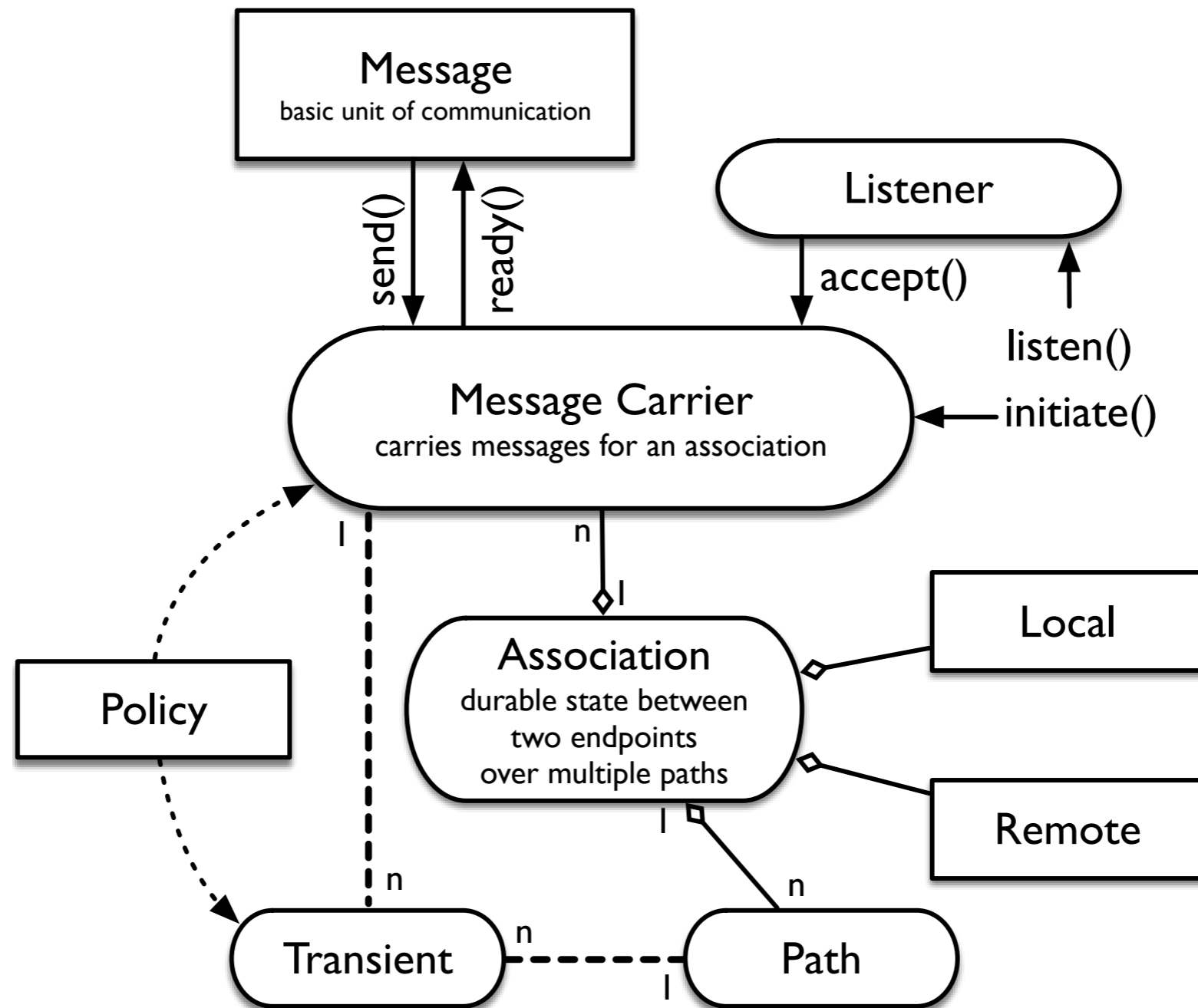


A few insights about transport APIs

- **Applications deal in messages** of arbitrary size
- Message reception is **inherently asynchronous**
- The network of the future is **explicitly multipath**
- Applications **don't care about the transport layer**
- Transport must **guarantee security properties**



Abstractions and Relationships

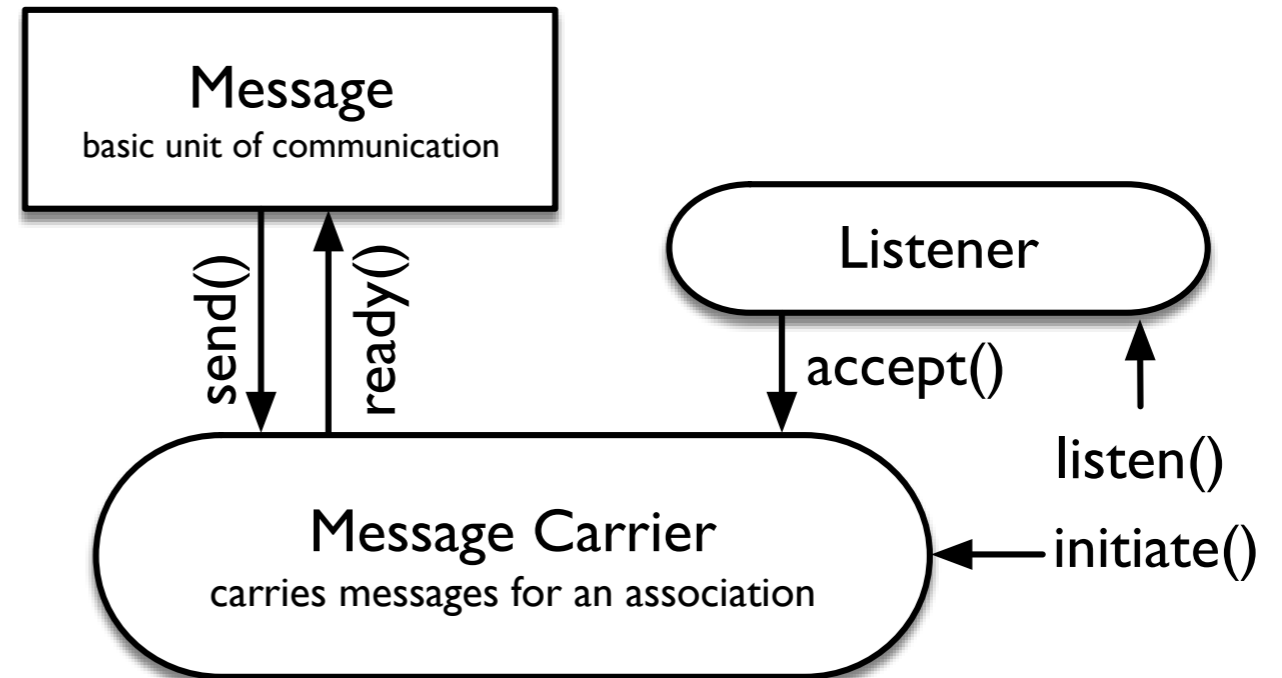




Message Carriers

Logical communications endpoint for a group of messages

- created actively via `initiate()`
- passively via `listen()` / `accept()`



- Special carriers for common application types
 - *source*: unidirectional send-only
 - *sink*: unidirectional receive-only
 - *responder*: server for common request/response protocols



Messages

collection of bytes, all delivered together

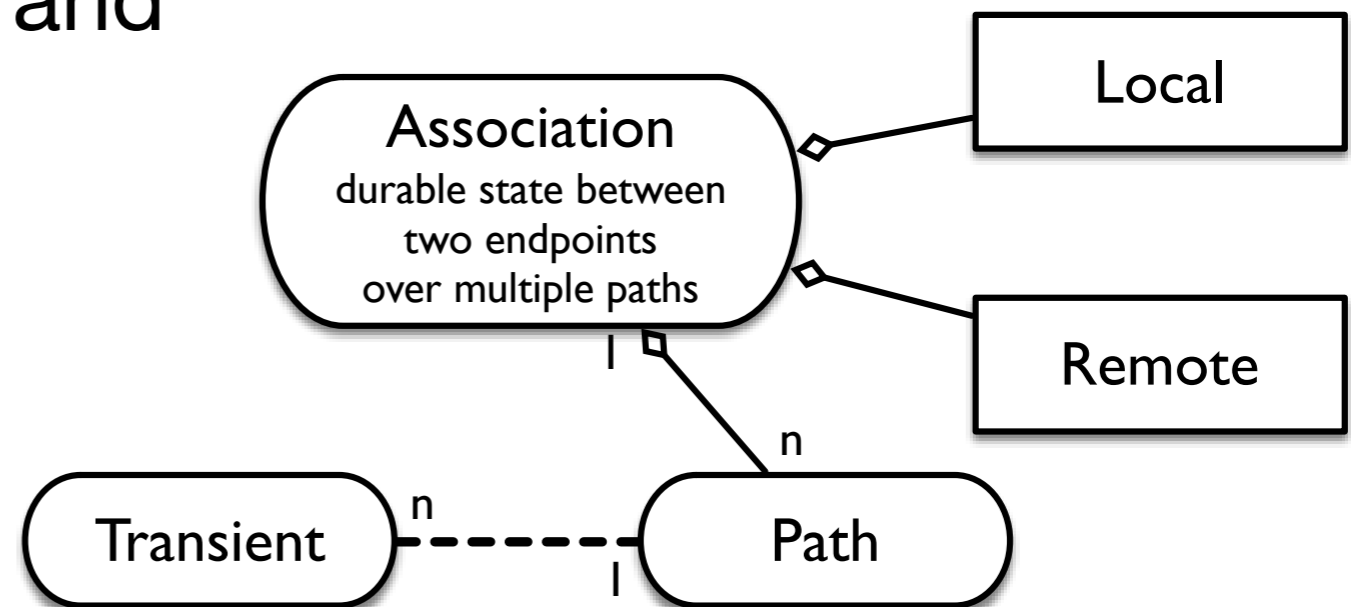
- Have set of optional properties including
 - **Lifetime:** maximum delay to remote for partial reliability; 0 = fully reliable delivery (default)
 - **Niceness:** relative priority class, 0 = max (default)
 - **Immediacy:** please don't coalesce
 - **Idempotence:** okay to send multiple times (i.e. for 0-RTT data)
- Properties allow sending scheduler flexibility
- Event callbacks on message reception, expiry, acknowledgment
- Message boundary preserved by the API



Associations (and Paths)

long-term state between a pair of logical endpoints

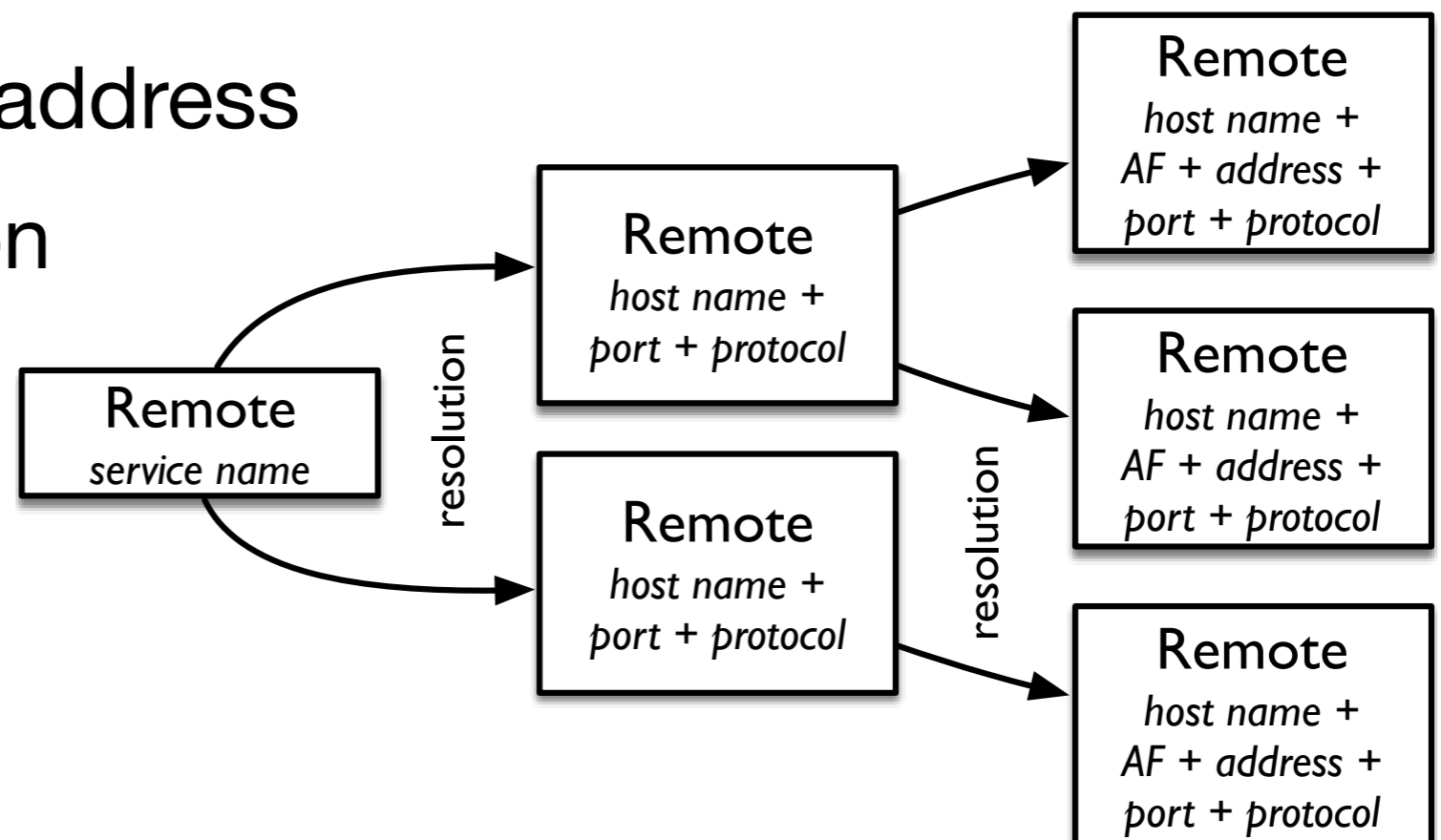
- Associated with one local and one remote endpoint
- e.g. cached cryptographic state
- Information about paths between endpoint pairs
- cached measurements (e.g. loss, latency, bandwidth)
- information discoverable through rendezvous





Locals and Remotes

- Local: “who am I?”
 - Identity, interface, associated properties
- Remote: “who are you?”
 - Identity and name/address
 - Recursive resolution

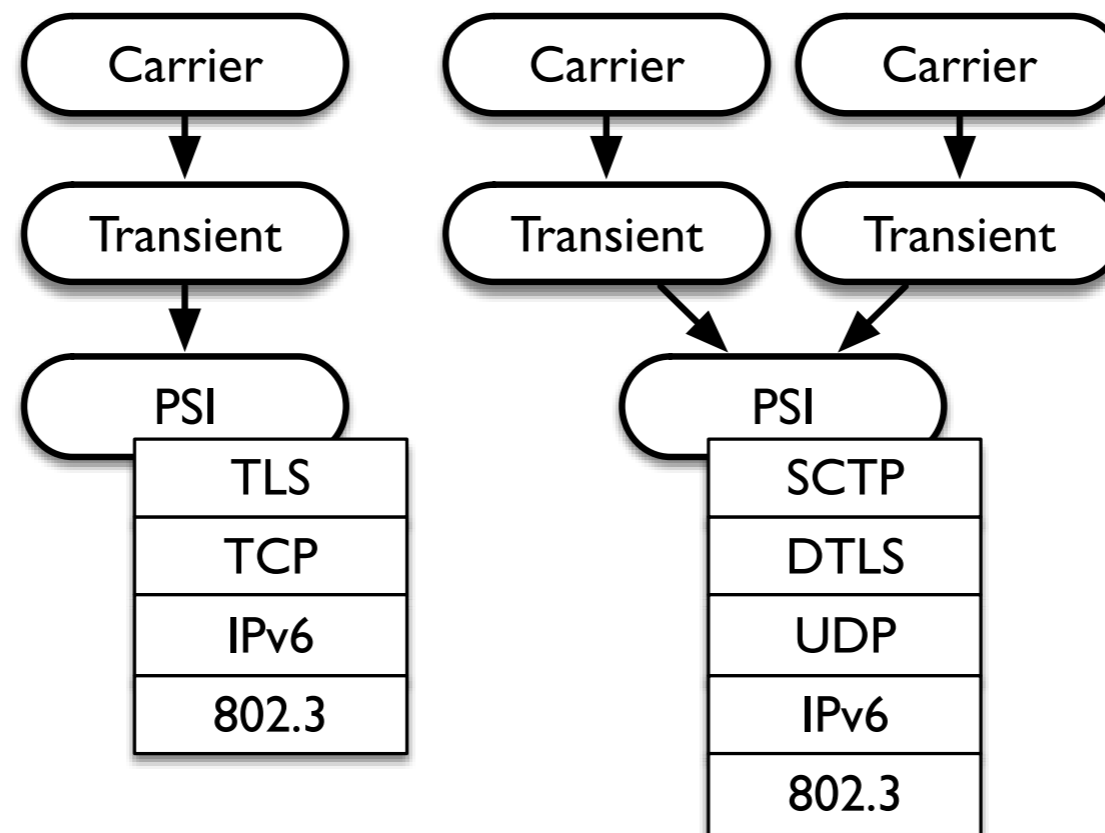




Transients

binds a carrier to the transport protocol stack instance

- *Protocol Stack Instance (PSI)*: set of instantiated protocols that will carry the packets containing messages

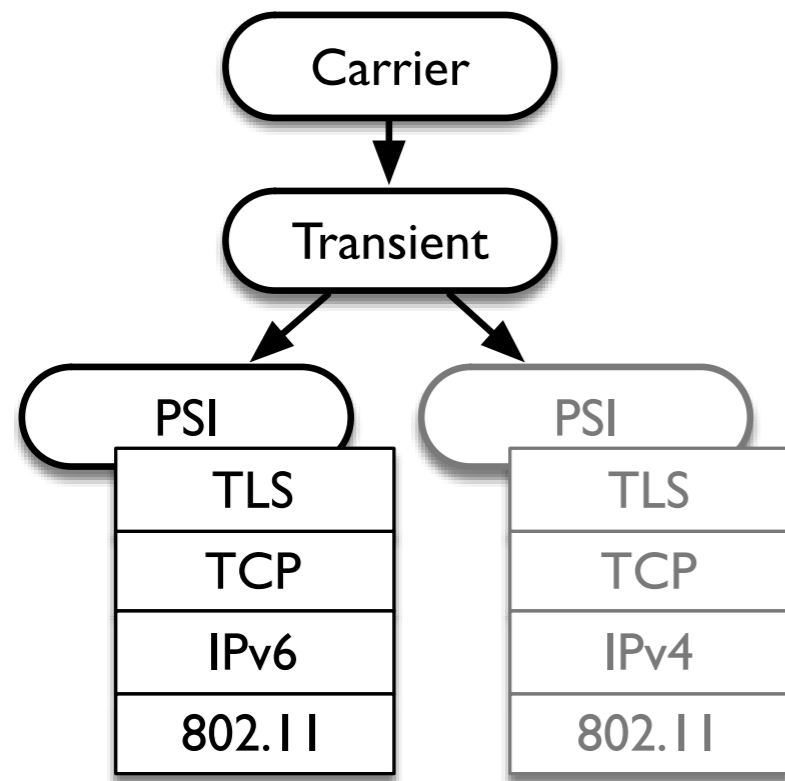


(a) Transient bound to a PSI

(b) Carrier multiplexing over a multistreaming protocol



Transient Establishment Lifecycle



(c) Multicandidate communication during association establishment

- During connection establishment, a transient may use multiple candidate PSIs to manage connection racing
- The “winning” PSI becomes bound to the transient after establishment



Policy

Expression of preferences for carriers and transients

- Local and remote identity constraints
 - Interface and path selection
 - Transport protocol selection and configuration
-
- ```
classDiagram
 class Policy
 class MessageCarrier["Message Carrier (carries messages for an association)"]
 class Transient
 Policy ..> MessageCarrier
 Policy ..> Transient
 MessageCarrier "1" ..> "n" Transient
```
- Multiple domains
    - application policy, system policy, user policy



# Interoperability: Message Boundaries and Streams

- Post promotes message framing to a transport service.
  - But no other API does, and many existing transports don't,
  - and it might be nice to interop.
- **Solution:** Allow applications to push deframing logic down into the stack, when necessary
- Post sends messages.
  - But sometimes what you have *really* is a stream.
- **Solution:** Carriers can be *morphed* into Streams
  - with platform-specific read()/write()/close() API
  - Stream morphing is irrevocable





# What's next?

- Post provides for...
  - asynchronous message reception
  - multi-path & multistreaming
  - connection establishment & resumption
- We still need...
  - generic light-weight framing protocol & negotiation
  - mechanisms and policies for protocol and path selection
  - separation of data transmission and support functions, e.g. crypto context

**Higher layer of abstraction enables application developers easier access to novel transports!**



# Does this sound familiar to Apple geeks?

At **Apple's WWDC** last week

- “User-Space Networking” in the current betas of iOS 11
- Transport and IP co-located with security & application protocols
- No BSD socket anymore!
- First step towards more flexibility and dynamic protocol selection!
- Also see

<https://datatracker.ietf.org/doc/draft-trammell-taps-post-sockets/>

