Post Sockets

a way to think about the world after sockets

draft-trammell-post-sockets-00

TAPS WG, IETF 97 서울, Wednesday 16 November 2016

Brian Trammell, Colin Perkins, Tommy Pauly, and Mirja Kühlewind

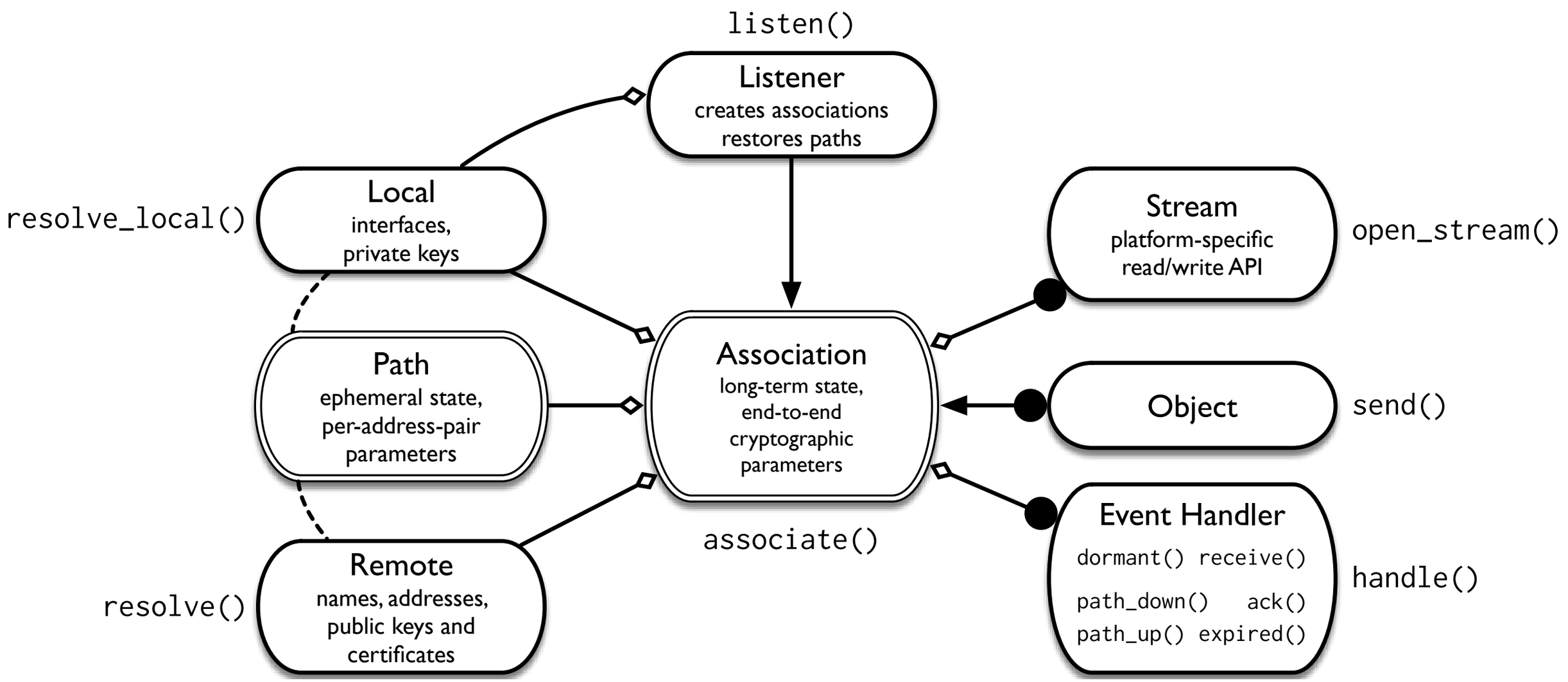
(with thanks to Jason Lee and Laurent Chuat, and acknowledgments to the authors of TCP Minion and of SCTP and its extensions, from whence many of these ideas came)

A few insights

- ***Applications deal in objects*** (messages) of arbitrary size
  - Files, assets, media frames, etc. may depend on each other, but usually don't require a strict ordering.
  - Streams exist too, but only when the underlying source of data is of unknown length and not easily divisble into objects.
- The network of the future is ***explicitly multipath***.
  - Applications must have access to the properties of these paths.
  - (And may be able to communicate with the path about these properties)
- Future transports must ***guarantee security properties.***
  - Path elements must not be able to see transport-layer metadata.
- Message reception is ***inherently asynchronous.***
  - Present scalable programming models enable (and require!) async IO.

Abstract Programming Interface

Classes and Entry Points



Abstract Programming Interface
Object and Stream properties

- Objects and streams have a **niceness**
  - Nicer `send()`s/`write()`s yield to less nice
- Objects have a **deadline**
  - An object will be cancelled if it cannot be realistically received before this deadline
  - Infinite-deadline objects are fully reliable
- Objects may have **antecedents**
  - other objects which should be sent before

Transport Independence

- Only two requirements for transport on the wire:
  - Framing for objects
  - Some (non-address) way to identify associations
- Assumption that the transport protocol provides encryption for payload confidentiality and public header integrity protection.
- Can make use of other transport features on demand:
  - Multipath load balancing and migration
  - Multistreaming for objects and streams
  - PLUS for path property exposure
- Object properties (niceness, deadline, dependencies) are sender-side only; path properties can be derived locally too.

Can TAPS use Post?

- Post places some requirements on the transport:
  - Won't run over TCP/MPTCP without a framing/association ID shim.
  - Not the generic API TAPS envisions, if TAPS must work with unmodified transport protocols.
- Can be modified for application-layer failover:
  - "Transport supports only `open_stream()`."

Work to do

- Define path properties beyond “up/down”
  - Defined properties: interface cost/preference
  - Measurable properties: RTT/loss rate
    - Exposed properties: Lo/La, etc. via PLUS
- Pilot implementation and experimentation
  - Post for QUIC/TLS over PLUS in Go?
  - Post for (Post stream shim) over TCP?
  - Within auspices of MAMI project, by mid-2018.