Post Sockets

A generic API for multipath-cooperative communication

Brian Trammell, Networked Systems and Network Security Groups, ETH Zürich
with Laurent Chuat and Jason Lee, Network Security Group, ETH Zürich
and Mirja Kühlewind, Networked Systems Group, ETH Zürich

TAPS WG, IETF 96, Berlin, Thursday 21 July 2016

mami
measurement and architecture for a middleboxed internet

ETH zürich

SOCK_STREAM:
yesterday's interface

- Synchronous
- Unicast
- No framing support
- Single-stream
- Single-path
- No path abstraction
- No security
- Implicit measurability

- But it makes the network look like a file. Simplicity wins!

SOCK_STREAM:

yesterday's interface, today

- Synchronous
- ~~Unicast~~ (nobody cares, multicast routing, security too hard)
- ~~No framing support~~ (nobody cares, apps do this anyway)
- ~~Single-stream~~ (just open multiple flows)
- ~~Single-path~~ (MPTCP might actually deploy…)
- ~~No security~~ (TLS/OpenSSL solves all our problems, right?)
- No path abstraction

- ***Can we do better than this?***

SOCK_SEQPACKET:
tomorrow's interface, yesterday

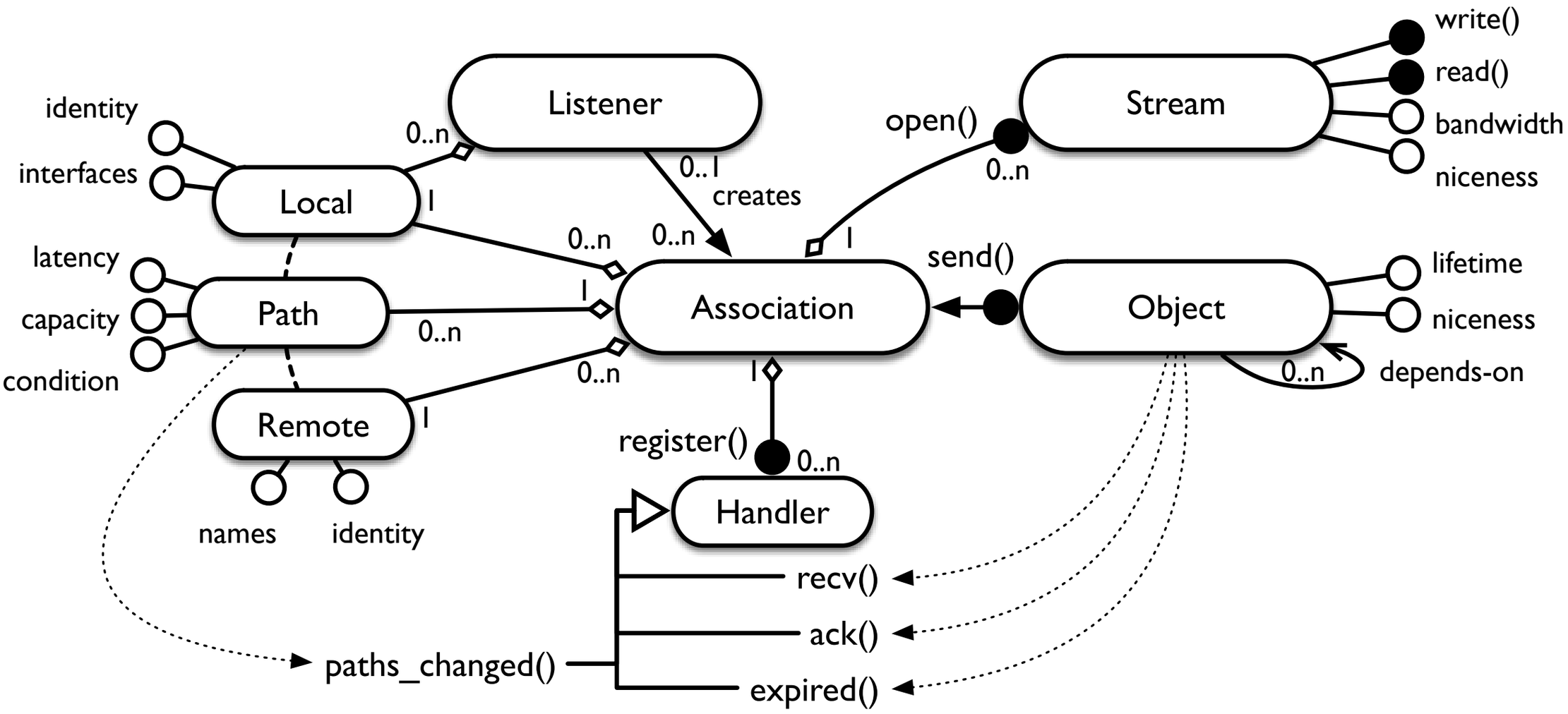
- Synchronous (with async event notification!)
- Unicast or multicast!
- Framing support!
- Single- or multiple-stream!
- Multipath! (for failover)
- No security
- No path abstraction
- Bound to Stream Control Transmission Protocol (SCTP),
  not extremely deployable in the open Internet today.

- ***Let's go back to the interface…***

A few insights
(or, alternately, silly assumptions)

- ***Applications deal in objects*** (messages) of arbitrary size
  - May depend on each other, but don't have a strict stream ordering
  - Let the transport layer solve the optimization problem!
- The network of the future is ***explicitly multipath***.
  - Applications must have access to path properties.
- Future transports must ***guarantee security properties.***
  - "Bolted-on" security (TLS) adds complexity, latency.
  - 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.

Post Sockets: Abstractions



Abstractions

- ***Associations*** represent communication state among a group (pair) of network-connected processes:
  - ***Remote*** and ***Local*** Public key and certificate information
  - Session and cryptographic state for fast resume
  - Currently available ***Paths*** (or interface addresses)
  - Callbacks for association events (object receive, etc)
- ***Listeners*** allow for passive opening of Associations
- ***Objects*** given to one end of an association appear at the other, subject to priority, lifetime, and dependency constraints.
  - Objects may require multiple segments to transport.
  - Object boundaries guaranteed to be preserved.
- ***Streams*** over Associations allow bandwidth reservations for nonmaterialized, streaming data to coexist with Objects.

Entry Points and Events

- Associations created with `associate()`, given Local, Remote.
- Most calls are conceptual methods on Association:
  - `.send()`: send an object
    - object properties include `lifetime, niceness, antecedents`
  - `.open()`: get a new stream compatible with platform's stream IO API
    - stream properties include `bandwidth, niceness`
  - `.register()`: register a handler for a given event
    - event types include `recv, ack, expired, paths-changed`
- Listener (created with `listen()`):
  rump Association with a single event, `accept`.
- Local and Remote API are architecture-dependent.

Post Sockets and TAPS

- TAPS allows you to select transport protocols…
- …but if each protocol has its own API, this is not very useful.

- The NEAT API is one solution to this problem.
- PostSockets represents an alternate (and more radical) approach.

(Potential) Implementations

Implementation/ Features	over TCP	over SCTP (or SCTP over UDP over DTLS)	native transport over UDP userland/ PLUS	native transport over UDP in-kernel/ PLUS
Async Receive	coroutines in userland	coroutines in userland	coroutines in userland	zero copy w/ coroutines
Object Framing and Interleaving	object header in TCP stream (can deadlock)	provided by SCTP	object header with native segmentation	object header with native segmentation
Object Lifetime and Reliability	sender-side-only expiry	sender-side-only expiry, provided by SCTP-PR	expiry at sender, receiver, and on-path	expiry at sender, receiver, and on-path
Multistreaming	multiple TCP sockets	multiple SCTP streams, single association	via object interleaving	via object interleaving
Path Primacy	interface only no path info MPTCP?	interface only no path info SCTP path failover	interface only path info via PLUS	interface only path info via PLUS
Security	using TLS	using DTLS	using DTLS 1.3	using DTLS 1.3