An Architecture for Transport Services

draft-pauly-taps-arch-00

Tommy Pauly
TAPS
IETF 101, March 2018, London

Recap from IETF 100

The last milestone in the TAPS charter is about mechanisms to allow Transport System to be deployed.

Several separate drafts captured parts of this work, including:

- draft-fairhurst-taps-neat
- draft-trammell-taps-post-sockets
- draft-pauly-taps-guidelines
- draft-grinnemo-taps-he
- draft-tiesel-taps-socketintents

How do we reconcile and/or combine these efforts?

Proposal

Combine and distill existing work into three drafts:

1. **Architecture**, to explain the approach and basic concepts *(draft-pauly-taps-arch-00)*

2. **API**, to explain in detail how applications use a TAPS system *(draft-trammell-taps-interface-00)*

3. **Implementation**, to explain in detail how to implement a TAPS system *(draft-brunstrom-taps-impl-00)*

Architecture Draft

Design Principles cover the constraints on the approach, based on Surveys and MinSet

Concepts & Terminology define the framework

- API concepts, items which may be exposed to applications
- Implementation concepts, items which are likely internal to a TAPS system

Design Principles

1. Common APIs for Common Features

Common operations across multiple protocols should use the same API

“Write code once, re-use across protocols”

This common set must at least include the minimal set from the MinSet draft

Design Principles

2. Access to Specialized Features

Non-generic protocol options must have some way to be set, otherwise this API would be less useful than sockets.

Protocol-specific features need to be exposed in a way that doesn't limit protocol selection or flexibility

"If you choose protocol X, use option X_A"

Some specific features, like security, may limit protocol selection

Design Principles

3. Scope for API and Implementation Definitions

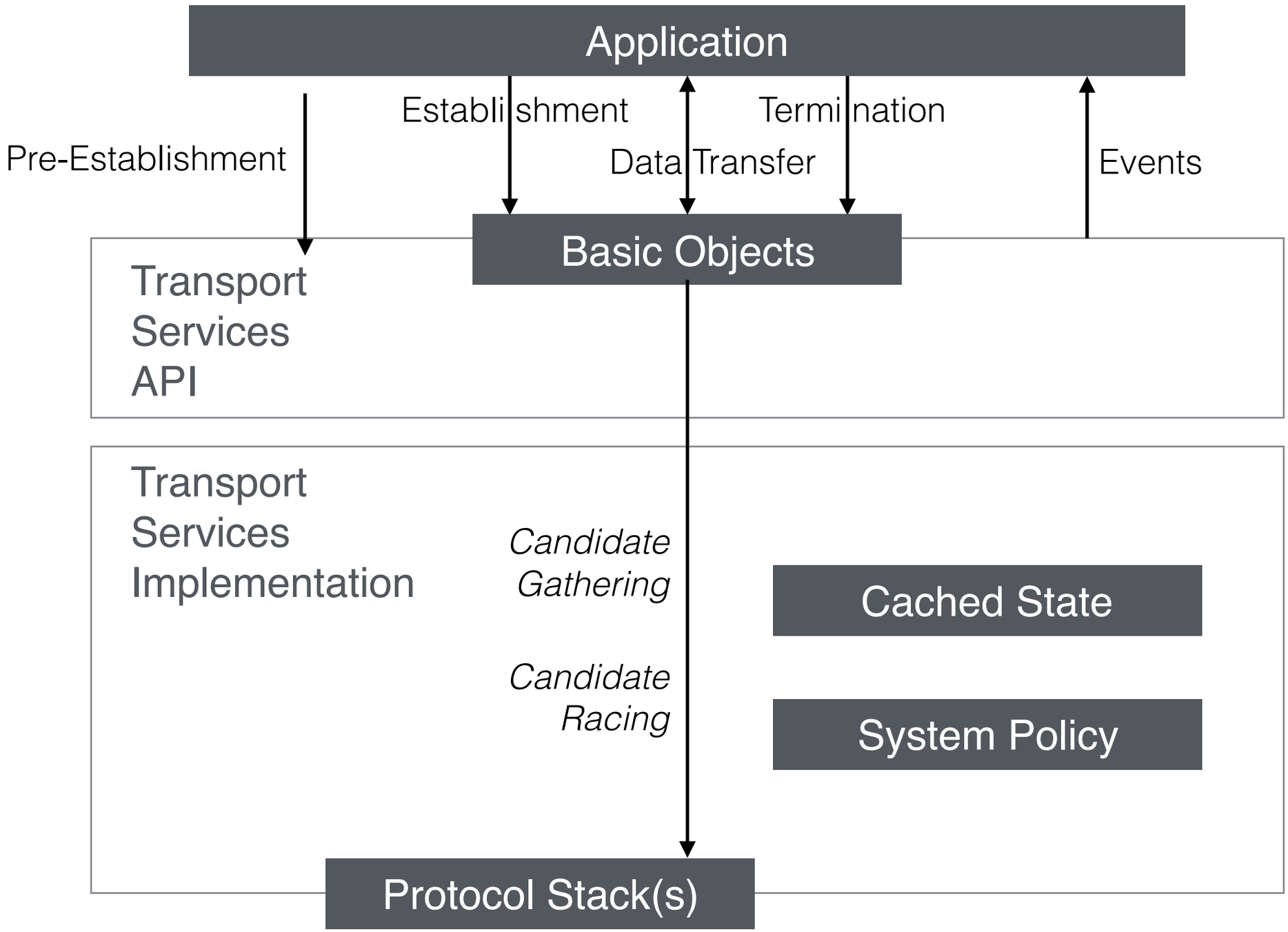
API must include MinSet (thus, basicTCP/UDP operations). Not all implementations must offer all protocols, however.

API should avoid referencing specific protocols

Implementation should define mappings of API onto specific protocols, which will be extended over time

No new protocols being defined, and no requirement to use a TAPS system on both sides

Architecture Diagram



API Concepts

Basic Objects

Preconnection is an object that defines the endpoints and properties that are usable by Connections and Listeners

A **Connection** represents an active transport protocol instance that can send and/or receive Messages between a Local Endpoint and a Remote Endpoint

A **Listener** accepts incoming transport protocol connections from Remote Endpoints and generates corresponding Connection objects.

API Concepts

Pre-Establishment

Local and **Remote Endpoints** define how an application represents local identity or remote destinations, with varying degrees of specificity

Path Selection Properties tell the TAPS system which paths to allow and prefer

Protocol Selection Properties tell the TAPS system which protocols to allow and prefer

Protocol Specific Properties contain extended options for specific protocols, without altering selection

API Concepts

Establishment

Initiate is the act of starting a Connection (either explicitly outbound, or setting up state for peer-to-peer protocols)

Listen is the act of marking the system as willing to accept inbound Connections

API Concepts

Data Transfer

Message is a unit of data that can be sent or received, which has in-order content. Boundaries may or may not be understood by protocols.

Send is the act of transmitting data from a Message on a Connection

Receive is the act of indicating that the application wants to receive data for a Message, which will be delivered asynchronously

API Concepts

Event Handling

Connection Ready and **Failed** events communicate the ability of a Connection to send and receive data

Message Sent and **Received** events indicate the status of data transfer

Path Properties Changed is an event to let an application know when properties of the connection have been updated

API Concepts

Termination

Close is the act of gracefully tearing down a Connection, by communicating to the remote endpoint if applicable

Abort is the act of immediately tearing down a Connection and not sending any outstanding data

Implementation Concepts

Candidate Gathering is the process of determining which **Protocols** and **Paths** are permissible and preferable based on API properties and **System Policy**

Candidate Racing is the process of attempting the various **Protocol Stacks** during establishment

Connection Groups represent Connections with shared properties, which may be multiplexed