The implementation of an extensible socket API for modern networks.

SRE. Rayhaan Jaufeerally, Prof. Dr. Adrian Perrig, SRE. Brian Trammell

March 2019

The BSD socket API

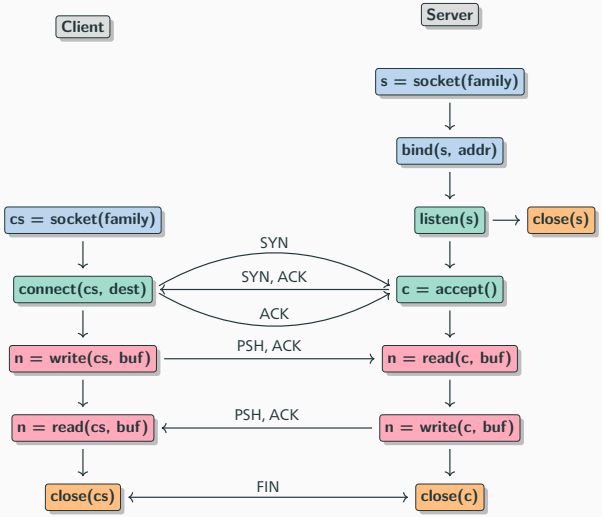

Figure 1: Illustration of an example usage of the BSD sockets API, annotated with TCP protocol semantics.

Problems with the BSD socket API

Simple design simple usage,

Expressing complex ideas is hard:

- `setsockopts` or `ioctls` are required for complex options,
  Such as `TCP_NODELAY` or `SO_REUSEADDR`,

Only one address pair for a connection,

No path selection, network property selection,

Only one uplink.

Our contributions

TAPS was already specified by TAPSWG.

This thesis:

- TAPS-like implementation in the Go programming language,
- Supporting infrastructure (e.g. beaconing service),
- SPAIR6,
- Demo application.

What does the Go TAPS implementation contain?

Support for UDP, TCP, TLS, SCION UDP,

Interface selection using information about local interfaces,

End to end path selection in SCION during dial,

Racing not fully implemented (only helper function for clients to “race” dials).

SPAIR6

Idea for path selection

End user AS's already have more than one uplink, but paths are automagically chosen,

It should not be too hard to expose the path data from the BGP router to the end host,

The end host could signal back to the network which path it wants to use.

The SPAIR6 architecture

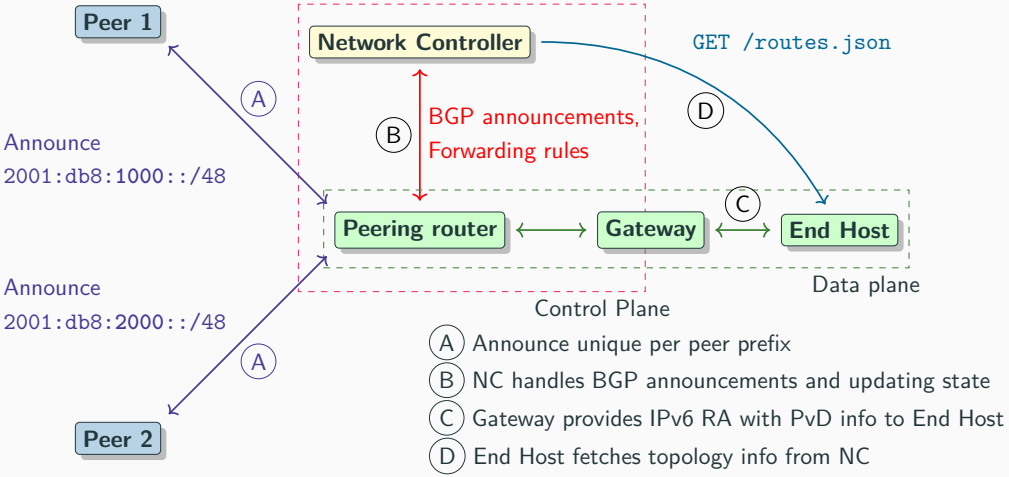

Figure 2: An illustration of the Multiprefixing approach.

A TAPS Implementation

TAPS interface overview

`Dialer` specifies what properties are required and what the destination is,

Resolution of paths and candidate destinations is handled seamlessly,

Interface exposes a message oriented transport.

Dialer interface

```
// Dialer is used to establish a connection
type Dialer struct {
    PropSpec    *PropertySpecification
    CapProf     CapacityProfile

    RequireDNSSEC bool
    RequireDoH    bool

    Local  net.Addr
    Remote net.Addr
    Creds *Credentials

    FastOpenReq  []byte
    FastOpenResp []byte

    SCIONPathChooser func([]*sd.PathReplyEntry) *sd.PathReplyEntry
}
```

Connection interface

```go
// Conn defines an interface all TAPS transports must provide.
type Conn interface {
    net.Conn
    // TransportProperties returns the TAPS transport properties supported.
    TransportProperties() []TransportProperty
    // Dial initializes the connection using the provided dialer.
    Dial(d *Dialer) error
    // SetFramer sets a TAPS framer to be used in Send and Recv.
    SetFramer(f Framer)
    // TAPS specific Send.
    Send(message interface{}, opts []MessageProperty, done chan MsgRef, err chan
     ↪  MsgFail) MsgRef
    // TAPS specific Recv.
    Recv(message chan interface{}, err chan error)
}
```

Implementation of Lifetime properties

In this thesis lifetime expiry is checked in the TAPS library before sending.

Other guarantees can be gained from deeper integration with the transport stack, or wider network.

Implementation of Transmission profiles

We implemented transmission profiles by mapping to DSCP codepoints.

Future work would be to integrate with more advanced systems such as COLIBRI.

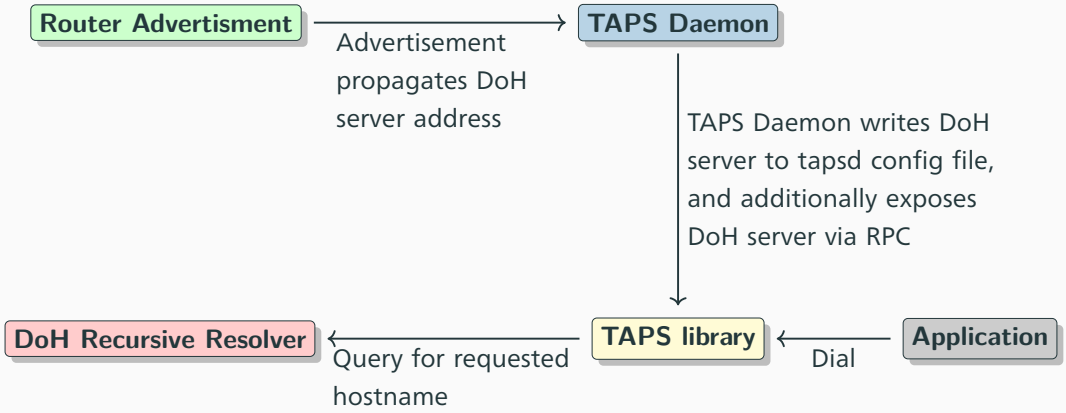

Figure 3: Diagram showing the propagation of DNS over HTTPS configuration information through the system.

Name resolution in TAPS

Name resolution handled completely independently of application

Can take progressive steps to deploy new security protocols without applications noticing,

Supports DNS over HTTPS if network advertises it,

Supports DNSSSEC by default,

Supports RAINS lookups for SCION addresses.

Discovering network services

Global system overview

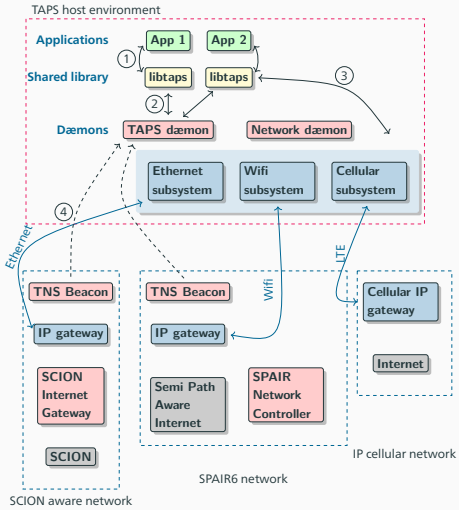

Figure 4: Global system overview

The TAPS Dæmon

Listens for IPv6 Router Advertisement messages with the Provisioning Domain (PvD) option,

Fetches the URL contained in the PvD option and decodes the JSON document,

Populates a map of interfaces with the contents of the JSON document,

Performs lookups on behalf of applications as to which interface should be used,

Data elements contained in PvD

`SCION_ONHOST` — Use local SCION resources,

`SCION_IPGWW` — Use SCION IP Gateway,

`SPAIR6_URL` — Location of the SPAIR6 route server,

`DoH_URL` — DNS over HTTPS resolver to use.

TAPS dæmon interface

```
1  // InterfaceQueryRequest is used to query which interface to use.
2  type InterfaceQueryRequest struct {
3      // RequireDNSSEC ensures any addresses returned are validated via DNSSEC.
4      RequireDNSSEC bool
5      // RequireDoH ensures queries are only made via DNS over HTTPS to ensure privacy
       ↪  and integrity to the resolver.
6      RequireDoH bool
7      Destination   string
8      MTU           uint
9      ASPathExact   []uint32
10     ASPathSome    []uint32
11     ASPathExclude []uint32
12     // LatencyOptimized prefers links with low latency.
13     LatencyOptimized bool
14     // BandwidthOptimized prefers links with high bandwidth.
15     BandwidthOptimized bool
16 }
```

Questions?

Thank you!