TAPS

Retiring the BSD Socket API

Philipp S. Tiesel <phils@in-panik.de>

The BSD Socket API

The BSD Socket API

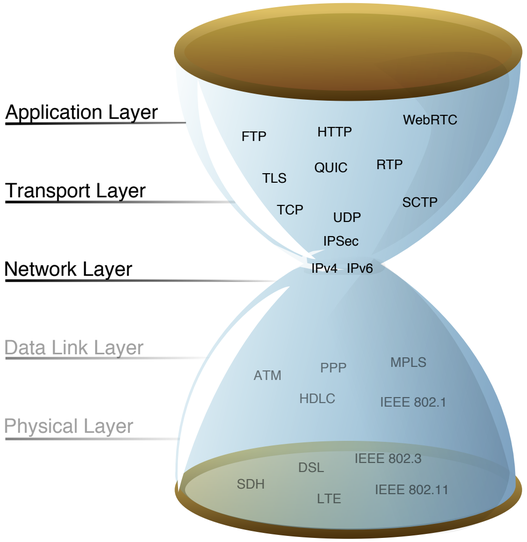
- Originates from 4.2 BSD from 1983
- Integrates with the UNIX file system API
- Template for most of today's networking APIs

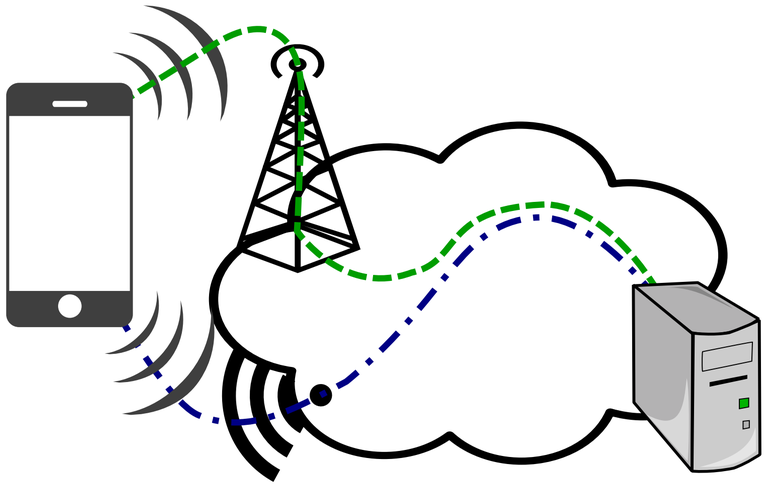
```c
#include "unp.h"
int tcp_connect (const char *host, const char *serv)
{
    int sockfd, n;
    struct addrinfo hints, *res, *ressave;
    bzero(&hints, sizeof (struct addrinfo));
    hints.ai_family = AF_UNSPEC;
    hints.ai_socktype = SOCK_STREAM;
    if ( (n = getaddrinfo (host, serv, &hints, &res)) != 0)
        err_quit("tcp_connect error for %s, %s: %s", host, serv, gai_strerror (n));
    ressave = res;
    do {
        sockfd = socket (res->ai_family, res->ai_socktype, res->ai_protocol);
        if (sockfd < 0)
            continue;           /*ignore this one */
        if (connect (sockfd, res->ai_addr, res->ai_addrlen) == 0)
            break;              /* success */
        close(sockfd);          /* ignore this one */
    } while ( (res = res->ai_next) != NULL);
    if (res == NULL)            /* errno set from final connect() */
        err_sys ("tcp_connect error for %s, %s", host, serv);
    freeaddrinfo (ressave);
    return (sockfd);
}
```

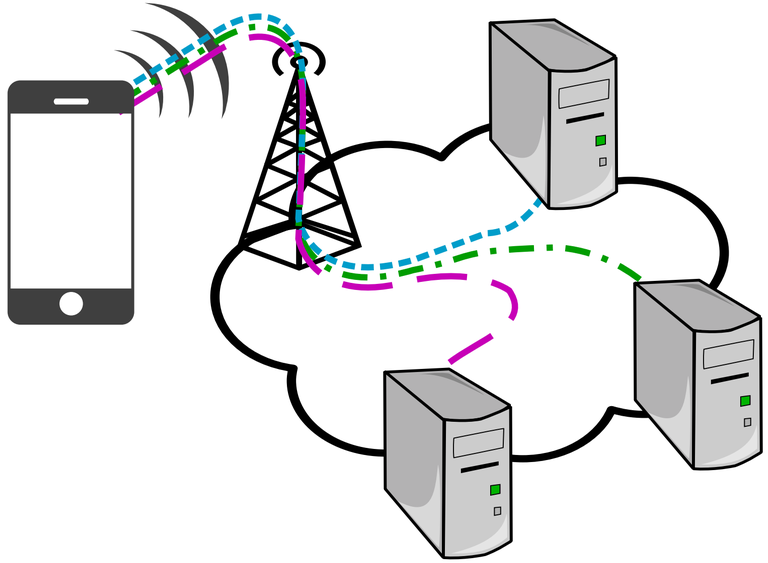
Stevens et at. – UNIX Network Programming Volume 1, Third Edition – The Sockets Networking API

What’s wrong with that?

Today’s Internet Transport







Protocols **Paths** **Endpoints**

Everything should be encrypted

```c
#include "unp.h"
int tcp_connect (const char *host, const char *serv)
{
    int sockfd, n;
    struct addrinfo hints, *res, *ressave;
    bzero(&hints, sizeof (struct addrinfo));
    hints.ai_family = AF_UNSPEC;
    hints.ai_socktype = SOCK_STREAM;
    if ( (n = getaddrinfo (host, serv, &hints, &res)) != 0)
        err_quit("tcp_connect error for %s, %s: %s", host, serv, gai_strerror (n));
    ressave = res;
    do {
        sockfd = socket (res->ai_family, res->ai_socktype, res->ai_protocol);
        if (sockfd < 0)
            continue;           /*ignore this one */
        if (connect (sockfd, res->ai_addr, res->ai_addrlen) == 0)
            break;              /* success */
        close(sockfd);          /* ignore this one */
    } while ( (res = res->ai_next) != NULL);
    if (res == NULL)            /* errno set from final connect() */
        err_sys ("tcp_connect error for %s, %s", host, serv);
    freeaddrinfo (ressave);
    return (sockfd);
}
```

Resolve names over multiple paths in parallel

Resolve protocol alternatives and security hints

Parallelise for Happy Eyeballs

Set up secure transport

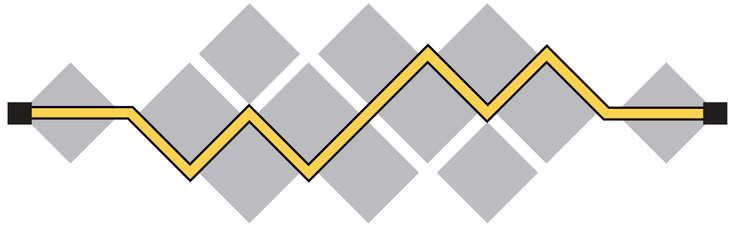
Pass transport protocols chosen

Stevens et at. – UNIX Network Programming Volume 1, Third Edition – The Sockets Networking API

How do we solve that?

IETF & The TAPS WG

What is the IETF?



I E T F®

"We reject kings, presidents and voting. We believe in rough consensus and running code".

David Clark

"Be conservative in what you send and liberal in what you accept".

Jon Postel

https://www.ietf.org/about/participate/tao/

IETF Areas

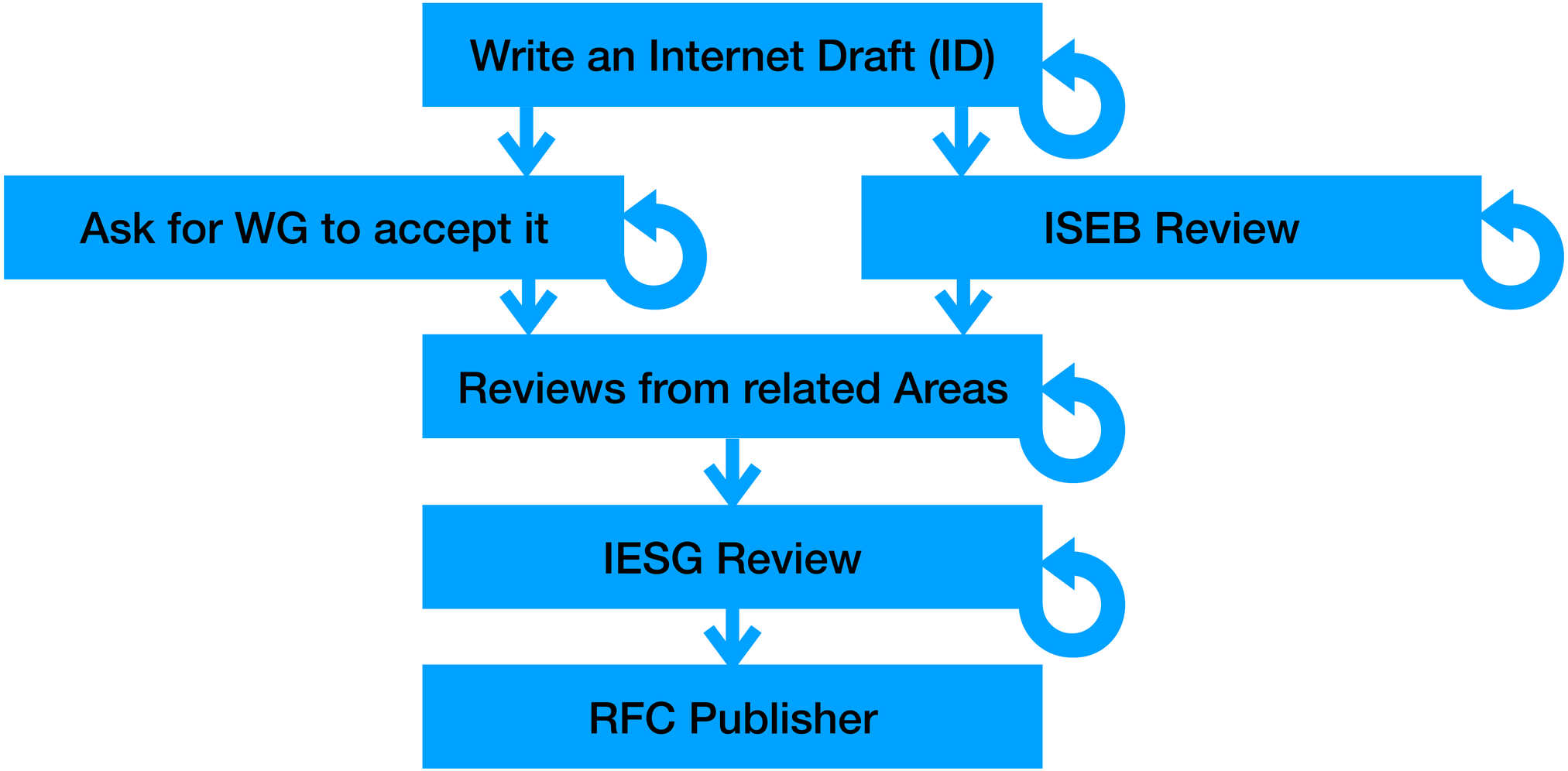
- Applications and Real-Time Area (art)
- General Area (gen)
- Internet Area (int)
- Operations and Management Area (ops)
- Routing Area (rtg)
- Security Area (sec)
- Transport Area (tsv)

TAPS WG

- Concerned with Transport Services
- Enable application developers to use protocols other than TCP and UDP
- Enable transport protocol evolution
- Describe an abstract API to use Transport Services

IETF only specifies abstract APIs
APIs for concrete languages are made by other bodies, e.g., POSIX for the UNIX C API

Publishing an RFC (roughly)



TAPS in a Nutshell

Event-Driven API

- Modern applications interact asynchronously
- Interactions with TAPS do not block.
  `initiate / listen / read / write / …`
  An event is sent when results are available.
- Additional events for message expiry, changed connection stated, and network conditions are generated

Flexibile Implementation

- Enabling transport protocol evolution by focusing on the transport services an application needs, not the protocol that usually provides it today.
- Flexible at connection establishment time, considering many different options and trying to select the most optimal combination, e.g., by using Happy Eyeballs.
- Flexibility after connection establishment to enable the use of connection migration, multi-path and multi-streaming.

Data Transfer Using Messages

- Structured interface for message based communication
- Support for message framing/deframing for massage based protocols over stream transports, e.g., HTTP over TCP
- Allows to control deadlines, reliability and other transport services on a per-massage basis
- Allows to assign messages to underlaying transport connections to utilise multi-streaming and pooled connections

Framers

- Framers allow to prepare messages for processing by the application.
- Main use case: enable message-based communication on stream transports.
- Simple protocol layers can also be implemented as framers.
- Nicely integrate with flow control and back-pressure.

Tansport Properties

- Means for configuring connection and message behaviour.
  - Selection Properties influence path- and protocol selection
  - Connection Properties influence per-connection behaviour
  - Message Properties influence per-message behaviour
- Dictionary-like Interface
- Well-defined name space for extensibility

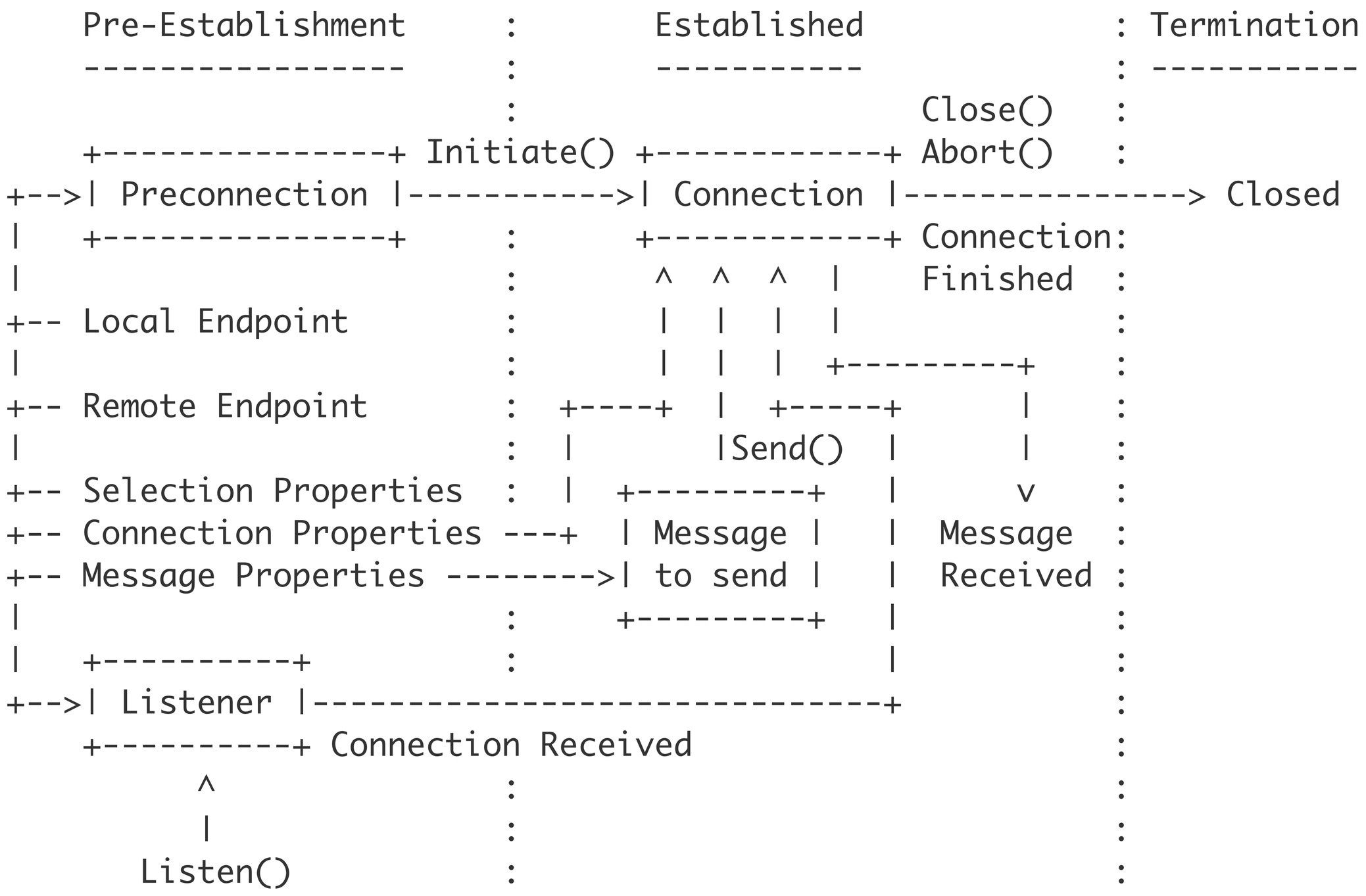
Application Interaction

Application:

I want a Connection to example.org, to service HTTPS
I need reliable transfer,
please optimize for low latency.

TAPS System:

Here’s your Connection object,
you can send and receive messages on it
and don’t have to care about protocols or paths


Pre-Establishment
Established
Termination
Initiate()
Close()
Abort()
Preconnection
Connection
Closed
Connection Finished
Local Endpoint
Remote Endpoint
Selection Properties
Connection Properties
Message Properties
Send()
Message to send
Message Received
Listener
Connection Received
Listen()

Abstract API Example

```
remoteSpecifier := NewRemoteEndpoint()
remoteSpecifier.WithHostname(“example.com")
remoteSpecifier.WithService("https")

transportProperties := NewTransportProperties(reliable-inorder-stream)
transportProperties.Add(conn-capacity-profile, low-latency)

securityParameters := NewSecurityParameters()
securityParameters.SetTrustVerificationCallback(myTrustCallback)

// Specifying a local endpoint is optional when using Initiate()
preconnection := NewPreconnection(None, remoteSpecifier,
                                  transportPreperties, securityParameters)

preconnection.AddFramer(http_framer)

connection := preconnection.Initiate()

connection -> Ready<>
```

Abstract API Example (cont.)

```
RequestMessageContext := NewMessageContext()
RequestMessageContext.add(msg-lifetime, 200ms)
RequestMessageContext.add(tcp.no-delay, true)

Connection.Send(RequestMessageData, MessageContext)

// Only receive complete messages
Connection.Receive()

Connection -> Received(ResponseMessageData, ResponseMessageContext)

Connection.Close()
```

python-asyncio-taps Example

```
import PyTAPS as taps

remote_endpoint = taps.RemoteEndpoint()
remote_endpoint.with_hostname("example.org")
remote_endpoint.with_service(“https”)

properties = taps.TransportProperties()

security = taps.SecurityParameters()
security.addTrustCA(args.trust_ca)
```

```
async def handle_ready(connection):
        print("Connection has been successfully established”)
        await connection.send_message(data)
        connection.on_received(handle_received)
        await connection.receive()

async def handle_received(self, data, context):
        print("Received data: " + str(data))
```

```
preconnection = taps.Preconnection(remote_endpoint=endpoint,
                                   local_endpoint=None,
                                   transport_properties=properties,
                                   security_parameters=security)

preconnection.on_ready(handle_ready)

loop = asyncio.get_event_loop()
loop.create_task(preconnection.initiate())
loop.run_forever()
```

TAPS Implementations

- python-asyncio-taps
  https://github.com/fg-inet/python-asyncio-taps
- Apple Network Framework
  https://developer.apple.com/documentation/network
- NEAT
  https://github.com/NEAT-project/neat
- Socket Intents
  https://github.com/fg-inet/socket-intents

TAPS Ressources

- Active IDs in the TAPS WG
  https://datatracker.ietf.org/group/taps/documents/
- Github Repository
  https://github.com/ietf-tapswg/api-drafts