NEAT – A New Evolutive API and Transport-Layer Architecture for the Internet

Karl-Johan Grinnemo1, Tom Jones2, Gorry Fairhurst2, David Ros3, Anna Brunstrom1, and Per Hurtig1

1Karlstad University, Karlstad, Sweden, Email: {karl-johan.grinnemo,anna.brunstrom,per.hurtig}@kau.se
2University of Aberdeen, Aberdeen, U.K., Email: {tom.gorry}@erg.abdn.ac.uk
3Simula Research Laboratory, Oslo, Norway, Email: dros@simula.no

Internet Transport is Ossified

- Middleboxes, e.g., NATs
- The Sockets API
- Obsolete IP options
- ...

Point Solutions

- Virtual network overlays – Hide the underlying network from the transport
- Application-layer transport enhancements (e.g., middleware)
- Fairly complex and limited scope
- Application-layer transport protocols (e.g., QUIC)
- Target a particular category of applications
- Sockets API extensions
- Not deployable and evolvable

New Transport

- Deployable – Independent of particular software and technologies
- Evolvable – Permit parts (e.g., protocols) to be added as needed
- Loose coupling of parts
- Flexible API – Higher abstraction level than the Sockets API
- Offers a transport service

Overview NEAT Architecture

- High-Level Transport System
  - Often uses low-level transport system
  - Essential for the creation, utilisation, and evolution of NEAT

Low-Level Transport System

Remarks:

- NEAT Flow Endpoint ≈ TCB
- NEAT Logic orchestrates the transport system
- NEAT Policy components administer application and network policies and comprise:
  - NEAT Policy Manager
  - Policy Information Base (PIB)
  - Characteristics Information Base (CIB)
- Connect to a name is the NEAT address resolver and provide the following features:
  - Asynchronous DNS lookup
  - Address monitoring
  - Multi-homing support
  - Private network marking
- Happy Eyeballs handles transport selection

NEAT API Framework

- Implements a callback-based API
- The base is an event loop
- Built on top of libuv

NEAT API Example

```c
// new neat flow
static neat_flow_operations ops;
static neat_flow_code on_connect(void *ctx, neat_flow_operations *opCB)
    
    // new neat flow
    if (neat_flow_operations(on_connect, ops) != NULL)
    
    // set callbacks
    ops.on_connect = on_connect;
    ops.on_message = on_message;
    ops.on_output = on_output;
    ops.on_destroy = on_destroy;
```

Ongoing and Future Work

- Implement libNEAT library
- Evaluate performance of libNEAT
  - E.g., Happy Eyeballs
- Transport Protocol Enhancements
  - E.g., multipath scheduling
- Transport System Extensions
  - E.g., transport selection