neat

# TAPS-related topics from the NEAT project

Naeem Khademi (NEAT project)

TAPS WG - IETF 97

Seoul- South Korea

16 November 2016

Horizon 2020
European Union funding
for Research & Innovation

# Introduction on NEAT

- NEAT project has been ongoing since March 2015

- NEAT library builds a TAPS-like prototype system
  - Open source, BSD Licensed (3 clause), implemented in C
  - Currently supports FreeBSD, Linux (Ubuntu), Mac OSX, and NetBSD
  - Callback-based, libuv-based
  - Still work-in-progress

- The NEAT API was first presented at IETF 95 (April'16, Buenos Aires)

# Some major updates since IETF 95

- Changes in the API properties
- Integration of NEAT Policy module (policies and profiles)
- Implemented QoS support
- Porting apps to use NEAT

# Application properties in NEAT

- **NEAT gives users a chance to control as much as they want, yet allow automatization**

- Key/value-based property system using JSON format
  - They can have different types and metadata attached to them, e.g. precedence
  - can set multiple/all properties with one API call

- Properties are given “precedence” -- e.g. 1=desired; 2=required
  1) Desired: try and fallback if unsuccessful
  2) Required: fail if unsuccessful

```
{
   "property_name": {
      value: "property_value",
           precedence: 1
      }
}
```

```
{
      "transport": [
            {
                  "value": "SCTP",
                  "precedence": 1
            },
            {
                  "value": "TCP",
                  "precedence": 1
            }
      ]
}
```

neat

# NEAT Policies and Profiles

- **NEAT provides a flexible way to define policies; also allows for creation of profiles depending on the networking scenario**

- **Policies:** based on NEAT properties with priorities among themselves, in JSON format - set by the user, system administrator or developer

- **Profiles:** matching property in the request is *replaced* with the associated profile properties

```
{
   "name": "Low latency",
   "match": {
      "low_latency": {
         "precedence": 1,
         "value": true
      }
   },
   "properties": {
      "interface_latency": {
         "precedence": 2,
         "value": [0,40]
      },
      "is_wired": {
         "precedence": 1,
         "value": true
      }
   }
}
```

An example profile

neat

# Porting apps to use NEAT

- **Built in NEAT:** many common network programming tasks like address resolution, buffer management, encryption, connection establishment and handling

- Address resolution and connection establishment with a *single* function call

```
neat_open(ctx, flow, "bsd10.fh-muenster.de", 80, NULL, 0);
```

- We ported Nghttp2 (a HTTP/2 implementation) web server/client and a few other smaller http/https-based clients to use NEAT
  - Interoperable with TCP
  - Can benefit from using SCTP

**20% reduction in code lines**

# QoS support in NEAT

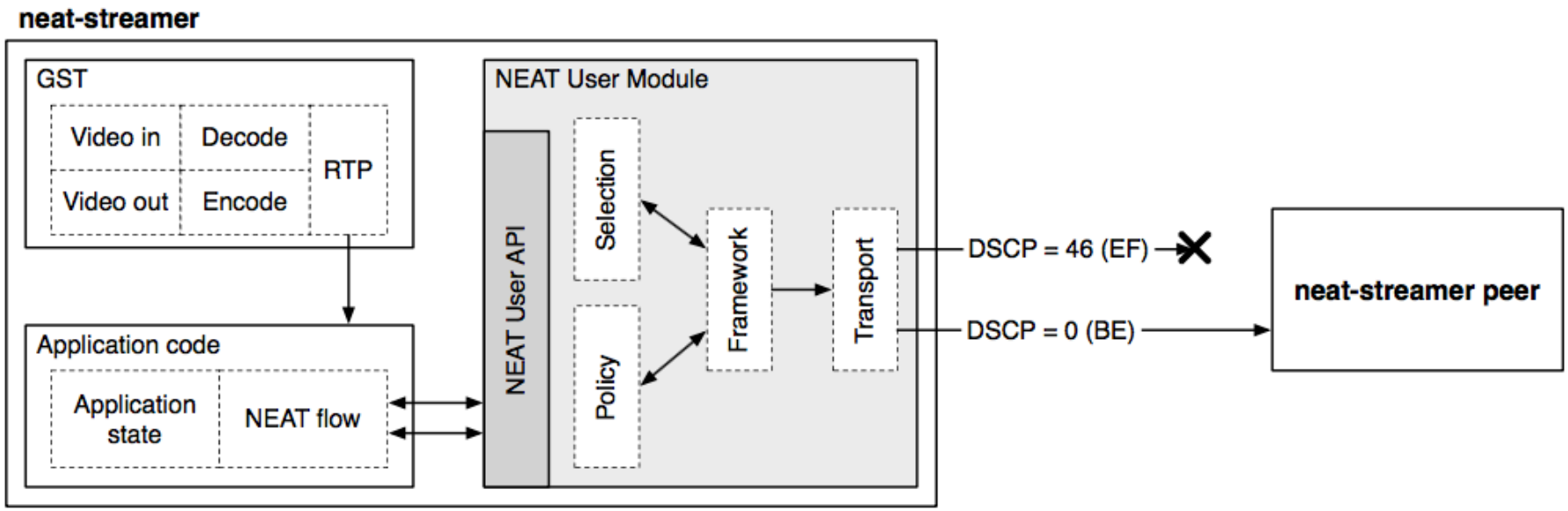
- **Network QoS:** often limited to controlled network environment due to lack of *high-level API*

- **Key challenge:** how to express service requirements, while still enabling policy to influence choice and providing flexibility when the network is unable to directly satisfy the requirements

- **With NEAT:** can use user requirements, policy, and dynamic info collected from other connections to choose appropriate DSCP value
  - Using table 1 in **draft-ietf-tsvwg-rtcweb-qos-18** as guidance

# QoS fallback using NEAT

- We developed **neat-streamer** based on Gstreamer (pipeline-based media library for audio/video)
  - Fall-back to DSCP=0 in case of black-holing
    ( similar to WebRTC - see **draft-ietf-rtcweb-transports-17** )



# More major updates

**draft-welzl-tcp-ccc**
Yesterday's ICCRG meeting

- A step towards full **multi-streaming** support:
  - API support for flow group (local) priorities (e.g. to leverage Coupled-CC)
  - Direct use of SCTP Multi-streaming (ongoing work on transparent use of it)

- Improvements in "transport protocol" **HE mechanism** and code
  - Including investigation of transport HE's cost **(presented in TAPS, IETF 96)**
  - Uses priorities among "candidate transport solutions" with a fixed delay

- **Datagram support** for the API (UDP, UDP-Lite) in addition to TCP, SCTP, SCTP/UDP

- **Server-side support** (listening on multiple protocols)

- **Security** (TLS over TCP; ongoing work on DTLS over SCTP|UDP)

- Lots of improvements, debugging and code optimizations

**NEAT EU project:** https://www.neat-project.org

**Github Repository:** https://github.com/NEAT-project/neat

**API documentation and tutorial:** http://neat.readthedocs.io/en/latest

*Comments, feedback, patches, test results, suggestions on target apps are welcome!*

# Q&A

neat