Transparent Flow Mapping for NEAT

Felix Weinrank, Michael Tüxen
Department of Electrical Engineering and Computer Science
Funded by EU H2020 NEAT project (Grant agreement no. 644334)

FH MÜNSTER
University of Applied Sciences

In a nutshell

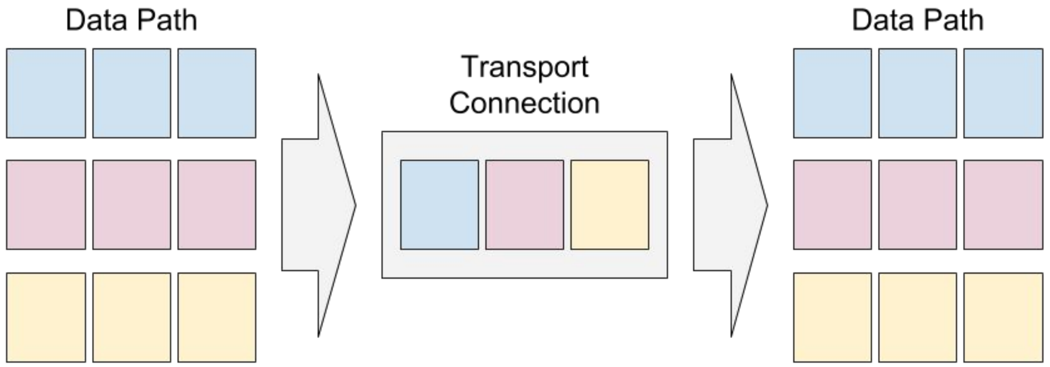
Automatic multiplexing + fallback

Transparent for the application

No additional coding effort

Multiplexing

- Bundling of several data paths to a single transport connection
- Key feature in widely used protocols
  - HTTP2 (TCP)
  - QUIC (UDP)
  - WebRTC Data Channel (SCTP)



Multiplexing - Pros and Cons

- Pros
  - Flow- and congestion-control mechanisms benefit from larger quantities of transferred data
  - Higher packet rates result in quicker loss detection
  - Shared congestion window is beneficial for new connections and connections with a low sending rate
  - Reduced amount of connections improves server capacities
- Cons
  - Additional coding effort
  - Fallback mechanism (optional)

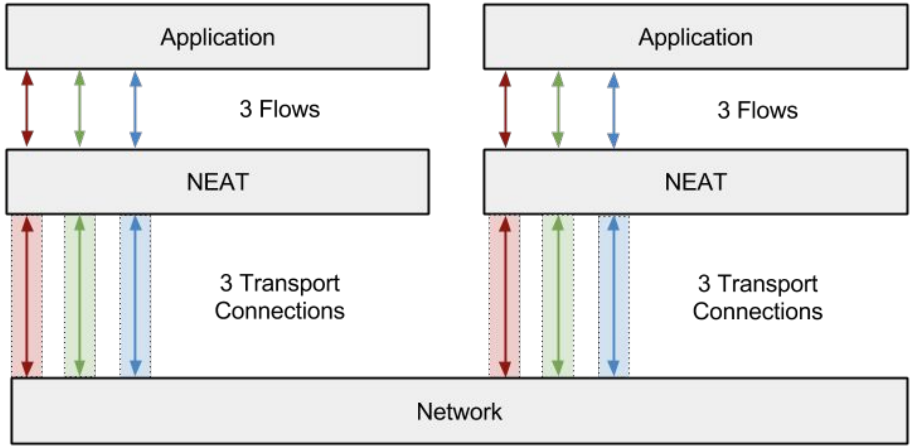
NEAT Library

- Userland library for network communication
- Non-blocking and callback-based concept
- Unified API for all network protocols
- Supports (MP)TCP, UDP, SCTP (Kernel + Userland)
- Runs on Linux, FreeBSD, NetBSD and macOS
- Based on libuv
- www.neat-project.org

neət

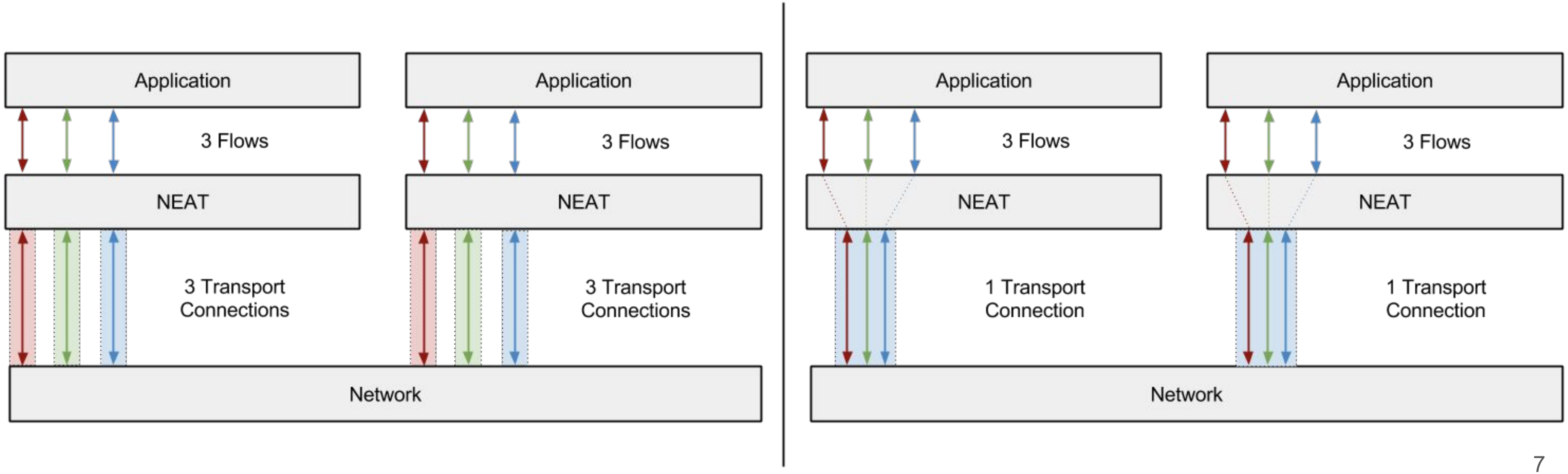
NEAT - Flow

- Bi-directional communication channel between two application endpoints
- Handles DNS resolution, buffer management, ...
- Can be grouped
- Unified API for all supported protocols
  - neat_open()
  - neat_write()
  - neat_read()
  - neat_close()
  - ...



Transparent Flow Mapping (TFM) - Concept

- Mapping multiple NEAT flows to a single transport connection while behaving like a 1:1 mapped flow



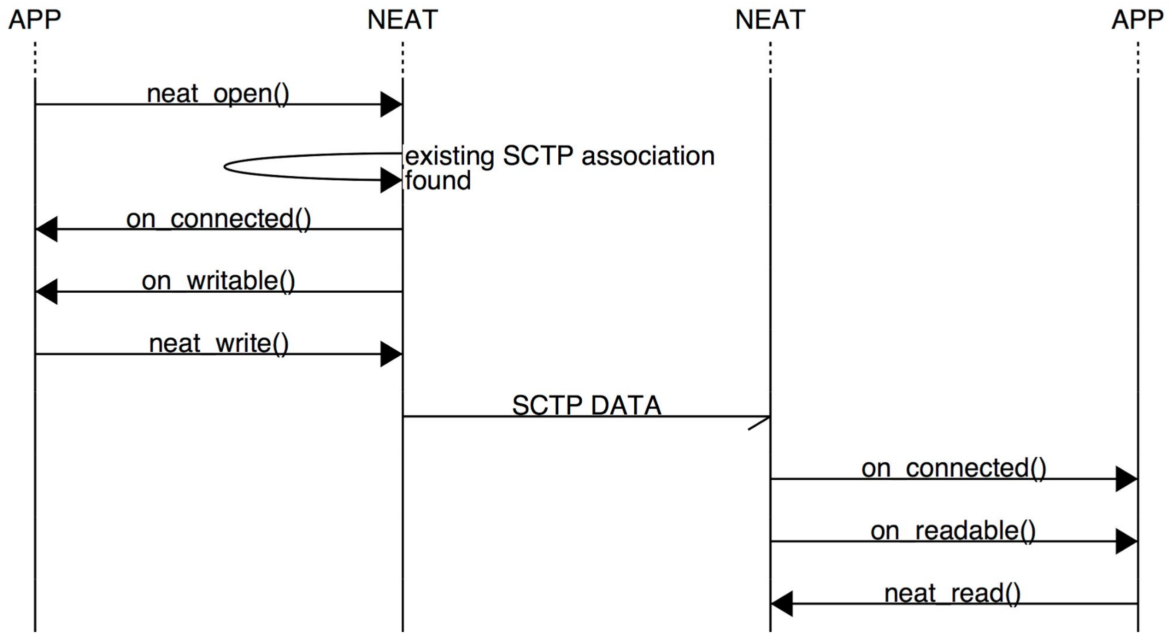
TFM - Requirements and Negotiation

- Both sides have to support
  - SCTP
  - SCTP - Stream Reconfiguration extension
  - SCTP - User Message Interleaving (IDATA) extension
- Support for TFM is negotiated via SCTP’s adaptation layer indication value
  - Carried via INIT / INIT-ACK chunk
  - TFM for NEAT specific value
  - If set by both sides → TFM support negotiated

TFM - Flow creation

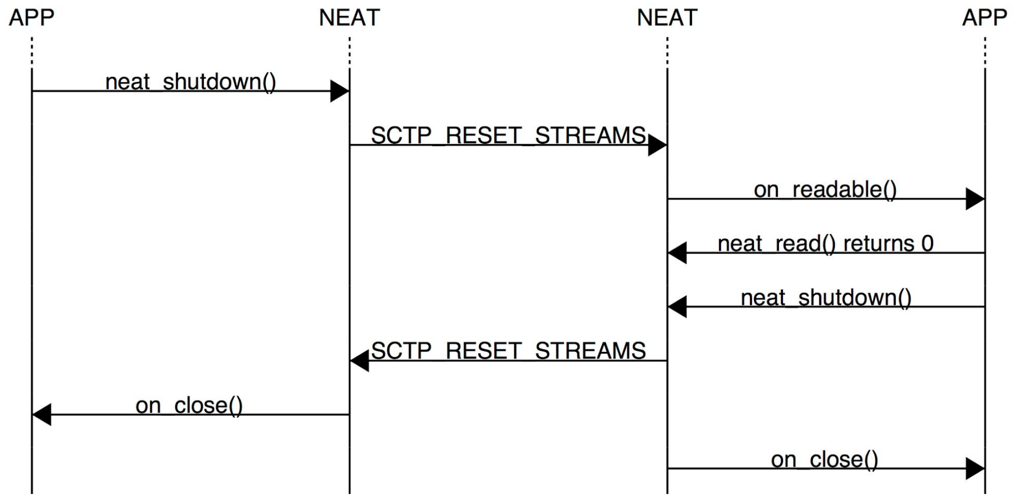
- Transparent mapping of a new flow requires an existing flow with
  - Same destination IP / DNS-Name
  - Same port number
  - SCTP connection
  - Unused SCTP stream
  - TFM support
- New flow is instantly mapped to existing transport connection
  - Zero RTT connection setup

TFM - Data Transmission



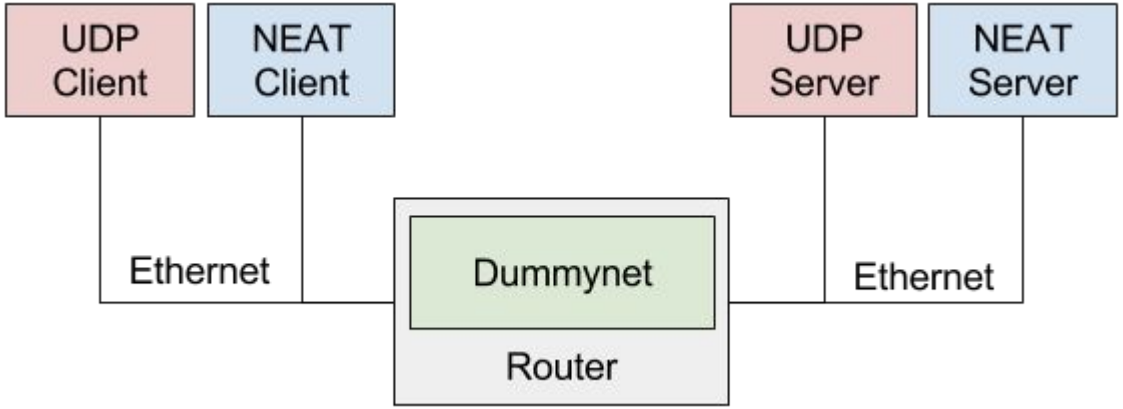
TFM - Flow Teardown

- Using SCTP's Stream Reset extension for closing procedure

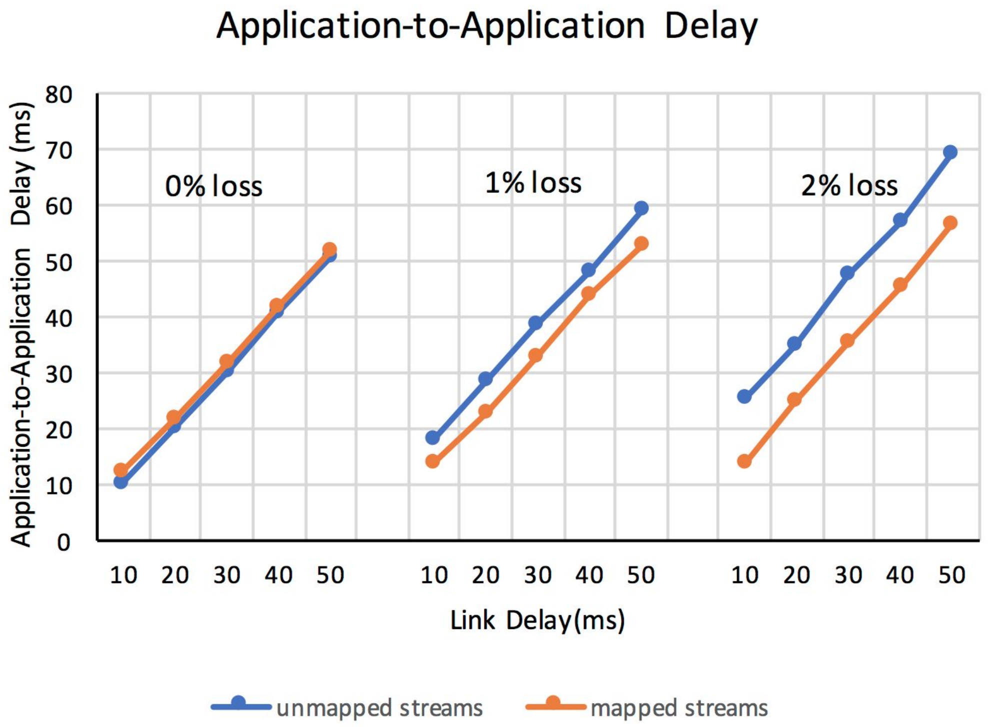


TFM - Measurement Scenario

- NEAT application using two flows, same target, low sending rate
- Comparing “1:1 mapping” vs “transparent flow mapping”
- Focus: Application-to-Application delay
- UDP background traffic



TFM - Measurement Results



TFM - Alternative Transport Protocols

- Our implementation uses SCTP with extensions
- Transparent approach allows usage of alternative protocols
  - Easy integration into Happy-Eyeballs mechanism
- Interesting Candidate: Google’s QUIC
  - Quick UDP Internet Connections
  - Multiplexing concept
  - Built-in encryption
  - Zero-RTT connection setup
  - Not standardized (yet)

Conclusion and Outlook

- Multiplexing without additional effort for the developer
- Automatic negotiation and integrated fallback solution
- Beneficial for multiple flows with a low sending rate
  - Faster loss detection
  - Congestion-Window reusage
  - Less server load
- Approach allows seamless integration of alternative protocols like QUIC

Questions? :)