# A Practical Introduction to NEAT at Hainan University

Thomas Dreibholz ( 托马斯博士 )
dreibh@simula.no



Horizon 2020
European Union funding
for Research & Innovation

# Contents

# Disclaimer

- This is work in progress
- The API can change anytime
- The code has not been tested substantially
  (Report bugs at https://github.com/NEAT-project/neat/issues)
- Comments welcome

- “This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 644334 (NEAT). The views expressed are solely those of the author(s).”

# Motivation

- Many different applications
- New Transport Layer protocols/extensions
  - Stream Control Transmission Protocol (SCTP)
  - Concurrent Multipath Transfer for SCTP (CMT-SCTP)
  - Datagram Congestion Control Protocol (DCCP)
  - Multi-Path TCP (MPTCP)
- Problem:
  - “SCTP is not available for Windows”
  - “Firewall does not support SCTP → no NAT with SCTP → no SCTP”
  - “CMT-SCTP is only available for FreeBSD => no support in Linux”
  - => Application developer: “I just use regular TCP, it works everywhere!”

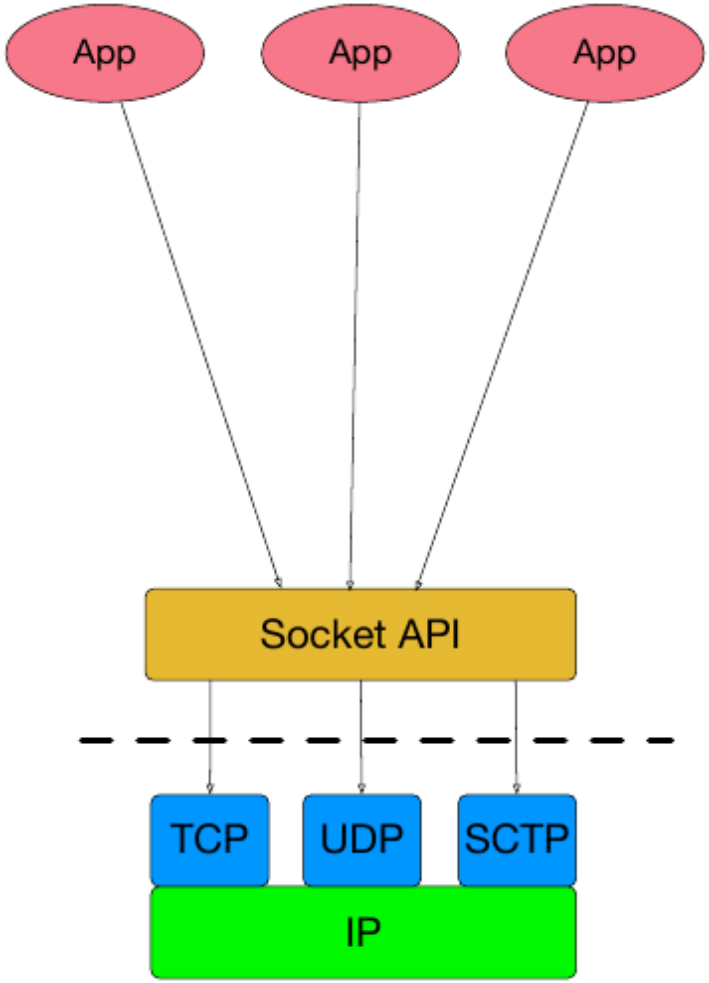
# The NEAT Project

- A New, Evolutive API and Transport-Layer Architecture for the Internet
- Partners:
  - Simula Research Laboratory (SRL)
  - University of Oslo (UiO)
  - Karlstads Universitet (KAU)
  - Münster University of Applied Sciences
  - University of Aberdeen
  - Celerway, EMC, Mozilla, Cisco (companies)
- https://www.neat-project.org
- https://github.com/NEAT-project/neat

# Goal

- A middleware between application and Transport Layer
- An application specifies its requirements:
  - Preferred Transport Layer protocols, maximum acceptable delay, …
- NEAT: According to the requirements
  - Selects Transport Layer protocol(s),
  - Configures the Transport Layer protocols.
- Transport Layer
  - Kernel remains unchanged (makes deployment easy)
  - Possibility to use user-space implementations
    (currently: usrsctp → state-of-the-art SCTP, including CMT-SCTP!)

# How Applications work Today



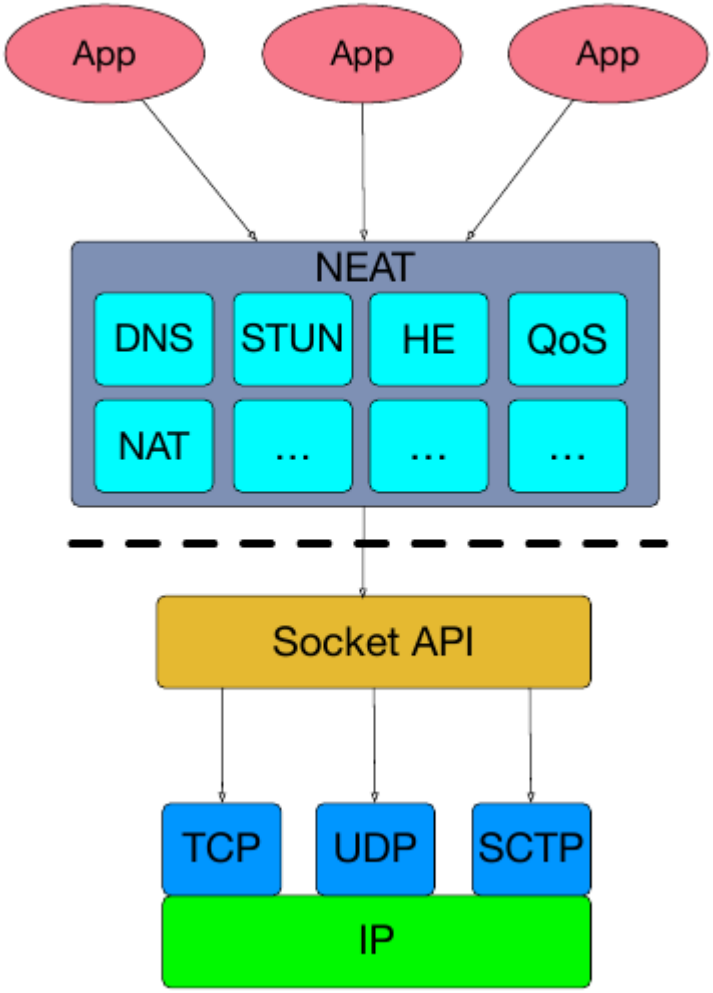

Note:
"BSD Sockets API" may significantly vary among different operating systems, e.g.:

- Different socket options
- Different features

# How Applications work with NEAT

<u>With NEAT:</u>

- Applications use common NEAT interface
  - Describe requirements/configurations by properties and options
- NEAT uses APIs of the available Transport Layer protocols (TCP, SCTP, UDP, DCCP, MPTCP, …)
- NEAT helps with additional features (DNS lookup, QoS settings, NAT, TLS, …)



# The Implementation

- Implementation
  - Implemented in C, Open Source under BSD license
  - Portable (currently supports FreeBSD, Linux, Mac OS X, and NetBSD)
- Source
  - Git repository: https://github.com/NEAT-project/neat
  - Branch: dreibh/neat-socketapi
  - Ready-to-use packages:
    - Ubuntu: https://launchpad.net/~dreibh/+archive/ubuntu/ppa/+packages
    - Fedora: https://packages.nntb.no/nornet-applications/fedora/
    - Note: use system's package management tool to add repository!

# The APIs

- NEAT Callback API
  - Asynchronous (non-blocking) functions
  - Application registers callbacks for events
    (like data/message received, new incoming connection, etc.)
  - Practical for asynchronous applications
  - Quite different from BSD Sockets-like API
- NEAT Sockets API
  - An API like the BSD Sockets API
  - Blocking mode (e.g. to wait for data), non-blocking mode (e.g. poll())
  - Easy porting of existing applications

# The NEAT Sockets API

- Documentation (as Internet Draft):
  - https://tools.ietf.org/id/draft-dreibholz-taps-neat-socketapi-00.txt
  - Work in progress!

- In the following:
  - Server pseudo-code example
  - Client pseudo-code example
  - How to get running code examples?

# Server Example: Blocking Mode (1)

```
uint16_t port=8888;
const char* properties = "{\
   \"transport\": [\
      { \"value\": \"SCTP\", \"precedence\": 1 },\
      { \"value\": \"TCP\",  \"precedence\": 0 } ] }";

int sd = nsa_socket(0, 0, 0, properties);
nsa_bindn(sd, port, NULL, 0, 0);
nsa_listen(sd, 10);
while(1) {
   int newSD = nsa_accept(sd, NULL, 0);
   Start new thread to handle requests from newSD;
}
nsa_close(sd);
nsa_cleanup();
```

# Server Example: Blocking Mode (2)

```
void serviceThread(int newSD) {
   ssize_t readBytes = nsa_read(newSD, ...);
   while(readBytes > 0) {
      // do something with the received data
      ...
      // send something ...
      nsa_write(newSD, ...);
      ...
      readBytes = nsa_read(newSD, ...);
   }
   nsa_close(newSD);
}
```

# Client Example

```
const char* properties = "{\
   \"transport\": [\
      { \"value\": \"SCTP\", \"precedence\": 1 },\
      { \"value\": \"TCP\",  \"precedence\": 0 } ] }";

int sd = nsa_socket(0,0,0,properties);
nsa_connectn(sd, “myserver.nntb.no”, 8888, NULL, NULL, 0);
while(1) {
      // send something ...
      nsa_write(newSD, ...);
      // receive something
      nsa_read(newSD, ...);
}
nsa_close(sd);
nsa_cleanup();
```

# How to Get Running Code Examples?

- Install NEAT libraries in Ubuntu (16.04 or later):
  - sudo apt-add-repository -s -y ppa:dreibh/ppa
  - sudo apt update
  - sudo apt install libneat-socketapi-dev
- Example HTTP servers and client:
  - git clone https://bitbucket.org/dreibh/neat-examples
  - cd neat-examples
  - ./autogen.sh
  - Read the README in the “examples” directory!

# Using NEAT in NorNet Core

- Install NEAT libraries in a NorNet Core sliver:
  - sudo dnf install libneat
  - /etc/yum.repos.d/nornet.repo contains the repository (it is already configured, i.e. nothing to do here!):
    - `[NorNet-Applications]`
    - `name=NorNet Applications`
    - `baseurl=http://packages.nntb.no/nornet-applications/fedora/$releasever/$basearch`
    - `enabled=1`
    - `gpgcheck=1`
    - `gpgkey=file:///etc/pki/rpm-gpg/nornet.key`
- Build and run the examples


https://www.nntb.no

# Conclusion

- Summary
  - NEAT provides a smart middleware layer between application and transport protocols
  - Easy-to-use NEAT Sockets API
- Ongoing and Future Work
  - Notifications (similar to SCTP's notifications handling API)
  - Transport Layer Security support in NEAT
  - Adapt applications (like NetPerfMeter, RSPLIB, …) to use NEAT

# Literature


- **Dreibholz, T.:** “NEAT Sockets API” (TXT, 43 KiB), IETF, Individual Submission, Internet Draft draft-dreibholz-taps-neat-socketapi-00, April 11, 2017.
- **Khademi, N.; Bozakov, Z.; Brunström, A.; Dale, Ø.; Damjanović, D.; Evensen, K. R.; Fairhurst, G.; Grinnemo, K.; Jones, T.; Mangiante, S.; Petlund, A.; Ros, D.; Stenberg, D.; Tüxen, M.; Weinrank, F.; Welzl, M.:** “NEAT – Core Transport System, with both Low-level and High-level Components” (PDF, 1643 KiB), no. D2.2, March 14, 2017.
- **Khademi, N.; Bozakov, Z.; Brunström, A.; Damjanović, D.; Evensen, K. R.; Fairhurst, G.; Grinnemo, K.; Jones, T.; Mangiante, S.; Papastergiou, G.; Ros, D.; Tüxen, M.; Welzl, M.:** “NEAT – First Version of Low-Level Core Transport System” (PDF, 287 KiB), no. D2.1, March 1, 2016.
- **Welzl, M.; Brunström, A.; Damjanović, D.; Evensen, K. R.; Eckert, T.; Fairhurst, G.; Khademi, N.; Mangiante, S.; Petlund, A.; Ros, D.; Tüxen, M.:** “NEAT – First Version of Services and APIs” (PDF, 752 KiB), no. D1.2, March 1, 2016.
- **Fairhurst, G.; Jones, T.; Bozakov, Z.; Brunström, A.; Damjanović, D.; Eckert, K. R. E. T.; Grinnemo, K.; Hansen, A. F.; Khademi, N.; Mangiante, S.; McManus, P.; Papastergiou, G.; Ros, D.; Tüxen, M.; Vyncke, E.; Welzl, M.:** “NEAT Architecture” (PDF, 568 KiB), no. D1.1, December 1, 2015.
- **Stewart, R. R.; Tüxen, M.; Poon, K.; Lei, P.; Yasevich, V.:** “Sockets API Extensions for the Stream Control Transmission Protocol (SCTP)” (TXT, 232 KiB), IETF, Informational RFC 6458, DOI 10.17487/RFC6458, ISSN 2070-1721, December 2011.
- **Stevens, W. R.; Fenner, B.; Rudoff, A. M.:** “Unix Network Programming”, Addison-Wesley Professional, Addison-Wesley Professional, ISBN 0-131-41155-1, 2003.


Thank you for your attention!
Any questions?
Thomas Dreibholz（托马斯博士）
dreibh@simula.no