A NEAT Way to Browse the Web

Felix Weinrank
Münster University of Applied Sciences

Karl-Johan Grinnemo
Karlstad University

Zdravko Bozakov
Dell EMC Research Europe

Anna Brunstrom
Karlstad University

Thomas Dreibholz
Simula Research Laboratory

Gorry Fairhurst
University of Aberdeen

Per Hurtig
Karlstad University

Naeem Khademi
University of Oslo

Michael Tüxen
Münster University of Applied Sciences

ABSTRACT

There is a growing concern that the Internet transport layer has become ossified in the face of emerging novel applications, and that further evolution has become very difficult. The NEAT system is a novel and evolvable transport system that decouples applications from the underlying transport layer and network services. In so doing, it facilitates dynamic transport selection. This demo shows how the NEAT system is able to dynamically select the most appropriate transport solution for the Mozilla Firefox web browser.

KEYWORDS

NEAT, ossification, transport selection, transport service, SCTP, TCP

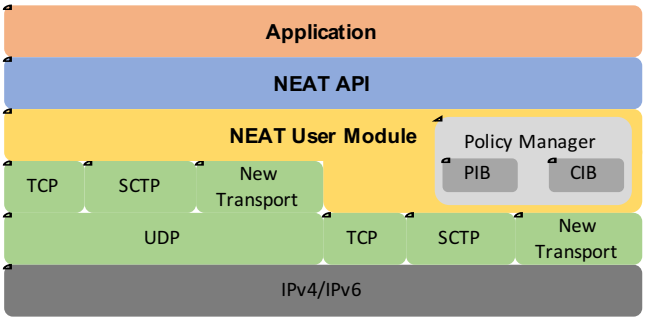

Figure 1: The architecture of the NEAT system.

1 INTRODUCTION

The Internet is often seen as having a common network layer and two widely deployed transport protocols, TCP [7] and UDP [6], with other transports, such as SCTP [9], struggling to find broad deployment. In line with ongoing standardization efforts within the Transport Services (TAPS) working group at the IETF [10], the NEAT system [4, 8] challenges transport-layer ossification by providing an API that is oblivious to specific protocols and instead focuses on requested transport services. To obtain a transport service, applications provide NEAT information about the remote peer and their service requirements. Based on this information, pre-specified policies, and measured network conditions, NEAT establishes and configures appropriate network connections.

In this demo, we demonstrate how a web browser, the Mozilla Firefox web browser, leverages the dynamic transport selection of NEAT, and in so doing is able to dynamically select the transport solution that gives the shortest latency. Particularly, we show how NEAT assists Firefox to use the SCTP transport protocol and its Concurrent Multipath Transfer (CMT) Extension [3] when SCTP is available and works along the network paths between the endpoints.

2 THE NEAT SYSTEM ARCHITECTURE

Figure 1 illustrates the NEAT system. Applications access NEAT via an event-driven API. The API interfaces the NEAT User Module, which constitutes the core of the system. The NEAT User Module is designed to be portable across different operating systems and network stacks. One of its primary responsibilities is to select the most appropriate transport solution for a requested transport service. At the core of this selection process is the Policy Manager (PM).

The PM has a repository of collected policies, the Policy Information Base (PIB), where each policy consists of a set of rules linking a set of matching requirements to a set of preferred or mandatory transport characteristics. To implement the policies, the PM may access a repository storing dynamic information collected about available interfaces and the paths towards destination endpoints, the Characteristics Information Base (CIB). On the basis of its policies, the PM creates a list of candidate transport solutions, sorted in order of appropriateness/priority, to use for a requested transport service.

3 DYNAMIC TRANSPORT SELECTION IN NEAT

The list of candidate transport solutions created by the PM is supplied to the Happy Eyeballs building block of the NEAT User Module, whose core functions are described in an IETF Internet Draft [2].

Happy Eyeballs traverses the candidate list, and makes asynchronous connection attempts for each candidate transport solution (Algorithm 1). A lower-priority transport solution is delayed with respect to a higher-priority solution. The length of the delay depends on the priority. In the callback routine that is invoked when a connection attempt returns, the outcome of the connection attempt (success or failure) is cached by the PM.

ANRW'17, Prague, Czech Republic
2017. 978-x-xxxx-xxxx-x/YY/MM…$15.00
DOI: 10.1145/3106328.3106335

Algorithm 1 NEAT Happy Eyeballs Algorithm

```
procedure HappyEyeballs( in listOfCandidates : list of transport solutions)
Require: listOfCandidates sorted in priority order and len(listOfCandidates) > 0
currentCandidate ← listOfCandidates.first()
repeat
  if getPriority(currentCandidate) > 0 then
    delta ← convertToTimeInterval(getPriority(currentCandidate))
    scheduleAt( now() + delta,
      doAsynchConnectionAttempt(currentCandidate, connectionCallback))
  else
    doAsynchConnectionAttempt(currentCandidate, connectionCallback)
  end if
  currentCandidate ← listOfCandidates.nextCandidate(currentCandidate)
until currentCandidate = endOfList(listOfCandidates)
end procedure

procedure ConnectionCallback( in candidate : transport solution,
  out connection : transport connection)
if connection ≠ NONE then
  policyManager.cacheResultConnectionAttempt(candidate, SUCCESS)
else
  policyManager.cacheResultConnectionAttempt(candidate, FAILURE)
end if
end procedure
```

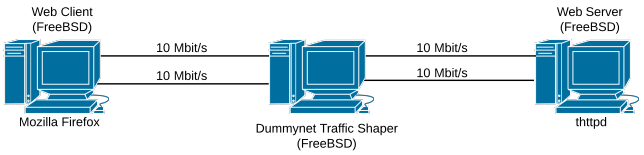

Figure 2: Demo of Mozilla Firefox over NEAT.

To avoid wasting networking resources by routinely making simultaneous connection attempts, the Happy Eyeballs building block instructs the PM to cache the outcome of previous connection attempts. The caching lifetime is part of the system configuration of NEAT. Cached connection attempts are valid for a pre-set time after which they become invalid and have to be repeated. Our experimental work [5] suggests a significant reduction in terms of CPU load with caching. We observed that the CPU load decreases linearly with increasing cache hit-rate, and resulted in a more than 40% reduction of CPU load for unencrypted traffic and almost a 20% reduction for encrypted traffic. In terms of memory, our work reported in [5] concluded that Happy Eyeballs only has a marginal impact on kernel memory usage.

4 FAST FILE DOWNLOADS IN MOZILLA FIREFOX OVER NEAT

In our demo, we show how the NEAT system enables the Mozilla Firefox web browser to choose the transport solution that offers the fastest file download times. Figure 2 depicts the demo setup. The setup comprises a client machine running a version of the Mozilla Firefox web browser modified to work over the NEAT system; a server machine running the thttpd [11] web server, a web server that supports both TCP and SCTP; and, in between these two machines, a Dummynet [1] traffic shaper. The client-side NEAT system employs a policy that for low-latency traffic, such as web traffic, gives priority to SCTP over TCP. All machines employ FreeBSD and thus support both TCP and SCTP. The Dummynet traffic shaper is operated through a web application, Happy BlueBox, that makes it easy to enable and disable SCTP traffic through the shaper. As follows from Figure 2, there are two 10-Mbit network paths between the client and server machines, in practice this mean that TCP is able to sustain a throughput of around 9 Mbit/s over each of the network paths.

Initially SCTP traffic is disabled, and file download takes place over TCP and a single network path. As a result, the throughput peaks at about 9 Mbit/s, and we experience noticeable download delays with files of size 32 Mbytes and larger. Next, SCTP is enabled; SCTP, via its CMT extension, sets up a dual-path association between the client and server and in so doing doubles the available bandwidth to 18 Mbit/s, and more or less halves the file download times for larger files.

5 CONCLUSIONS

This demo shows how the NEAT system is able to shorten the file download times of a web browser —the Mozilla Firefox web browser— by dynamically selecting the most appropriate transport solution. The demo illustrates only a subset of the NEAT capabilities, with focus on the Happy Eyeballs mechanism. For example, it does not show how information from CIB sources may modify candidate transport solutions, and how NEAT may interact with SDN controllers to configure transport parameters, or detect traffic flows that need particular treatment, e.g., 'elephant' flows.

ACKNOWLEDGMENTS

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).

REFERENCES

[1] Marta Carbone and Luigi Rizzo. 2010. Dummynet Revisited. *SIGCOMM Comput. Commun. Rev.* 40, 2 (April 2010), 12–20. DOI: http://dx.doi.org/10.1145/1764873.1764876

[2] Karl-Johan Grinnemo, Anna Brunstrom, Per Hurtig, Naeem Khademi, and Zdravko Bozakov. 2017. *Happy Eyeballs for Transport Selection.* Internet-Draft draft-grinnemo-taps-he-02. Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/draft-grinnemo-taps-he-02 Work in Progress.

[3] J. R. Iyengar, P. D. Amer, and R. Stewart. 2006. Concurrent Multipath Transfer Using SCTP Multihoming Over Independent End-to-End Paths. *IEEE/ACM Transactions on Networking* 14, 5 (Oct 2006), 951–964.

[4] N. Khademi, D. Ros, M. Welzl, Z. Bozakov, A. Brunstrom, G. Fairhurst, K. J. Grinnemo, D. Hayes, P. Hurtig, T. Jones, S. Mangiante, M. Tuxen, and F. Weinrank. 2017. NEAT: A Platform- and Protocol-Independent Internet Transport API. *IEEE Communications Magazine* 55, 6 (2017), 46–54.

[5] G. Papastergiou, K.-J. Grinnemo, A. Brunstrom, D. Ros, M. Tüxen, N. Khademi, and P. Hurtig. 2016. On the Cost of Using Happy Eyeballs for Transport Protocol Selection. In *Proceedings of the 2016 Applied Networking Research Workshop (ANRW)*. Berlin, 45–51.

[6] J. Postel. 1980. User Datagram Protocol. RFC 768 (INTERNET STANDARD). (Aug. 1980). http://www.ietf.org/rfc/rfc768.txt

[7] J. Postel. 1981. Transmission Control Protocol. RFC 793 (INTERNET STANDARD). (Sept. 1981). http://www.ietf.org/rfc/rfc793.txt Updated by RFCs 1122, 3168, 6093, 6528.

[8] NEAT Project. 2017. NEAT GitHub Repository. (2017). https://github.com/NEAT-project/neat/ Accessed on June 22, 2017.

[9] R. Stewart. 2007. Stream Control Transmission Protocol. RFC 4960 (Proposed Standard). (Sept. 2007). http://www.ietf.org/rfc/rfc4960.txt Updated by RFCs 6096, 6335, 7053.

[10] TAPS. 2015. IETF Transport Services (TAPS) Working Group Charter. (2015). https://datatracker.ietf.org/doc/charter-ietf-taps/

[11] Michael Tuexen. 2017. thttpd with SCTP Support. (2017). https://github.com/nplab/thttpd