# A NEAT framework for application-awareness in SDN environments

Ricardo Santos*, Zdravko Bozakov†, Simone Mangiante†, Anna Brunstrom* and Andreas Kassler*
*Karlstad University, Karlstad, Sweden
{name.surname}@kau.se
†Dell EMC Research Europe, Ovens, Ireland
{name.surname}@dell.com

***Abstract*—Software-Defined Networking (SDN) has led to a paradigm shift in the way how networks are managed and operated. In SDN environments the data plane forwarding rules are managed by logically centralized controllers operating on global view of the network. Today, SDN controllers typically posses little insight about the requirements of the applications executed on the end-hosts. Consequently, they rely on heuristics to implement traffic engineering or QoS support. In this work, we propose a framework for application-awareness in SDN environments where the end-hosts provide a generic interface for the SDN controllers to interact with. As a result, SDN controllers may enhance the end-host's view of the attached network and deploy policies into the edge of the network. Further, controllers may obtain information about the specific requirements of the deployed applications. Our demonstration extends the OpenDaylight SDN controller to enable it to interact with end-hosts running a novel networking stack called NEAT. We demonstrate a scenario in which the controller distributes policies and path information to manage bulk and low-latency flows.**

## I. Introduction

The Software-Defined Networking (SDN) paradigm promises to facilitate the management and operation of datacenter networks by enabling an automated, centralized control and optimization of pooled network resources. To achieve this goal, network controllers strive to maintain a rich and up-to-date global view of the network topology and the resources available therein. In order to efficiently map network traffic to the physical network substrate, controllers require information about the properties of flows traversing the network. Flow level information may be extracted by monitoring forwarding devices and used to obtain valuable metrics for traffic engineering. However, such information often does not offer the SDN controller a full picture of the myriad of network applications deployed at the edges and their respective requirements. For example, one may consider the wide range of network applications which tunnel traffic over TCP on port 80. In such cases, inferring whether a network flow is associated with a conferencing application with low latency requirements or a low bandwidth instant messaging application is a challenging task.

Application-aware networking [1], [2] refers to the ability of an intelligent network to consider the requirements of applications connecting to it and, as a result, optimize the performance of the individual applications as well as that of the overall network. To achieve application awareness, end-host applications may interact with a network controller to express their specific resource demands or obtain feedback about the current network conditions. Several recent proposals utilize local agents in end-hosts dedicated to collect metrics from applications and characterize them (e.g., [3]). Further, controllers may utilize their global view of the network to configure the behavior of network connections initiated by end-host applications and tune transport protocol parameters (e.g. [4]). The use-case specific approaches in these and other related works illustrate a clear demand for mechanisms that integrate end-host applications into SDN environments.

Building on previous successes, in this demo we introduce a generic, overarching framework for end-host SDN integration, integrated into the NEAT transport architecture [5]. Our framework provides structured interfaces for communication between applications and external controllers. The framework includes an expressive policy system which is able to fulfill a large range of use-case requirements, without requiring a re-engineering of the involved applications. We introduce our framework in the next section, followed by a description of the demo in Section III.

## II. End-host SDN integration with NEAT

The NEAT transport architecture [5] and accompanying software stack [6] is designed to offer a flexible and evolvable transport system. Applications interface the NEAT System through an enhanced API that effectively decouples them from the operation of the transport protocols and the network features being used. This allows the best transport option to be configured at run time based on application requirements, current network conditions, and the supported transport services on the path. Applications may supply the NEAT System with the requirements desired for each connection as well as optional hints about the type of traffic that will be transferred.

A key component of the NEAT framework is the Policy Manager (PM), which is responsible for matching application requirements with system policies as well as available information about system and network state, stored in a Policy Information Base (PIB) and a Common Information Base (CIB), respectively. For each application request, NEAT configures and establishes the most suitable transport connection, based

ISBN 978-3-901882-94-4 © 2017 IFIP

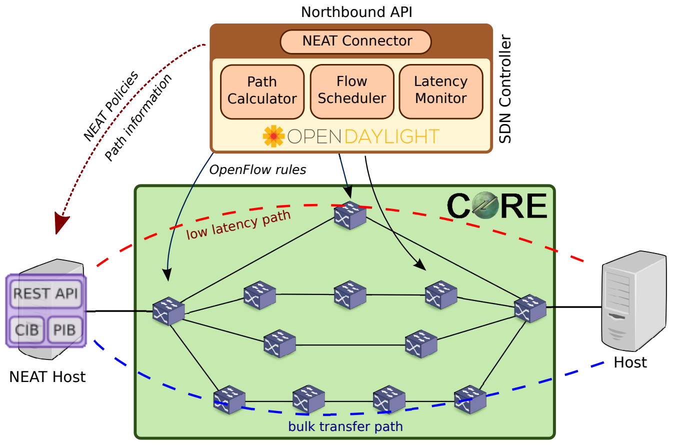

Fig. 1. Demonstration architecture

on a ranked list of feasible connection candidates generated by the PM. As a PM runs on each NEAT-enabled end-host, it provides an ideal hook, enabling SDN controllers to influence the parameters of any connection instantiated through NEAT. To this end, NEAT exposes a REST API through which external entities can query and modify the contents of the PIB and CIB repositories. As a result, logically centralized controllers may supply end-hosts with information about end-to-end, network-wide path characteristics such as latency, available bandwidth or loss rates. This information is automatically utilized by the PM to determine connection candidates which best match specific application requirements. In addition, the PM may mandate which network paths should be used for certain traffic classes and influence the parameters of transport connection by deploying suitable policies at specific end-hosts.

## III. Demonstration

This demonstration showcases the interaction between NEAT-enabled end-hosts and an SDN controller using the NEAT framework. The scenarios illustrate the management and handling of flows with different requirements. The NEAT-enabled end-hosts are interconnected through an emulated network using the CORE emulator [7] (Fig. 1). Inside CORE, we use nodes running Open vSwitch that are controlled by the OpenDaylight (ODL) controller using the OpenFlow protocol. We extended ODL with a module that implements a northbound interface for communicating with NEAT-enabled end-hosts. In addition, the controller is responsible for installing flow rules in the managed switches.

In the demo, we show how an SDN controller may optimize the placement of bulk transfers and latency-sensitive flows for a given network topology. The controller uses NEAT policies to instruct applications to tag transfers as elephant flows using a dedicated DSCP pattern if the flow size exceeds a predefined threshold. In addition, the controller implements a path probing approach by creating monitoring packets to determine latencies for different network paths [8]. The controller then provides end-to-end latency estimates to the end-hosts through the host's CIB repository. As client applications, we implemented a simple NEAT file transfer client that runs on a host with the NEAT framework enabled, and a corresponding server (not necessarily NEAT-enabled). For each transfer, the application on the NEAT-enabled host indicates the amount of data to be transferred, through the NEAT API. In addition, we implemented a NEAT-enabled real-time monitoring agent, which transmits periodic sensor readings to another host. The agent also uses the NEAT API to indicate its low-latency requirement.

Whenever a new file transfer is initiated, the flow size may trigger the elephant flow policy installed by the controller. The NEAT stack thus marks the packets with the DSCP value, which is used by the controller to map the large flows to high capacity paths. Packets without DSCP markings are instead forwarded along a pre-calculated path. The controller uses its Path Calculator and Flow Scheduler modules to install suitable forwarding rules on the switches between the source and destination nodes. Similarly, whenever the low latency agent creates a new connection, a system policy translates the low-latency requirement to a predefined value of end-to-end latency. The NEAT stack automatically selects the most suitable network interface based on the latency information pushed to the host's CIB through the controller. The effects of the installed policies and supplied path metrics on the network flows can be monitored in real time through CORE's GUI. Also, we will demonstrate the impact of different network topologies and different traffic specifications.

## Acknowledgement

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] Z. A. Qazi, J. Lee, T. Jin, G. Bellala, M. Arndt, and G. Noubir, "Application-awareness in SDN," in *Proc. of ACM SIGCOMM 2013*, 2013, pp. 487–488. [Online]. Available: http://doi.acm.org/10.1145/2486001.2491700
[2] H. Mekky, F. Hao, S. Mukherjee, Z.-L. Zhang, and T. Lakshman, "Application-aware data plane processing in SDN," in *Proc. of HotSDN 2014*, 2014, pp. 13–18. [Online]. Available: http://doi.acm.org/10.1145/2620728.2620735
[3] P. Sun, M. Yu, M. J. Freedman, J. Rexford, and D. Walker, "Hone: Joint host-network traffic management in software-defined networks," *Journal of Network and Systems Management*, vol. 23, no. 2, pp. 374–399, 2015.
[4] H. Ballani, P. Costa, C. Gkantsidis, M. P. Grosvenor, T. Karagiannis, L. Koromilas, and G. O'Shea, "Enabling end-host network functions," in *Proc. of ACM SIGCOMM 2015*, 2015, pp. 493–507.
[5] N. Khademi, D. Ros, M. Welzl, Z. Bozakov, A. Brunstrom, G. Fairhurst, K.-J. Grinnemo, D. Hayes, P. Hurtig, T. Jones, S. Mangiante, M. Tüxen, and F. Weinrank, "NEAT: a platform- and protocol-independent internet transport API," *IEEE Communications Magazine*, March 2017, accepted for publication.
[6] "NEAT GitHub repository," https://github.com/NEAT-project/neat, 2016.
[7] J. Ahrenholz, "Comparison of CORE network emulation platforms," in *Military Communications Conference, 2010-MILCOM 2010*. IEEE, 2010, pp. 166–171.
[8] K. Phemius and M. Bouet, "Monitoring latency with openflow," in *Network and Service Management (CNSM), 2013 9th International Conference on*. IEEE, 2013, pp. 122–125.