Self-Generated Intent-Based System

Mehdi Bezahaf*, Marco Perez Hernandez †, Lawrence Bardwell‡, Eleanor Davies*,
Matthew Broadbent*, Daniel King* and David Hutchison*
*School of Computing and Communications
Lancaster University
Email: [mehdi.bezahaf, eleanor.davies, m.broadbent,d.king,d.hutchison]@lancaster.ac.uk
†Department of Engineering
University of Cambridge
Email: mep53@cam.ac.uk
‡Mathematics and Statistics
Lancaster University
Email: l.bardwell@lancaster.ac.uk

***Abstract*—We propose an intent-based system where, on top of the user intentions, the system itself generates suitable Quality of Service and resilience parameters and may augment the intent characteristics if it detects any room for improvement. We demonstrate the feasibility and challenges of such a system using mininet and the ONOS controller.**

***Index Terms*—Intent, Self-generated, networking**

I. Introduction

Recently, intent and Intent-Based Networking (IBN) concepts have created enormous interest in academia and industry, despite the fact that the idea is not original at all. In fact, in 2015, the concept of IBN has been presented in RFC 7575 [1] and proposed as a new network management framework in OpenDaylight Network Intent Composition [2].

The idea behind these concepts is to give the opportunity to the user and the operator to express their intentions (i.e., a desired state or behavior) without the need to specify every technical detail of the process and operations to achieve it [3]. The IETF Network Management Research Group (NMRG) has already submitted three "work in progress" Internet-Drafts about the topic. In their active work, they define the concept and they give an overview of Intent-based networking [3], a classification of different intents [4], and they propose a framework of intents [5].

In this paper, we present a new approach where the intent is not only generated by the end-user, the application, or the operator but also by the system itself. In fact, for Quality of Service (QoS) purposes, the system can itself detect network improvements and expresses them through intents.

This paper demonstrates the feasibility of such an approach under a flexible testbed based on mininet[1], OVS switches[2], and the ONOS OpenFlow controller[3].

[1]An Instant Virtual Network on your Laptop - www.mininet.org
[2]Open Virtual Switch - www.openvswitch.org
[3]A new carrier-grade SDN network operating system designed for high availability, performance, scale-out - https://onosproject.org

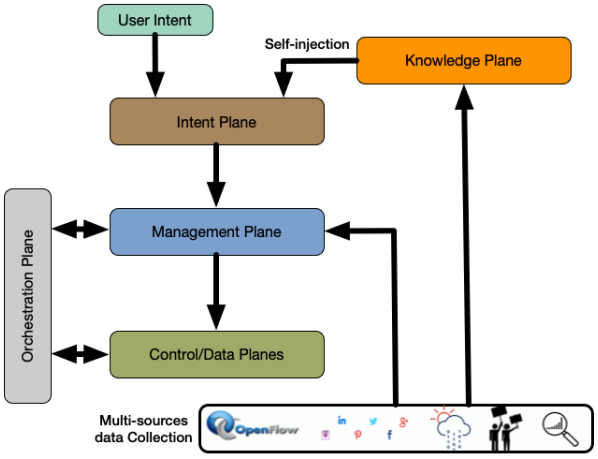

Fig. 1. Architecture Planes.

II. Self-intent Framework

In this section we introduce briefly our framework and how the system itself generates intents. Due to space constraints, we are not going to define each plane in detail. As shown in Figure 1, intents from the user (application or operator) go to the intent plane, where the request is translated, normalized, decomposed and validated before it gets transferred to the management plane.

The management plane makes sure that there are enough resources available to answer the intent. It actively collects data from the data plane and uses techniques like continuous integration, continuous deployment (CI/CD) to ensure that the new intent will not impact the existing intents in the system. Once the verification is done, the new configuration will be delivered to the control plane to be applied.

In parallel to the user's intent, the system collects data from different sources (e.g. weather, political or social networking information) and serves them to the knowledge plane as an input. The knowledge plane filters, adapts and classifies the data in the first place, then using big data algorithms, machine learning, and deep learning, analyses and does some

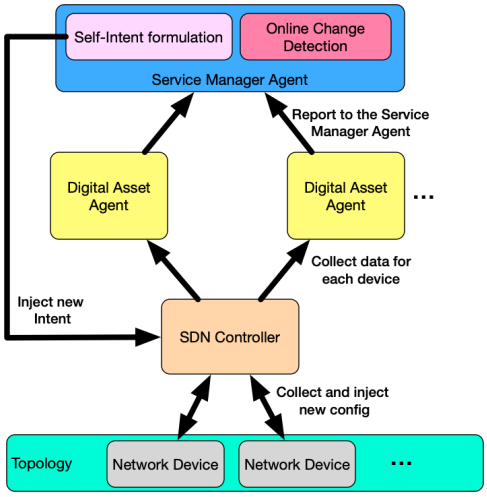

Fig. 2. System description.

reasoning to finally generates a self-intent that can improve the performance of the system.

III. Implementation

In this paper, we do not propose how to express intent and how to handle it. Instead, we use the ONOS intent framework to express intent [4]. Our aim is to focus on how we collect the data and how to use it in order to self-generate a new intent. For the purpose of the demo, we use ONOS as the SDN controller but it can be any other controller. Moreover, we have implemented online change detection to trigger the self-intent, but we can use any other technique or algorithm to do that. The change detection is only an example. Another point to clarify is that we use only data collected from OpenFlow, but of course, we can use data collected from multiple sources.

Figure 2 depicts a description of our implementation, which is composed of three main blocks:

A. SDN Controller

The Software-Defined Networking (SDN) architecture brings the ability to separate physically the network control plane form the forwarding plane. The SDN controller or network controller is capable to interact directly with the networks node through different APIs [6]. It defines exactly which actions to take when certain conditions are valid. For example, creating flow rules or getting the list of meter entries applied to the specified infrastructure device.

B. Agents

The agents are in charge of monitoring the state of the network periodically and aggregate measurements taken according to configured time windows in order to trigger control actions via SDN Controller. The agents can cross data coming directly from the network (via SDN Controller) with other sources of data not available at the SDN Controller (e.g. environmental sensors linked to the network). There are two key control agents:

[4]Intent Framework - wiki.onosproject.org/display/ONOS/Intent+Framework

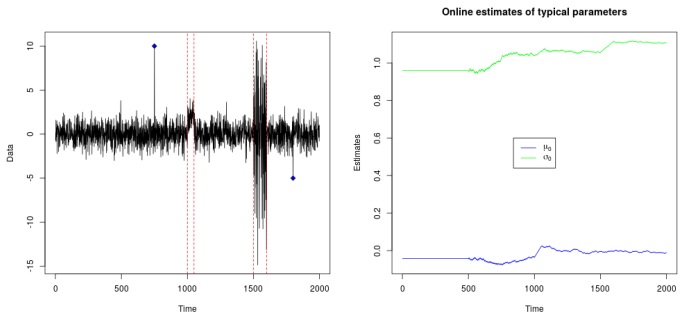

Fig. 3. Online data with collective and point anomalies (LHS). Online estimates of the typical parameters (RHS).

1) Digital Asset: A digital asset can be in charge of one or multiple nodes (or switch) of the network at the same time. It queries the SDN Controller (on real-time and periodically) for the set of relevant metrics (e.g. the throughput) related to the controlled nodes. It aggregates temporarily the data collected and reports aggregated data to the Service Manager agent. A digital asset agent can also collect data from devices that are not representing nodes on the network (e.g. a set of environmental sensors).

2) Service Manager: It collects aggregated data from each digital asset agent and run in real-time the online change detection algorithms (defined just below). In the case of an anomaly, this agent checks the actions required given the current state of the network and triggers a new intent that will be handled by the network controller.

C. Online Change Detection

As highlighted above, the Service Manager agent is capable to run in real-time the online change detection algorithm against the received data coming from a monitored node. Our model for the data is that at time t, the received data process is in one of three states:

1) Typical behavior: Observations are drawn independently from a Normal distribution with (known) mean μ_0 and standard deviation σ_0.

2) Collective anomaly: For an interval $[s, t]$, all observations within that interval are drawn independently from a Normal distribution with (unknown) mean μ and standard deviation σ, where $\mu \neq \mu_0$, and $\sigma \neq \sigma_0$.

3) Point anomaly: A single point isolated in time modeled as an observation drawn from a Normal distribution with a different standard deviation.

The plot on the left hand side of Figure 3 shows the online data set of length of two thousand time points. The typical parameters are $\mu_0 = 0$ and $\sigma_0 = 1$. This time series has two point anomalies (at times 750 and 1800) highlighted by the blue diamonds. There are also two collective anomalies present: the first a change in mean on the interval $[1000, 1050]$ and the second a change in variance on the second interval $[1500, 1600]$.

Our online change detection algorithm is made up of two parts. One part of the algorithm estimates the typical

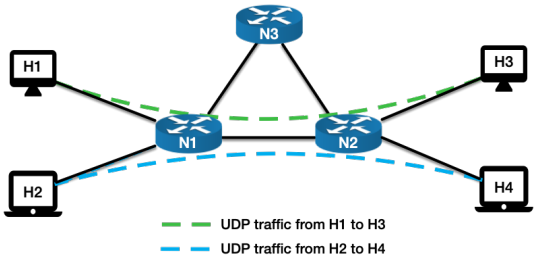

Fig. 4. Scenario Topology.

parameters μ_0, σ_0 in an online fashion as they are assumed to be unknown when monitoring begins. A small burn in period is typically used to initialise these estimates. In the plot on the right hand side of Figure 3 a burn in period of five hundred time points was taken. Due to the fact that anomalies are present in the data we estimate these parameters robustly by using the median and inter-quartile range. The online quantile estimator developed in [7] is used as this algorithm requires only a minimal amount of storage unlike other quantile estimation procedures.

The second part of the algorithm infers the position of collective and point anomalies using a penalised likelihood approach based on the offline approach described in [8]. In essence every time a new datum is observed it is standardised using the most recent estimates for the mean and standard deviation and then allocated to one of the three states described above that maximises the log-likelihood of the data up-to that point.

IV. Demonstration

We use the ONOS OpenFlow controller as an SDN controller and OVS switches. Each agent (both the digital asset and Service manager) is running in separate Docker containers. For each individual switch, we initiate a digital asset agent that will be responsible for collecting data of that switch. All agents code is written in Scala[5]. The anomaly detection code is written in R[6].

A. Scenario

In order to evaluate our implementation, we use a simple scenario with four hosts and three network nodes (Figure 4). Please note that all links have a capacity of 1GB/s.

Users connect through "Host-to-Host" intent (user-intent), where H_1 wants connectivity to H_3 and H_2 wants connectivity to H_4. We modify iPerf[7] code to be able to generate traffic from different sources with varying throughput between hosts to exercise the anomaly detection.

We generate UDP traffic between the different end hosts (H_1 to H_3 and H_2 to H_4, both through N_1 and N_2), and vary traffic to saturate the link $N_1 - N_2$.

[5]The Scala Programming Language - https://www.scala-lang.org
[6]The R Project for Statistical Computing - https://www.r-project.org
[7]iPerf - The ultimate speed test tool for TCP and UDP - https://iperf.fr

It is true that even if the link $N_1 - N_2$ is saturated, the user-intent is still valid (i.e., some packets are lost but the connectivity intent is still valid).

By injecting QoS and resilience to the initial user intent, the system detects, using our online change detection algorithm, that the link $N_1 - N_2$ is saturated and expresses a self-intent asking the SDN controller to re-forward, for example, the traffic between H_1 and H_3 through N_3. The self-intent can also be expressed to the human board to take further decisions.

V. Conclusion

We have demonstrated the viability of implementing a self-generated intent-based system.

In this paper, we use the ONOS intent framework to express intent and we focus on the process of how to trigger self-generated intent.

We have used throughput as the main feature to observe and to react to. However, we can use other features such as device temperature, changing costs, or even multiple features. The initial choice of using change detection is purely practical, and of course, other algorithms and events can be used to trigger the self-generation intent.

For future work, we can focus on how to predict failure or possible improvements in advance and apply them through self-intent. We also want to explore multiple sources data collection and how we can predict actions by using them.

Acknowledgment

The authors gratefully acknowledge the support of the Next Generation Converged Digital Infrastructure (NG-CDI) Prosperity Partnership project funded by UK's EPSRC and British Telecom plc. This work has benefited from an early discussion with Laurent Ciavaglia from Nokia.

References

[1] M. Behringer, M. Pritikin, S. Bjarnason, A. Clemm, B. Carpenter, S. Jiang, and L. Ciavaglia, "Autonomic Networking: Definitions and Design Goals", RFC 7575, June 2015.
[2] OpenDaylight, Network Intent Composition:Main, Jan 2015.
[3] A. Clemm, L. Ciavaglia, L. Granville, and J. Tantsura, "Intent-Based Networking - Concepts and Overview," Internet Engineering Task Force, Internet-Draft. July 2019.
[4] C. Li, Y. Cheng, , J. Strassner, O. Havel, W. Liu, P. Martinez-Julia, J. Nobre, and D. Lopez, "Intent Classification," Internet Engineering Task Force, Internet-Draft. July 2019.
[5] Q. Sun, W. Liu, and K. Xie, "An Intent-driven Management Framework," Internet Engineering Task Force, Internet-Draft. July 2019.
[6] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, and G. Parulkar. "ONOS: towards an open, distributed SDN OS". In Proceedings of the third workshop on Hot topics in software defined networking (HotSDN '14). ACM, New York, NY, USA, 1-6. August 2014.
[7] L. Tierney. "A space-efficient recursive procedure for estimating a quantile of an unknown distribution". SIAM Journal on Scientific and Statistical Computing, 4(4):706711. 1983.
[8] A. T. M. Fisch, I. A. Eckley, and P. Fearnhead. "A linear time method for the detection of point and collective anomalies". ArXiv e-prints. 2018.