DNA: An SDN Framework for Distributed Network Analytics (Demo Paper)

Alexander Clemm, Mouli Chandramouli, Nitish Gupta, Robert Lerche, Ashwin Pankaj, Manjunath Patil, Ganesan Rajam, Anbalagan V, Joe Zhang, Yifan Zhang
Cisco Systems, Inc.
San Jose,California – Bangalore,India – Shanghai,China
{alex|moulchan|nitisgup|rlerche|apankaj|mpatilr|grajam|anbv|yzhang4|yifazhan}@cisco.com

Abstract— **Analytics of network telemetry data helps address many important operational problems. Traditional Big Data approaches run into limitations even as they push scale boundaries for processing data further. One reason for this is the fact that in many cases, the bottleneck for analytics is not analytics processing itself but the generation and export of the data on which analytics depends. The amount of data that can be reasonably collected from the network runs into inherent limitations due to bandwidth and processing constraints in the network itself. In addition, management tasks related to determining and configuring which data to generate lead to significant deployment challenges.**

In order to address these issues, we propose a novel distributed solution to network analytics that we have implemented as a proof of concept, called DNA (Distributed Network Analytics). In DNA, analytics processing is performed at the source of the data by specialized agents embedded within network devices, which also dynamically set up and reconfigure telemetry data sources as required by an analytics task. An SDN controller application orchestrates network analytics tasks across the network to allow users to interact with the network as a whole instead of individual devices one at a time. Our demonstration of DNA includes a GUI front end used to specify and monitor analytics tasks as well as visualize analytics results.

I. Overview

Networks offer rich sets of network telemetry data, such as flow records, interface statistics and MIB data, or IPSLA performance measurements. Analyzing this data can yield important insights about a network, such as trends and changes occurring in traffic patterns and network usage, about network intrusions and attacks on the network or its users that may be in progress, or about instabilities that may lead to degradations or fluctuations in service quality. This means that analytics can help operators of networks and networking services solve many operational problems, and vice-versa, networks are an interesting application area for analytics.

Traditional analytics involves hauling large volumes of data to a backend where that data is subsequently processed and analyzed. In addition to the ability to process the collected data, the effectiveness of analytics depends on the availability of a large set of raw data in the first place. Generating network telemetry data is expensive and limited by resources in the network. For example, interface statistics provided by a Management Information Base (MIB) can realistically be polled or read out from the device perhaps once a minute but not every millisecond, although such data would be interesting for analytics involving fast-changing statistics and health data. Similarly, network measurements can only involve a limited number of probes and amount of test traffic at any one time. Netflow is rarely fully turned on but typically sampled. All these remedial measures are taken in order to conserve system resources at network devices as well as bandwidth resources needed to export that data, which can be a particularly critical consideration in deployment scenarios involving WAN links.

This means that even when a network analytics application is able to ingest and process network telemetry data at very large scale using "Big Data" techniques, it is inherently limited by the fact that not all the data that might be useful for the analysis may be available. This is an important limitation that may not be evident at first. This issue is compounded by the inherent wastefulness of many Big Data applications, in which a lot of the generated data ends up being effectively thrown away, while in hindsight other data that might have been generated in its place would have been more valuable. Finally, there is an issue of very practical nature that is often considered a matter of mundane network administration and therefore ignored, although it turns out to be in practice a big hurdle to deployments of network analytics at scale. This concerns the fact that the network needs to be set up for export and collection of the data that is to be subjected to analytics. Targets for exported data need to be configured, measurement probes set up, and thresholds and sampling rates determined. Any solution ignoring those issues or leaving them as "exercise for the users" is fundamentally incomplete.

For those reasons, we are introducing a novel solution for real-time network analytics, which addresses those shortcomings. Distributed Network Analytics (DNA), depicted in Figure 1, is a solution that performs analytics across the network inside network devices themselves, right at the source where operational telemetry data is generated. The solution involves two components: A DNA agent is an embedded application that is deployed at the network element, performing analytics inside the device right where the data is generated. The DNA Agent also sets up and dynamically adjusts the configuration of data sources to generate exactly the data that is subjected to the analytics queries. The DNA Agent is complemented by a DNA Controller, a Software Defined Networking (SDN) control application which orchestrates analytics tasks across the network. This way, users do not have

to set up analytics on a device-by-device basis. Instead, network analytics are provided as a network service in which a user merely specifies the ***scope*** of the network for which the task is to be applied. The scope can be defined by a set of policies or criteria that determine to which devices the analytics task applies and which DNA Agents should therefore participate in analytics processing.

The DNA solution offers important advantages. Not only is the amount of required off-box processing reduced, but CPU and bandwidth within the network are conserved as well. Additional cycles for analytics performed in the device are offset by the avoidance of cycles that would otherwise be required to generate, format, and transmit data that is now no longer required. Likewise, management of analytics tasks themselves is greatly facilitated.

We have implemented a proof of concept of DNA. No inferences about Cisco product strategy or direction should be made. Details of the solution are described in [1]; in the following we describe the aspects that are shown in the demo.

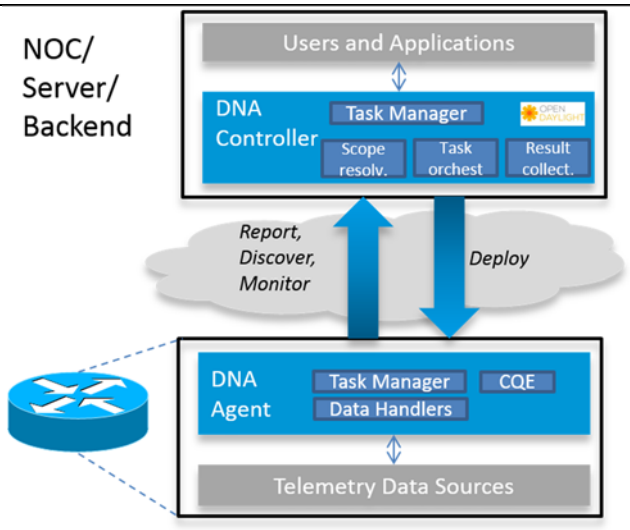

Figure 1: DNA high-level architecture

II. Proof-of-Concept Implementation and Demo Setup

The DNA Controller was implemented as an application on top of Open Daylight. Our implementation took extensive advantage of the model-driven toolchains provided by Open Daylight. From a YANG data model for the definition of network analytics tasks, we generated a Restconf interface that is used by external applications to interact with DNA. This includes our demo application used to configure and monitor network analytics tasks, as well as visualize analytics results. The GUI is depicted in Figure 2.

Because it is readily supported by Open Daylight, Netconf was chosen as communication mechanism between DNA Controller and DNA Agent to orchestrate the deployment of analytics tasks and monitor their status. The analytics results themselves are exported using JDBC (default), or alternatively (in particular when an export destination other than the controller is chosen) IPFIX or syslog. We have integrated Logstash **Error! Reference source not found.** with DNA, allowing users to visualize results from analytics tasks using open source and off-the-shelf applications.

Figure 2: Implementation GUI

A GUI has been implemented on top of the DNA Controller to allow network administrators to interact with the Network Analytics Service provided by DNA, and to visualize network analytics results. Analytics queries are specified by entering an analytics query directly or by picking from a list of templates of "precanned" queries, customizing aspects such as which data items to apply the query to (e.g. link utilization? packet drop rates? round trip jitter?), which aggregators to use (top-n? percentiles? trending?), and any thresholding to apply.

The DNA Agent is implemented as a container-based application, running inside an LXC Linux container hosted on the network element, with a footprint of around 120MB. Support for Netflow, SNMP MIBs, and IPSLA is provided. Communication of results from DNA Agents to the DNA Controller occurs through a JDBC-based push protocol.

The demo consists of a GUI application running inside a browser on a laptop, connecting to a remote DNA Controller. The DNA Controller in turn controls a set of virtualized network devices running inside a remote virtual testbed, VIRL [3][4]. The demo will show how network analytics tasks are configured using a library of customizable templates supplied by the controller. Example tasks include analytics for network brownouts, analyzing trends of interface statistics and IPSLA service level measurements. Network scopes are specified by tags, or simply by drawing a scope on the GUI topology. The analytics task is then deployed across the virtualized network devices, with the GUI providing visual feedback on overall analytics task status. Behavior in presence of certain error conditions, such as a missing capability to support an analytics task in one of the devices, will also be demonstrated. Analytics results returned from the devices are visualized using a timeline and a results panel. A traffic generator, Pagent, is used to introduce interesting conditions in the network, such as additional delay on one of the network links.

References

[1] A. Clemm, M. Chandramouli, S. Krishnamurthy: DNA: An SDN Framework for Distributed Network Analytics. IFIP/IEEE IM 2015.

[2] Logstash. http://logstash.net.

[3] VIRL – Virtual Internet Routing Lab. http://virl.cisco.com.

[4] J. Obstfeld, et al: VIRL, The Virtual Internet Routing Lab, ACM SIGCOMM 2014.