A network protocol for distributed orchestration using intent-based forwarding

Jordan Augé
Cisco Systems
augjorda@cisco.com

Marcel Enguehard
Cisco Systems
mengueha@cisco.com

***Abstract*—Network management systems have undergone tremendous changes to adapt to the growing complexity and diversity of network deployments. A new trend of intent-based frameworks has emerged, where network administrators only input an abstract view of their desired network model instead of specifying the necessary steps to attain it. Intent-based networking holds the promise of autonomous, agile, and learning networks that configure themselves according to abstract policy without requiring human intervention.**

In this demonstration, we introduce a novel architecture for distributed intent-based network orchestration. For this purpose, we first extend the standard YANG to carry intent rather than configuration, adding missing features for in-network orchestration. Second, we use the model intents as network-native objects on which to perform routing and forwarding in-between orchestrators. Finally, we highlight the benefits of pushing intent in the network rather than fully processing it in a centralized orchestrator and illustrate potential improvements in term of scalability, programmability, and reliability.

I. An intent-based orchestration protocol

Most intent-based management frameworks, such as OpenStack Heat[1] or the Open-Networking Foundation Boulder project[2], act as north-bound interfaces on top of a general-purpose orchestrator. Their goal is then to translate intent into orchestration actions (e.g., configuration files or command-line instructions), which are then offloaded to other modules of the orchestrator. While such centralized approaches represent a first step towards intent-based networking, they suffer from stark limitations. Mainly, they keep orchestration centralized, which results not only in scalability issues (OpenStack, for instance, has over 9M lines of code, and running a controller requires at least 3 physical machines, each with 12 CPU cores and 64GB of RAM[3]) but also restricts the possibilities offered by intent-based networking in terms of network automation and innovation.

We argue that many limitations of current orchestrators come from the necessity to perform an early and centralized binding from the user's request to a set of device configurations, requiring full knowledge and preventing any further actors to participate in the resolution of the intent. We thus propose to push the intent deeper into the network fabric rather than limiting it to the edge, which means both transporting and processing it in-between network elements or orchestrators. Fortunately, the recently established consensus around network management protocols such as NETCONF [1] and YANG [2] brings an opportunity to realize such an architecture.

A. Intent-based network model

The first requirement towards distributed orchestration is to have all involved entities – users, orchestrators and resources – agree on a representation model for intent. The work around the YANG data model and its widespread adoption provide a major step in that direction, as it is extensible and allows decoupling the model structure from its semantic aspects. Despite sharing some similarities with object-oriented programming or relational databases, YANG does not include all of their features and thus lacks some expressiveness for encoding user intent or exposing relations and constraints within or in-between resources. To that end, we enhance the model with abstraction, foreign models, and scheduling information, as done in the aforementioned research communities.

Abstraction: Like [3], we propose to extend YANG with abstract objects by proposing standard models for abstract services (e.g., DNS server, node, or relational database) and resource inheritance through the `extends` keyword. This brings two advantages: it unifies both configuration and intent models, simplifying the translation process, and enable late resource specialization (e.g., from node to Linux Container or Xen VM) in subsequent orchestrators or even at end-devices.

Foreign models: As such, YANG models only describe per-device configuration, which prevents, for instance, device-to-device cooperation to offload part of the scheduling from the orchestrator. We thus propose to introduce in YANG models the notion of abstract *foreign* objects. For instance, a DNS server can be made aware of the existence of a new node for which it must record a domain name and retrieve the assigned IP address directly from the device on which the node is deployed instead of relying on the orchestrator.

Scheduling information: Foreign requirements are not enough on their own to enable efficient device-to-device cooperation. Indeed, it still presents a flat temporal model while certain services are ordered (some services require others to be completed before them, e.g., a domain-name registration comes after IP address assignment). We thus enrich YANG `groupings` to distinguish *parallel* and *sequential* groupings.

[1] https://wiki.openstack.org/Heat

[2] https://www.opennetworking.org/incubator-projects/boulder/

[3] https://www.stratoscale.com/blog/openstack/openstack-hardware-requirements-and-capacity-planning-servers-cpu-and-ram-part-1/

B. Model-based routing and forwarding

In a second step, we propose to depart from the traditional star deployment where all resources are reachable from a single orchestrator, to a more distributed approach where resources attached to different orchestrators advertise their capabilities throughout the network and allow users' intents to be routed back to those able to best satisfy the request. This essentially means building a routing and forwarding plane and more specifically as they are built in Information-Centric Networking (ICN, e.g. NDN [4]). Like the former, our proposal uses location-independent identifiers but these are multidimensional structures representative of the intent itself rather than one-dimensional names. Each orchestrator now becomes a router and is in addition able to split (composition) or transform the incoming request (specialization, attribute binding), which gets fully resolved upon reaching end devices of interest. This approach replaces the point-to-point transport of NETCONF by a multipoint-to-multipoint network propagation.

The design of our intent router closely follows the structure of an ICN router [4], where the three main data structures have been adapted to our new addressing scheme: the Forwarding Information Base (FIB), used to match incoming intent requests to network locations, the Pending Intent Table (PIT), that keeps state to symmetrically route answers back to the origin of the corresponding intent, and a Content Store (CS) that can be used to cache answers.

The FIB is the main component of our router. It maps received resource advertisements to their ingress interface and uses this information to forward an intent to one or several next hops able to further satisfy it. This happens after an eventual local processing where the intent can be (partially) bound, specialized, or decomposed into multiple sub-intents using the network model. To perform the matching process over multi-dimensional intent objects, we developed a preliminary solution based on maximal subset matching algorithm [5].

Like ICN, we store incoming requests into the PIT along with their ingress interface and associated outgoing sub-requests. The PIT thus collects all the necessary information to send the corresponding answers back up to their origin, thereby implementing symmetric routing. It serves as a distributed scheduling module, keeping track of requests in progress and allowing hop-by-hop reconciliation of concurrent messages. The PIT is thus crucial to ensure the synchronization and consistency of the execution in a multipoint-to-multipoint concurrent environment. Furthermore, the PIT contributes to the scalability of the system by aggregating redundant requests and execute them only once.

The CS caches responses flowing back through the router, and stores the current state of a network model deployment for improved performance. It can be used to access such information with lower RTT and network overhead. It also brings resiliency as the network can still at least partially operate during disconnection periods.

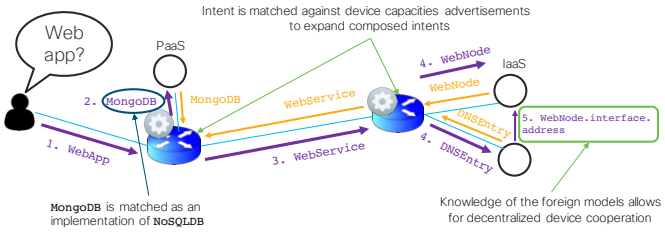

Fig. 1. Demonstration use case

II. Demonstration

We demonstrate the feasibility of our Intent-based Forwarding Architecture by implementing it in the vICN orchestrator [6]. It is an intent-based orchestrator for ICN networks, designed to address the lack of adequation between existing IP-based tools with the requirements of the community.

As an example, we consider a client provisioning a web application (`WebApp`) composed of a front-end (`WebService`) and a NoSQL back-end (`NoSQLDB`). The user has a single point of contact F that can provide such functionality (for convenience of use, billing, etc.), only requiring a unique name that will be used to construct and manage the DNS entry of the service. Database service is ensured by a specialized Platform-as-a-Service (PaaS) advertising a NoSQL service, while the hosting platform for the web application is managed as Infrastructure-as-a-Service (IaaS) and provides containers to its users. F federates and integrates the other platforms to offer a consistent service to its user while hiding the implementation details. The corresponding deployment is depicted in Figure 1.

First, we illustrate how our orchestration protocols allow for *distributed intent resolution*. Indeed, as the intent propagates through the first router, the `WebApp` is decomposed in its Database service and its front end using the model. The *inheritance model* is then used to match the NoSQL service to the MongoDB capacities of the PaaS in the router FIB. The characteristics of that service (e.g., IP address, identifiers, etc.) can then be used to update the `WebService` intent with the necessary information. Similarly, at the next orchestration router, the `WebService` is divided into `WebNode`, which is a containerized HTTP server, and `DNSEntry`, which maps this server's IP address to a domain name for public access. In both cases, our protocol allows resolving intent *on-demand* in the network in a distributed fashion. This simplifies orchestrator design and frees it from being all-knowledgeable.

References

[1] R. Enns, et al. "Network Configuration Protocol (NETCONF)." RFC 6241, Jun 2011. URL https://rfc-editor.org/rfc/rfc6241.txt.

[2] M. Bjorklund. "The YANG 1.1 Data Modeling Language." RFC 7950, Aug 2016. URL https://rfc-editor.org/rfc/rfc7950.txt.

[3] S. Kuryla, et al. "Extending YANG with Language Abstractions." RFC 6095, Mar 2011. URL https://rfc-editor.org/rfc/rfc6095.txt.

[4] L. Zhang, et al. "Named data networking." *ACM SIGCOMM CCR*, 2014.

[5] N. Alon and R. Yuster. "Fast algorithms for maximum subset matching and all-pairs shortest paths in graphs with a (not so) small vertex cover." In *European Symposium on Algorithms*, Springer, 2007.

[6] M. Sardara, et al. "Virtualized ICN (vICN): towards a unified network virtualization framework for ICN experimentation." In *ACM ICN'17*.