SDN Heading North: Towards a Declarative Intent-based Northbound Interface

Shiyam Alalmaei*†, Yehia Elkhatib*, Mehdi Bezahaf*, Matthew Broadbent*, and Nicholas Race*
*Lancaster University, United Kingdom †Princess Nourah Bint Abdulrahman University, Kingdom of Saudi Arabia
Email: {s.alalmaei, y.elkhatib, mehdi.bezahaf, m.broadbent, n.race}@lancaster.ac.uk

***Abstract*—The Intent-Based Northbound Interface (NBI) offers users the ability to express *what* they want to achieve instead of *how* to achieve it, enabling improvements to network management and reducing operational costs. However, development of an Intent-Based NBI remains in its infancy. Existing solutions do not allow users to express high-level operational targets that appropriately capture business objectives, nor link these to lower-level management policies and operations. We propose an extensible Intent-based NBI framework and a higher-level declarative intent expression to enable service-oriented intents with different targets. We focus on the creation of intents and their mapping from high-level expressions to low-level policies, and consider this from the perspective of an intent developer in the context of a Cloud CDN use case.**

***Index Terms*—SDN, northbound interface, declarative, intents, cloud CDN**

I. Introduction

The success of SDN relies on the ability of application developers to leverage the underlying network to design and build services. This is achieved through the *northbound interface* (NBI) [1]. Without an intuitive and efficient NBI, SDN will fail to reach its potential. It is crucial that SDN gains momentum within user communities who do not necessarily have detailed networking backgrounds. Hence, the NBI is the key enabler for the realisation of the ultimate SDN promise.

Intent-based NBIs can be used by network application developers and/or users who might not be network experts. They provide a more intuitive mechanism to express specific service demands in the form of *intents*. It is, therefore, unsurprising that intent-based NBIs are gaining a lot of attention both in industry and academia. However, there has been little progress in their development that would enable users to express high-level targets in terms of business objectives, and linking them to lower-level management policies and operations. This is a crucial mechanism that enables users to express *what* they want rather than *how* to do it. It facilitates faster network management, service deployment, realization of user objectives and lower OPEX due to reduced human involvement.

In this paper, we propose an Intent-Based NBI framework to address the above challenges. Our contributions are:

1. An outline of the state of the art in Intent solutions (§II–III).
2. An Intent-Based NBI framework architecture (§IV).
3. A declarative intent expression and corresponding decomposition into prescriptive policies (§V).
4. An intent cloud CDN use case with experimental results (§VI–VII).

978-3-903176-31-7

II. Background

A. Intents and policies

An Intent is a declarative expression of the operational goals or expected outcomes that the system should deliver. Therefore, ultimately, intents could be used in many domains and by different users, even non-experts.

In contrast, a Policy is a single or set of rules that govern system behavior [2]. Typically, a rule consists of events, conditions, and actions (ECA). These prescriptive policies let users specify precisely what to do and under which circumstances [3]. Unlike intents, policies do not specify desired goals, but both notions provide system abstraction that could be technology-agnostic.

B. Different intent types

Intent user requirements can be classified as client-facing service-layer requirements and operator-facing resource-layer ones. Network clients tend to express their high-level requirements in a service-layer expression that is concerned with service performance guarantees (e.g., throughput), whereas network operators use a resource-layer expression to express internal resource and operational requirements (e.g., energy-savings). Due to the inherent nature of each, service-layer requirements should be expressed in a declarative way whereas resource-layer requirements in a prescriptive fashion. A mechanism is needed to decompose high-level declarative intents to abstract prescriptive policies that define service behavior, which then gets translated to lower-level commands.

Most existing Intent-Based NBIs are prescriptive, requiring some knowledge from users. They are still limited and mainly focused on networking operations (i.e., connectivity), whereas users' high-level requirements have now evolved to other services (e.g., load balancing, placement, etc.). Also, they do not provide tools to create new intents, limiting extensibility.

In this paper, we focus on Service Consumer intents, which are developed by service providers who have the technical knowledge, and understanding of possible user requirements and desired targets. To narrow down our scope, we focus on a Cloud CDN scenario, where service providers (CDN Operators) are the intent and policy developers, while service consumers (Content Providers) are the intent's users.

TABLE I: Summary of the results of our meta-analysis of intent-based solutions.

Intent-Based Solution	Intent Expression	Domain	Level
Boulder [3]	*Subject, Predicate, Object: {Constraints, Conditions}*	Networking	Presc.
ONOS Intent Framework [4]	*Network Resource, Constraints, Criteria, Instructions*	Networking	Presc.
(NEMO) by Huawei [5]	*Object + Operation* or *Object + Result* (under test and not used yet)	Networking	Presc. Decl.
Group-based Policy (GBP) [6]	*Endpoint group, contract {subject: {rules: {classifier and action set}}}*	Networking / NFV	Presc.
(NIC) by HP [7]	*Source Composite Endpoint, Destination Composite Endpoint, Traffic operation and constraints*	Networking / NFV	Presc.
(DOVE) by IBM [8]	Not specified	Networking / NFV	–
Intent-based virtualisation Platform [9]	*Resources, Conditions, Priority, and Instructions*	Networking / NFV	Presc.
(INSpIRE) [10]	*Traffic Type, Source, Destination, Context level, Contexts list*	Networking / NFV	Presc.
Intent-based NBI service-oriented architecture [11]	application-specific language	Networking	–
(iNDIRA) [12]	*Subject (Service or Condition), Relationship (has Arguments), Objects (multiple parameters)*	Networking	Presc.
(SENSE) [13]	*Service type, Service alias, Connections: {name,terminals, bandwidth: { qos_class, capacity, unit}}, schedule: {start, end, duration}*	Networking / NFV	Presc.
Interactive Intent-based Negotiation Scheme [14]	*Verbs, Nouns, Modifiers*	Networking / NFV	Presc.
(MD-IDN) [15]	*Action, Endpoint 1, Traffic type, Endpoint 2*	Networking	Presc.
Janus system [16]	*Endpoint-Group1, Connection attributes: {protocol, port, bandwidth, latency, middle-box }, Endpoint-Group 2*	Networking	Presc.
Northbound Interface [17]	*Predicate, Commodity, Target (resources), Constraint, Condition*	Networking	Presc.
Adaptive Service Deployment [19]	*Verb, Object, Modifiers, Subject*	General use cases, e.g. storage, caching, IDS	Presc.

C. Intent-Based NBIs challenges

The Intent-based NBIs challenges faced by intent users and developers are as follow:

- Service consumers require declarative rather than prescriptive intent expressions.
- Translating service-oriented intents to system operations needs intermediate interpretation as policies.
- Intent-based NBI has to be platform-independent, extensible and beyond network-level operations.

III. Meta-Analysis of Related Work

We scrutinized various Intent-based solutions that have been proposed by standard development organizations and academic researchers. Table I shows an overview of our comparison based on several criteria: *Intent Expression* shows how intents are expressed, *Domain* indicates the domain that is being addressed by the intent (network, NFV, general); and *Level* classifies intents as either prescriptive (Presc.) or declarative (Decl.). In general, current Intent-based NBIs are still limited, ad-hoc and to an extent vendor-specific. Even though they offer important prescriptive intents for network services like connectivity and chains of VNFs, they do not allow expressing declarative intents that handle other requirements beyond the network-level. Moreover, most current NBIs offer a predefined set of intents and don't provide the tools to create new intents and map them to lower-level policies.

IV. Intent-Based NBI Framework

In order to address the identified gap, we propose an intent-based framework as depicted in Fig. 1. Its components follow.

Service Consumer Intent API: an interface allowing users to express (add, update, remove) declarative intents (Step ①).

Intent/Policy Mapper: maps between declarative intents and corresponding prescriptive abstract policies (Step ②), by referring to Intent and Policy Descriptors which contain the descriptions and expression templates

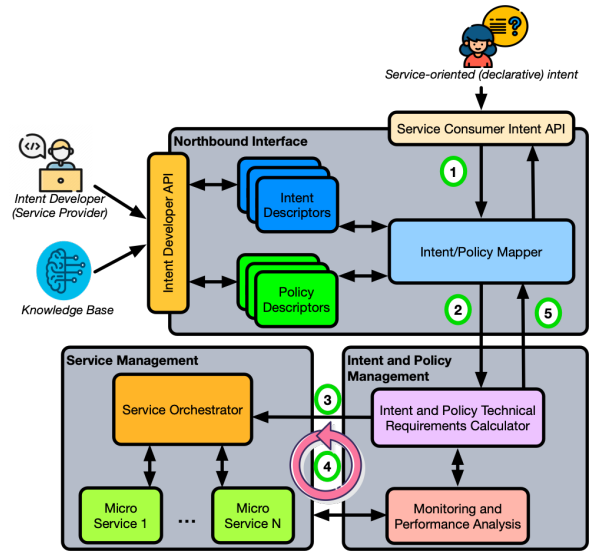

Fig. 1: The proposed intent-based framework.

Intent Developer API: allows intent developers to create intents and corresponding policies based on their expertise; or a knowledge base could be used to derive new intents and policies from previous experience.

Intent and Policy Technical Requirements Calculator: calculates and dynamically updates operational parameters and technical requirements of the abstract policies while considering consumer targets, available resources, etc. It feeds the Service Orchestrator with the behavioral guideline policies that dictate the technical requirements (Step ③), which will get translated to their corresponding micro-services API calls. It is also responsible for sending feedback to the consumer about the intent status and the capability of honoring it (Step ⑤).

Service Orchestrator: coordinates the service behavior by regulating the interactions and configurations of the corre-

TABLE II: Basic expression syntax.

Tag	Basic Expression
<SERVICE>	Caching
<RESOURCES>	contents to be cached
<CONJUNCTION>	that can handle, that can meet, etc.
<TARGET>	<WORKLOAD>
<WORKLOAD>	<NUMBER> <UNIT> or <ADJECTIVE> <UNIT> or <ADJECTIVE> "workload"
<NUMBER>	numeric values that can represent the workload
<UNIT>	GB/min, requests/sec, etc.
<ADJECTIVE>	max, min, dynamic, high, medium, etc.
<CONDITIONS>	new caching request, max threshold exceeded, ...
<ACTIONS>	allocate cache servers(), scale out(), ...
<CONSTRAINTS>	with max storage, with least latency, ...
<PRIORITY>	optional indicator of policy priority
<OPERATOR>	Policy will be executed in parallel / sequential way

sponding micro-services while considering the prescriptive policies as guidelines (from Step ③).

Monitor and Performance Analysis: monitors intents and policies performances. It enables the Intent and Policy Technical Requirements Calculator to keep refining the policies dynamically (Step ④).

V. Intent Expressions, Syntax, and Descriptors

As service-oriented intents are high-level descriptions, we use the following syntax to express the user targets:

<SERVICE><RESOURCES><CONJUNCTION><TARGET>

where each element is defined in Table II. This description is decomposed into a set of prescriptive policies that work jointly to achieve the target as:

<SERVICE><RESOURCES>
<POLICY 1><OPERATOR><POLICY 2><OPERATOR>...
{<POLICY k_1><OPERATOR><POLICY k_2><OPERATOR>...<POLICY k_m>}
<OPERATOR>...<POLICY n>

where the curly brackets ({ }) express a block of grouped policies and each <POLICY i> is expressed as:

<CONDITIONS><ACTIONS><CONSTRAINTS>[<PRIORITY>]

There are two levels of policies (i.e. abstract and technical) as shown in steps ② and ③ in Fig. 1. They both follow the same syntax but the technical policies determine the micro-service technologies used in <ACTIONS> and the technical requirements and operational parameters in <CONSTRAINTS>.

VI. Use Case: Cloud CDN

Content Providers (CPs) were early adopters of cloud services to meet their end-user demands. Recently, public cloud vendors started to provide CDN services on a pay-per-service basis; e.g., Amazon Cloudfront and Microsoft Azure CDN.

Our assumptions for this use case are:

1) The Cloud CDN (CCDN) service type is SaaS; i.e. the CP is provided with a software-based caching that hides away all of the infrastructure details, and only express their targets.
2) Elastic virtual resources provisioning, which could dynamically scale based on demand, resources, etc.
3) Caching and infrastructure management is delegated by the CP to the CDN operator who has direct access and control.
4) The caching scheme is *Pull*-based, i.e. reactively places cache content based on demand.

Nowadays, CCDNs do not consider high-level CP targets (e.g. requests/region) in its caching and resource management.

CCDNs usually carry out several operations: allocate resources, redirect end-user content requests and resize the caching service. These operations could form up the intent's policies. Here, the intent user is the CP and the intent developer is the CCDN Operator. When an intent developer wants to create a new intent, he has to decide how users can express their high-level targets using the declarative intent expression. Therefore, he can provide several targets that can be achieved via the equivalent policies and implementation, based on his knowledge of the CDN, which could be assisted by technical requirements calculator modules.

For example, the CP's declarative caching intent (*"I want caching for content X to handle 20 GB/min"*) can be decomposed into two abstract policies (i.e. allocate and resize the cache service) and expressed in JSON format as in Listing 1. They get injected into the Service Orchestrator to translate them to micro-service API calls.

```
{  "Resources-Allocation": {
          "Conditions": "New Caching Service Request",
          "Actions": "Allocate Cache Servers",
          "Constraints": "Average Number Of Servers"     },
   "Cache-Service-Resizing": {
          "Scale-up": {
                "Conditions": "Max Threshold Exceeded",
                "Actions": "Add more caches",
                "Constraints": "Number Of Caches to Add" },
          "Scale-down": {
                "Conditions": "Underutilized Threshold",
                "Actions": "Remove some Caches",
                "Constraints": "Time"                    } }
}
```

JSON Listing 1: Abstract Policies.

VII. Experimental Results and Discussion

To demonstrate our proposed workload intent for the CCDN use case, we carry out a functional simulation of the Intent and Policy Management Layer in Fig. 1.

A. Dataset

We utilized the data from a major ISP that represents the measurement of the bitrate of data transferred between the CDN caches (in the ISP's facilities) and the end-users. Each measure is average bits/second over the granularity of 1 minute. Our data contains observations from 11–20 Oct 2018.

B. Functional simulation

We assumed that the CCDN provisions cache servers as VMs. The time required to spin up a VM and start the cache varies depending on the location of the VM image and the round-trip delay time [20]. We assumed that the VM image has to be fetched remotely, resulting in a total of 5 minutes to spin up and start the cache. Each cache server has a threshold (i.e. bandwidth, CPU utilization, etc) beyond which

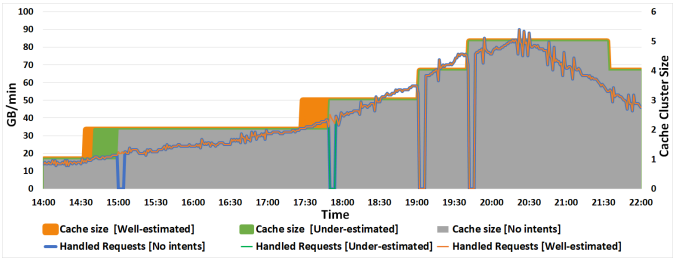

Fig. 2: Cache cluster size and handled data rate in a reactive traditional CCDN (without intents) and an Intent-based one.

the server would not be able to handle any new incoming requests and need some time to relax again, and then resume serving new requests within its threshold limit. The number of concurrent caches (i.e. cluster size) is calculated and updated depending on the incoming request load as obtained from the aforementioned dataset.

C. Evaluation

CP and CCDN Interaction via Intents: Our solution allows CPs to proactively inform the CCDN of their expected workload target through the intent. However, the CP's intent could be *under-*, *over-*, or *well- estimated*.

Traditional CCDNs (without using intent) plan resources reactively without prior knowledge of the estimated demand, which led to 3.21% of unhandled requests ($\simeq$ 1 TB of lost data) when the average workload was 21.63 GB/min. When the CP under-estimates the average workload to 15 GB/min, the unhandled requests drop to 2.88% ($\simeq$ 898 GB of lost data), and this number decreases to 2.25% ($\simeq$ 702 GB of lost data) when the CP well-estimates the workload to 22 GB/min.

We compared the baseline solution (without intent) with our approach (under- and well-estimated workload intent); see Fig. 2. In the traditional CCDN, the cluster would start at 1 cache and add/remove caches reactively (grey area). Each time the CCDN adds a new cache to the cluster (4 times), the number of handled requests drops (blue line). Using the under-estimated workload intent, the system avoids the first dropping (around 15:00) by proactively resizing the cluster to two caches (green area). With the well-estimated workload intent, the system avoided two of these drops by proactively resizing the cluster to two then to three caches (orange area).

Policy refinement: In order to maximize the number of handled requests, the intent system would refine the intent's policies based on its analysis of the previous demand pattern, cluster size changes, and times. One way to achieve this is to scale out/in proactively and start new caches earlier than actually needed. Therefore, two approaches can be taken: a *conservative* method, which aims to preserve the additional cost of deploying extra caches by sacrificing the handled requests, and a *greedy* method, which handles more requests at the expense of increasing the extra-deployment cost.

Fig. 3 shows an example of the initial abstract policies and their refinement. Since the greedy approach is more costly

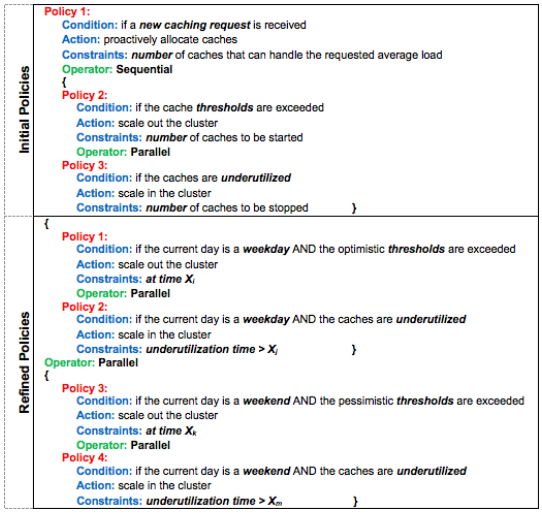

Fig. 3: Abstract intent policy refinement.

compared to the conservative one (for the additional caches' uptime), the CDN operator can alternate between them by applying the conservative cluster resizing (Policy 1 and Policy 2) during the weekdays and the greedy policies during the weekend (Policy 3 and Policy 4).

CP Intent Update: Our solution allows the CP to keep interacting with the CDN to express his changing needs. For example, if a CP expects a workload of 42 GB/min instead of 22 GB/min, he can update his previous intent with a new average workload target. The results show that the traditional approach has resulted in 4.46% ($\simeq$ 2.6 TB) unhandled requests in 24h. The conservative approach reduces this loss to 1.20% ($\simeq$ 718 GB) with a cost of 77mins and the greedy to 0.41% ($\simeq$ 285 GB) with a cost of 417mins. For a comparison purpose, an oracle solution will handle all the requests at a cost of 40mins. It is the CCDN operator's responsibility to tune the thresholds and constraints to minimize the cost. These optimization problems are beyond this paper's scope.

VIII. Conclusion and Future Work

We proposed a Service Consumer Intent declarative expression along with its corresponding prescriptive policy expressions. Unlike current intent expressions, our intent allows service consumers with limited technical knowledge to express their operational targets that are beyond network-level. We discussed our proposed Intent-Based NBI framework and workload intent, and its corresponding policies in the context of a Cloud CDN use case. We discussed some possible intent calculation and refinement to maximize achieving the intent's workload target with respect to different cost alternatives. In our future work, we plan to implement the workload intent and the intent-based framework in a real cloud-based testbed along with the required APIs and translations. Different targets could be investigated as well to discuss how can Cloud CDNs leverage this solution in their futuristic communication mechanisms with content provider.

REFERENCES

[1] ONF, "SDN architecture overview". https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/SDN-architecture-overview-1.0.pdf [Accessed: Dec. 22, 2019].

[2] M. Sloman., "Policy driven management for distributed systems," Journal of Network and Systems Mngmnt., 2(4), pp. 333–360, 1994.

[3] OpensourceSDN, "Boulder intent-based NBI". https://www.opennetworking.org/images/stories/sdn-solution-showcase/germany2015/Boulder\%20-\%20Intent\%20Based\%20NBI.pdf [Accessed: Dec. 03, 2019]

[4] ONOS, "Intent framework". https://wiki.onosproject.org/display/ONOS/Intent+Framework [Accessed: Nov. 04, 2019].

[5] ODL, "NEMO". https://wiki.opendaylight.org/view/NEMO:User_Manual [Accessed: Dec. 03, 2019].

[6] OpenStack, "Group based policy". https://wiki.openstack.org/wiki/GroupBasedPolicy [Accessed: Dec. 03, 2019].

[7] OpenDayLight, "Network intent composition". https://wiki.opendaylight.org/view/Network_Intent_Composition:Main [Accessed: Dec. 03, 2019].

[8] R. Cohen, et al., "An intent-based approach for network virtualization," in IM, 2013.

[9] Y. Han, J. Li, D. Hoang, J. H. Yoo, J. W. K. Hong, "An intent-based network virtualization platform for SDN," in CNSM, 2016.

[10] E. J. Scheid, et. al, "Inspire: integrated NFV-based intent refinement environment," in IM, 2017.

[11] M. Pham, D. B. Hoang, "SDN applications - The intent-based northbound interface realisation for extended applications," in NetSoft, 2016.

[12] M. Kiran, et. al, "Enabling intent to configure scientific networks for high performance demands," in FGCS 79, pp. 205–214, 2018.

[13] I. Monga et al., "SDN for end-to-end networked science at the exascale (sense)," in INDIS, 2018.

[14] A. Marsico, et al., "An interactive intent-based negotiation scheme for application-centric networks," in NetSoft. 2017.

[15] S. Arezoumand, et al., "MD-IDN: Multi-domain intent-driven networking in software-defined infrastructures," in CNSM, 2017.

[16] A. Abhashkumar et al., "Supporting diverse dynamic intent-based policies using Janus," in CoNEXT, 2017.

[17] D. Tuncer et al., "A northbound interface for software-based networks," in CNSM, 2018.

[18] Y. Elkhatib, et al., "Charting an Intent Driven Network," in CNSM, 2017.

[19] A. Elhabbash, et al., "Adaptive service deployment using in-network mediation," in CNSM, 2018.

[20] P. A. Frangoudis, L. Yala and A. Ksentini, "CDN-As-a-Service Provision Over a Telecom Operator's Cloud," in IEEE Trans. Netw. Service Manag., 14(3), pp. 702-716, 2017.