Decentralized Intent-driven Coordination of Multi-Domain IP-Optical Networks

Filippos Christou
Institute of Communication Networks and Computer Engineering (IKR)
University of Stuttgart
Stuttgart, Germany
filippos.christou@ikr.uni-stuttgart.de

***Abstract*—Intent-based networking is increasingly used to improve network control and management. Network operators have already begun to adopt this paradigm, which leads to a simplified and automatized network operation. The operators can interact with their intent-driven networks through the Northbound Interface (NBI). Given a standardized NBI, the same approach can scale to coordinate intent provisioning across multi-domain networks in a decentralized fashion. This can outdate traditional decentralized protocols and open new opportunities for flexible and scalable communication mechanisms. This paper proposes a minimal and general high-level architecture, relying on a standard IBN (Intent-Based Networking) architecture, for multi-domain intent deployment in IP-optical networks. Our architecture is consistent between diverse network operators that use the same NBI, respects confidential information, promotes accountability, and can scale for various network services. To achieve this, we introduce a hierarchical system-generated intent schema with automatic intent delegation between the different domains.**

***Index Terms*—architecture, decentralized, deployment, IBN, multi-domain, NBI**

I. Introduction

Today's Internet owes its existence to various decentralized operations and protocols. However, we have witnessed a strong urge toward centralized approaches during the last decade resulting in the Software-Defined Networking (SDN) paradigm. SDN accumulates the network's knowledge to make centralized decisions and separates the control from the data plane. The Intent-Based Networking (IBN) paradigm complements SDN as an evolution of the policy-based approaches to introduce a systematic way of operating a network with intents as basic building blocks. IBN decouples the implementation details from the network operator's desires or intentions, i. e., intents. The operator's intents, e. g., end-to-end (E2E) connections, can be abstractly defined, while the implementation is handled automatically from the system internals, respecting all physical-level constraints like spectrum contiguity/continuity and maximum optical reach. An IBN Northbound Interface (NBI) is provided to receive the operator's intents. The IBN NBI is the interaction point between the network operator and the networking system. It serves as the interface to add, modify, delete and monitor the intents. A common or standardized IBN NBI will expand these functionalities to allow interaction with authorized entities outside the considered network.

This work has been performed in the framework of the CELTIC-NEXT EUREKA project AI-NET-ANTILLAS (Project ID C2019/3-3), and it is partly funded by the German BMBF (Project ID 16KIS1312).

As much as centralized approaches can be beneficial, coordination often needs to be achieved across different domains, e. g., Autonomous Systems. In these cases, all parties must agree on some common terms while, at the same time, they try to minimize the amount of proprietary information they share. Below we mention some possibilities of how decentralized multi-domain (MD) coordination can be achieved, and we highlight the advantages of an intent-driven solution:

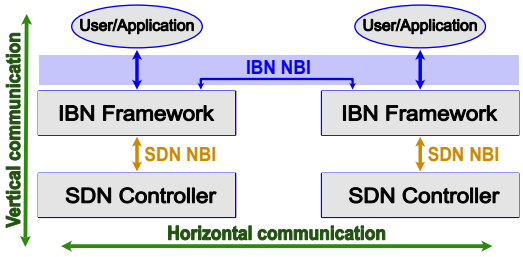
1) MD IBN: MD IBN inherits all the advantages of the IBN paradigm by opening up the IBN NBI to users other than the network owner, like customers, other network operators, or even machines for future autonomous networks. A well-engineered IBN NBI is needed to cope with the complexity of several different users.

2) Proprietary mechanisms: Compared to proprietary mechanisms, the MD IBN coordination is open to every system that complies with the IBN NBI. This accelerates inclusivity among network operators, decreases overall development effort, and leads to open networking.

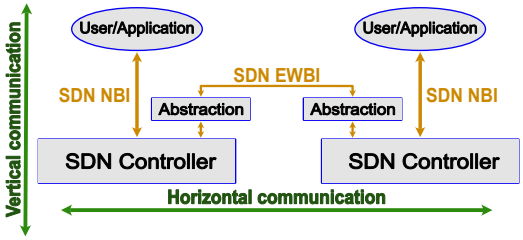
3) Traditional communication protocols: MD IBN provides flexibility, customization, and fast business plan adaptation, unlike most traditional decentralized communication protocols, which support limited operations and are hard to upgrade. For example, new MD communication, like service advertising, can be defined using special interconnection intents, which traditional protocols like Border Gateway Protocol (BGP) cannot support [1].

4) MD SDN: Many approaches have emerged to tackle decentralized SDN control [2]. The SDN East/Westbound Interface (EWBI) is used to achieve horizontal control between the controllers. Although there are many similarities between MD SDN and MD IBN, adding one additional layer of intent abstraction can be beneficial. Due to the high-level abstraction, the IBN NBI can be used for vertical and horizontal communication. Moreover, IBN comes with a built-in architecture to handle the intent lifecycle (as explained in Section III and Fig. 2), which needs only to be expanded for the MD logic. In contrast, such an abstraction layer must be created from scratch for the MD SDN; otherwise, low-level access closer to the devices will be provided to potential competitors. A visual comparison can be found in Fig. 1.

978-3-903176-51-5 © 2022 IFIP



(a) MD IBN



(b) MD SDN

Fig. 1. Decentralized coordination.

MD IBN may be, at the time, an immature solution compared to its alternatives, but the above reasons make it highly promising for the future and very significant for research. This paper presents a decentralized MD IBN coordination architecture, assuming a common IBN NBI, for multi-layer networks, i. e., networks consisting of nodes with an Optical Cross-Connect (OXC) and IP Router.

II. Related Work

Although there are activities toward developing an IBN NBI [3]–[7], a standardized interface still does not exist. Velasco et al. considered MD coordination using an E2E machine learning-powered IBN service orchestrator [8]. Past work also examined IBN orchestration across multiple platforms [9] and across technologically different domains for E2E Service Function Chain management [10]. There were also efforts to introduce a distributed IBN architecture by extending the YANG standard [11]. Moreover, Augé and Enguehard developed some intent-related algorithms for MD IBN [12].

Our work differs in that we consider decentralized domains and build an intent architecture that supports MD multi-layer interoperability. In particular, we employ system-generated intents [13] to compile a user intent to a hierarchical structure we call an *intent tree*. We introduce a process of updating the intent tree state, and we establish the notion of automatic intent delegation between domains, assuming the intents are aligned with the underlying Service Level Agreements (SLAs) as [14] finds necessary. Although the ideas could be applicable for more intent types, we showcase a specific intent compilation strategy for E2E connectivity intents, which are more relevant to Infrastructure-as-a-Service (IaaS) scenarios.

III. Architecture

Our architecture commonly places the IBN framework above the SDN controller. The IBN framework contains all the logic and is responsible for the intent-related decisions. Meanwhile, the SDN controller is treated as a driver, enabling communication between the IBN framework and the network devices. The SDN controller translates the requests of the IBN framework to be device-comprehensible and forwards them.

Fig. 2 illustrates the different stages of an intent. First, the intent enters the system expressed in an intent language. The intent language engine uses the IBN NBI to insert the intent into the IBN framework (Intent Delivery). The IBN framework processes the intent, generates a potential implementation (Intent Compilation), and forwards it to the SDN controller to be deployed in the required devices (Intent Installation). Continuous monitoring assures that the intent is being satisfied (Intent Monitoring). When multiple intents require the same resources, a conflict will be caused. A primitive solution to deal with this is total intent recompilation in order to, for example, minimize spectrum fragmentation. The objective of this work is to focus on the IBN framework.

A. Intent state machine

A fundamental step in developing an intent system is designing the intent state machine, which ranges from minimal designs like Fig. 3 to very complex ones. The intent state machine describes the intent's status and all possible transitions. Fig. 3 shows that an intent must first be compiled to get installed. *Compiling* and *Installing* are intermediate states which signify dependence on the child intents in the intent tree. *Compiling* indicates that some intent compilation takes place, and *Installing* that the intent is already fully compiled but still not up and running on every device that it should. The compilation or installation will fail if resources are unavailable or the intent requirements are not satisfied.

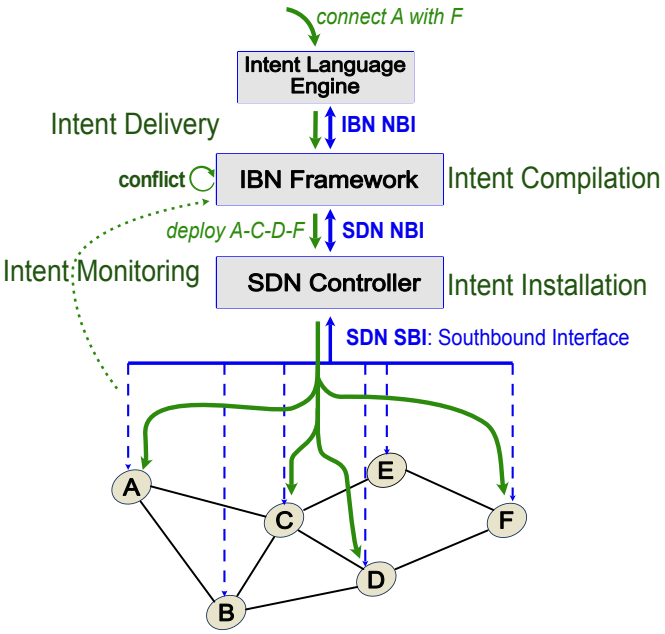


Fig. 2. IBN over SDN architecture. Connection A to F is implemented with the A-C-D-F path.

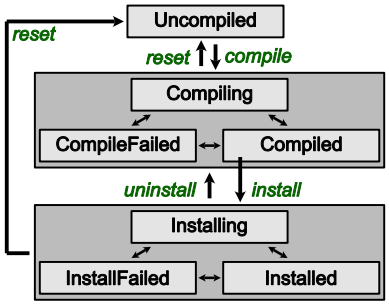


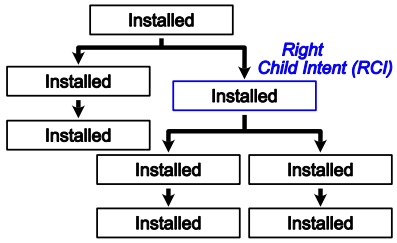
Fig. 3. Intent state machine.

B. Intent tree

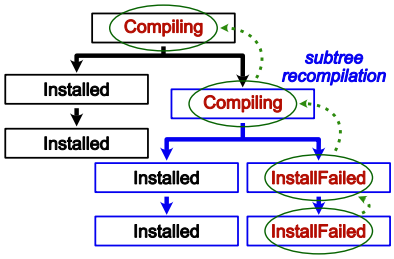
Our work is based on the idea of an intent tree, where the received user intent is the root. Instead of directly compiling the intent to an implementation, we demonstrate a strategy of breaking the problem into a series of subproblems with automatic intent generation. Each intent (problem) can be broken down to system-generated child intents (subproblems) and is considered installed or compiled when all children are so. This triggers updates of the parent's state based on the child's state. For example, an intent will be *Compiled* (*Installed*) only if all the descendants are also *Compiled* (*Installed*). If at least one of the descendants is in *Compiling* (*Installing*) state, this state will propagate to all the ancestors. If at least one of the descendants is in *CompileFailed* (*InstallFailed*) state, then the parent can choose, based on the specifics of the implementation, whether to enter the *CompileFailed* (*InstallFailed*) state or the *Compiling* state. Entering the *CompileFailed* (*InstallFailed*) state means passing on the responsibility for action to the parent intent. On the other hand, entering the *Compiling* state means handling the problem by locally recompiling the subtree defined with this node as a root. An example is shown in Fig. 4.

After intent compilation, we get a hierarchical tree structure of intents, with the intent nodes getting less abstract as we move away from the root. We call the leaves of the tree *low-level intents*. These are device-level intents that serve to request specific resources and can never be in the *Compiling*, *CompileFailed*, and *Installing* states since they have no children and compilation or installation is trivial. Each intent can also have several constraints to tune the desired intent behavior appropriately. For example, constraints can include availability, latency, or bandwidth requirements. With this design of gradually concretizing an intent together with the intent constraints, we can efficiently deal with the abstraction level that an intent system requires [15].

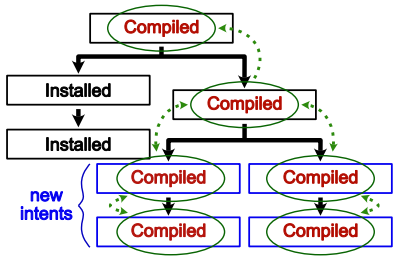
We currently deal with best-effort connectivity intents, which correspond to the Routing and Spectrum Assignment (RSA) problem [16], but similar mechanisms could be built for other intent-driven services. Since RSA is an NP-hard problem, instead of solving it jointly, we split it into (i) the routing and (ii) the spectrum allocation subproblems.We employ a simplified strategy to our intent system, which for every *ConnectivityIntent*, adds a *PathIntent* to solve the routing using k-shortest-path and a *SpectrumIntent* to solve the spectrum allocation using first-fit [17]. More advanced strategies can be incorporated, but it is not the aim of this study.



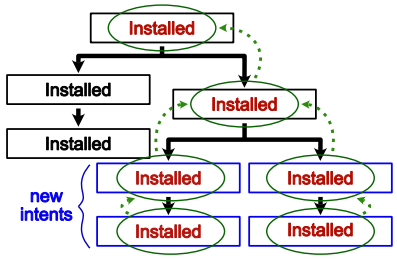
(a) The intent is installed. The Right Child Intent (RCI) is the right child of the root intent.



(b) A network fault caused a low-level intent to fail. Information flows from the low-level intent upwards. Recompilation is triggered in the subtree defined by the RCI. During recompilation, all descendants of the RCI are deleted, and new intents are generated.



(c) The intent is recompiled. The RCI created and triggered the compilation of the new intents that substituted the old ones. The new intents report back whether compilation is successful. The RCI forwards the report to the root intent and later moves to the *Installing* state, signaling the installation of the subtree.



(d) Finally, the intent is reinstalled. Installation status information flows from the low-level intents upwards.

Fig. 4. Intent tree state propagation in case of a network fault.

C. MD IBN

We scale from single-domain IBN to multi-user MD IBN by extending the intent tree to span several domains. We define a special *RemoteIntent*, which delegates an intent to another domain by binding the local intent to a new replica on the remote domain with a parent-child relationship. The

state update properties still hold between the remotely bound intents because of the parent-child relationship. This way, intent states can propagate across different domains. However, the intent tree content is not accessible outside the domain. The parent intent may delete or create child intents, but it can only know their states and nothing about their internal information (e. g., compilation strategy) or descendants. This guarantees confidentiality across the different domains.

The expansion of the intent tree outside a proprietary domain is done using a common IBN NBI. Of course, special permissions and roles must be incorporated into the IBN NBI to support the above operations securely. We define two roles: (i) *admin*, with the ability to set permissions, and (ii) *client* with limited permissions as defined by the admin. The *admin* would usually be the network owner. The *client* is whoever wants to access the network without owning it. Permissions, as roles, apply per IBN framework instance and include adding, compiling, (un)installing, resetting, and deleting an intent. The same user could be an *admin* to a specific IBN framework instance but a *client* to another.

IV. Evaluation

We evaluated the feasibility of our architecture with a proof-of-concept simulation of three decentralized domains and the border nodes being the only shared information. Every domain is directed from a single IBN framework instance, and every IBN instance includes two SDN controllers. We issued numerous connectivity intents under different constraints, and the appropriate permissions were given to all IBN instances to add, compile, and install intents to any other intent-driven domain. All connectivity intents were compiled and installed successfully. We also simulated link faults that caused some low-level intents to transition to the *InstallFailed* state, triggering intent recompilation and reinstallation.

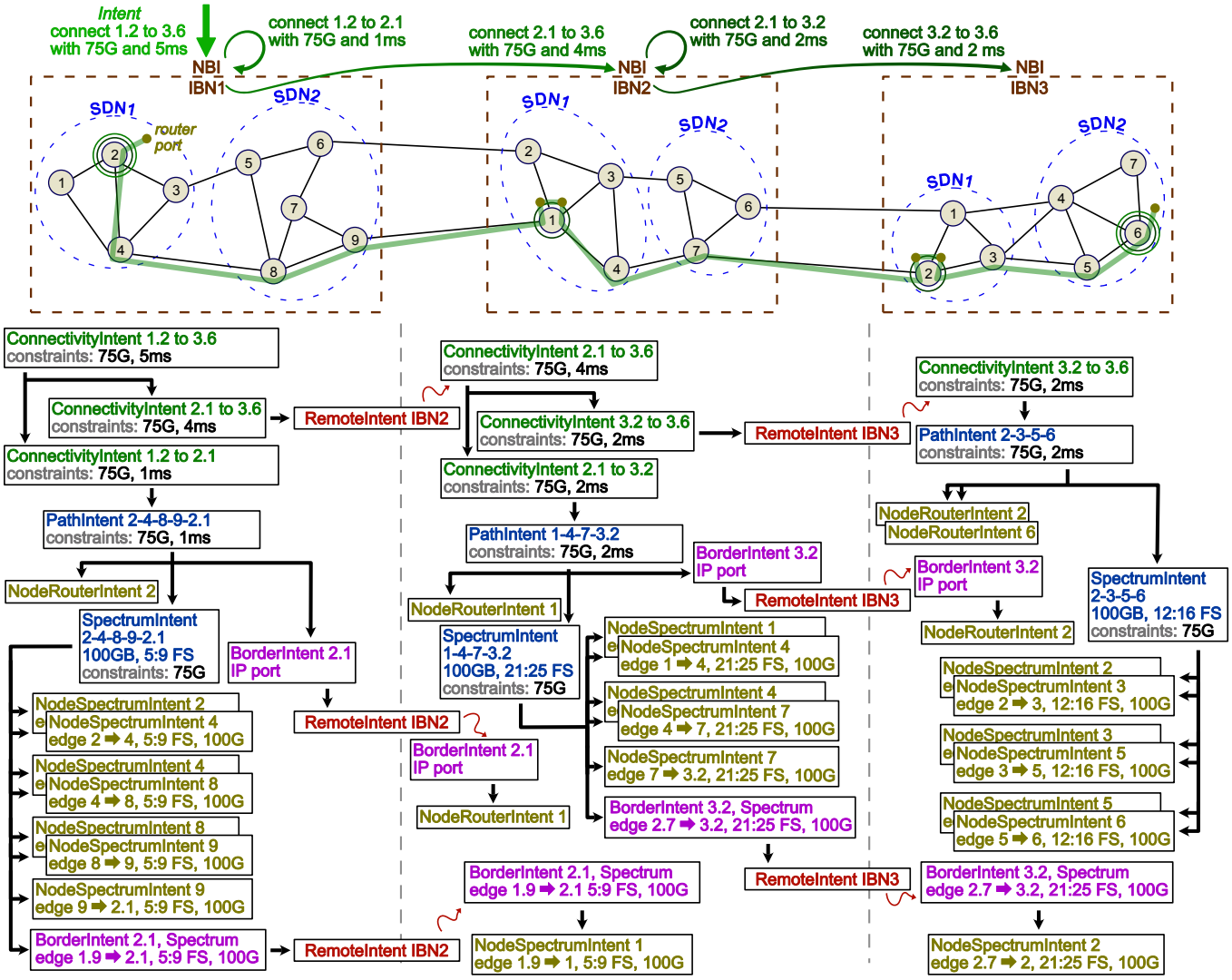


Fig. 5. Decentralized MD IBN intent deployment and the intent trees. Spectrum contiguity and continuity are respected since the same frequency slots (FS) are allocated across the optical connections. The hidden part of each *NodeSpectrumIntent* is the same as its visible pair.

Fig. 5 illustrates an example where the intent trees are generated when issuing to IBN1 a MD *ConnectivityIntent* between nodes 1.2 and 3.6 with $5\,\mathrm{ms}$ latency and $75\,\mathrm{Gbps}$ bandwidth requirements. Node x.y signifies the y-th node of the x-th IBN domain. Overall, the IBN1 compiled the intent by subdividing it into two *ConnectivityIntents*, one implemented locally while the other delegated to the neighboring domain. The current intent compilation strategy performs signal regeneration in the IP layer at every border node, i. e., nodes 2.1 and 3.2. The selection of the border nodes and the neighboring domain is based on the specifics of the deployed implementation algorithm, i. e., the operator's decision-making process.

We observe that *PathIntents* and *SpectrumIntents* compile down to low-level intents, i. e., *NodeRouterIntents* requesting IP router ports and *NodeSpectrumIntent* pairs requesting fiber spectrum slots for each node participating in the link. However, the IBN instance cannot control the neighboring domain for the inter-domain links, and a *BorderIntent* is generated instead, creating remote low-level intents for the border nodes. For example, to use the link between 1.9 and 2.1, the frequency slots $5, 6, 7, 8, 9$ must be allocated at nodes 1.9 and 2.1. IBN1 creates a *NodeSpectrumIntent* for the local node 1.9 and a *BorderIntent* that will issue a *RemoteIntent* to IBN2 for 2.1.

We also notice that constraints are propagated altered to the child intents, depending on whether they are guaranteed to be already (partly) satisfied by the parents or not. For example, the latency constraint of $5\,\mathrm{ms}$ is propagated to one of the child intents as a constraint of $1\,\mathrm{ms}$. This means the parent guarantees that the intent constraint of $5\,\mathrm{ms}$ will be satisfied as long as the child satisfies the intent constraint of $1\,\mathrm{ms}$. We only consider propagation delay here since it is the main contribution of total latency in large-scale high-performance transport networks [18]. The *PathIntent* can decide if the delay constraint is satisfied since it knows the path. If it is satisfied, there is no reason to propagate the constraint further down to the child intents. If it is not satisfied, then the intent state will transition to *CompileFailed*. If it is generally unknown whether the constraint is satisfied, the intent will transfer the constraint to the child intents unaltered.

When all the IBN instances successfully compile and install the system-generated intents, the E2E connection will be available. If one of the IBN instances does not stand up to the requirements of an intent, this will be spotted from the monitoring procedure, which will update the state of the corresponding intent to *InstallFailed*, making it clear whom to hold responsible. Such monitoring promotes accountability and conformity with the intent requirements.

V. Conclusions

We presented an architecture for adopting IBN in MD coordination, and we showed a novel strategy to compile multi-layer connectivity intents using hierarchical system-generated intents. Our approach focuses on scenarios of decentralized control, where each party has partial knowledge of the global network. We underline the importance of a common IBN NBI and hope to inspire future efforts toward standardization.

References

[1] L. M. Contreras and P. Lucente, "Interconnection intents," Internet Engineering Task Force, Internet-Draft draft-contreras-nmrg-interconnection-intents-02, Jan. 2022, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-contreras-nmrg-interconnection-intents-02

[2] D. Espinel Sarmiento, A. Lebre, L. Nussbaum, and A. Chari, "Decentralized SDN control plane for a distributed cloud-edge infrastructure: A survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 1, pp. 256–281, 2021.

[3] S. Alalmaei, Y. Elkhatib, M. Bezahaf, M. Broadbent, and N. Race, "SDN heading north: Towards a declarative intent-based northbound interface," in *2020 16th International Conference on Network and Service Management (CNSM)*, 2020, pp. 1–5.

[4] C. Janz, N. Davis, D. Hood, M. Lemay, D. Lenrow, L. Fengkai, F. Schneider, J. Strassner, and A. Veitch, "Intent NBI - definition and principles," Open Networking Foundation, Tech. Rep. ONF TR-523, October 2016, (visited on 05/05/2022). [Online]. Available: https://opennetworking.org/wp-content/uploads/2014/10/TR-523_Intent_Definition_Principles.pdf

[5] Y. Xia, S. Jiang, T. Zhou, S. Hares, and Y. Zhang, "NEMO (network modeling) language," Internet Engineering Task Force, Internet-Draft draft-xia-sdnrg-nemo-language-04, Apr. 2016, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-xia-sdnrg-nemo-language-04

[6] ONOS. (2016) Intent framework. [Online]. Available: https://wiki.onosproject.org/display/ONOS/Intent+Framework

[7] OpenDaylight. (2021) Network intent composition. [Online]. Available: https://wiki.opendaylight.org/display/ODL/Network+Intent+Composition

[8] L. Velasco, M. Signorelli, O. G. De Dios, C. Papagianni, R. Bifulco, J. J. V. Olmos, S. Pryor, G. Carrozzo, J. Schulz-Zander, M. Bennis, R. Martinez, F. Cugini, C. Salvadori, V. Lefebvre, L. Valcarenghi, and M. Ruiz, "End-to-end intent-based networking," *IEEE Communications Magazine*, vol. 59, no. 10, pp. 106–112, 2021.

[9] A. Rafiq, A. Mehmood, T. Ahmed Khan, K. Abbas, M. Afaq, and W.-C. Song, "Intent-based end-to-end network service orchestration system for multi-platforms," *Sustainability*, vol. 12, no. 7, 2020. [Online]. Available: https://www.mdpi.com/2071-1050/12/7/2782

[10] G. Davoli, W. Cerroni, S. Tomovic, C. Buratti, C. Contoli, and F. Callegati, "Intent-based service management for heterogeneous software-defined infrastructure domains," *International Journal of Network Management*, vol. 29, no. 1, p. e2051, 2019, e2051 nem.2051. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/nem.2051

[11] J. Augé and M. Enguehard, "A network protocol for distributed orchestration using intent-based forwarding," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2019, pp. 718–719.

[12] S. Arezoumand, K. Dzeparoska, H. Bannazadeh, and A. Leon-Garcia, "MD-IDN: Multi-domain intent-driven networking in software-defined infrastructures," in *2017 13th International Conference on Network and Service Management (CNSM)*, 2017, pp. 1–7.

[13] M. Bezahaf, M. P. Hernandez, L. Bardwell, E. Davies, M. Broadbent, D. King, and D. Hutchison, "Self-generated intent-based system," in *2019 10th International Conference on Networks of the Future (NoF)*, 2019, pp. 138–140.

[14] K. Mehmood, K. Kralevska, and D. Palma, "Intent-driven autonomous network and service management in future networks: A structured literature review," *ArXiv*, vol. abs/2108.04560, 2021.

[15] A. Campanella, "Intent based network operations," in *2019 Optical Fiber Communications Conference and Exhibition (OFC)*, 2019, pp. 1–3.

[16] B. C. Chatterjee, N. Sarma, and E. Oki, "Routing and spectrum allocation in elastic optical networks: A tutorial," *IEEE Communications Surveys Tutorials*, vol. 17, no. 3, pp. 1776–1800, 2015.

[17] B. C. Chatterjee and E. Oki, "Performance evaluation of spectrum allocation policies for elastic optical networks," in *2015 17th International Conference on Transparent Optical Networks (ICTON)*, 2015, pp. 1–4.

[18] B.-K. Choi, S. Moon, Z.-L. Zhang, K. Papagiannaki, and C. Diot, "Analysis of point-to-point packet delay in an operational network," in *IEEE INFOCOM 2004*, vol. 3, 2004, pp. 1797–1807 vol.3.