# Intent-based Networks: An Industrial Perspective

**Conference Paper** · October 2018

DOI: 10.1145/3243318.3243324



**4 authors**, including:

Barun Saha
Hitachi ABB Power Grids
**31** PUBLICATIONS **151** CITATIONS
SEE PROFILE

Deepaknath Tandur
ABB
**47** PUBLICATIONS **636** CITATIONS
SEE PROFILE

# Intent-based Networks: An Industrial Perspective

Barun Kumar Saha
ABB Corporate Research Center
Bangalore, India
barun.kumarsaha@in.abb.com

Deepaknath Tandur
ABB Corporate Research Center
Bangalore, India
deepaknath.tandur@in.abb.com

Luca Haab
ABB Power Grids
Bern, Switzerland
luca.haab@ch.abb.com

Łukasz Podleski
ABB Power Grids
Bern, Switzerland
lukasz.podleski@ch.abb.com

## ABSTRACT

Intent-based Networks (IBNs) present a paradigm shift in conventional networking by allowing users to express what is required from a network, while internally resolving the detailed steps necessary to achieve that goal. Consequently, not only human effort, but chances of errors are also reduced. Moreover, once an intended state is achieved, IBN ensures that it is maintained unless a user requests otherwise. In this paper, we present an industrial perspective of IBN. In particular, we identify several use cases in the context of industrial networks that can potentially benefit from IBN. Subsequently, we propose an architecture of IBN and elaborate how the different use cases can be realized using it. In addition, we present a case study on the use of intents with a widely used framework. Although, there still are a few challenges along the IBN way, when addressed, IBN can bring down an organization's expenses while improving its efficiency and reliability at the same time.

## CCS CONCEPTS

• **Networks** → **Network architectures**; **Bridges and switches**; *Network manageability*; Network experimentation; • **Applied computing** → *Industry and manufacturing*;

## KEYWORDS

Intent-based Networks, Software Defined Networks, Industrial Networks, Network Architecture, Intent, Network Management

## 1 INTRODUCTION

Although today's communication networks are growing massively in scale, their operation and management largely involves a manual and intricate process. In contrast, IBN promises a paradigm shift where network users and operators would manage their operations by stating at a high-level "what to do," rather than specifying the low-level details of "how to do." To present a real-life analogy, consider the scenario where a person wishes to travel from Baden to Bangalore. He/she has to book the flight, reserve a hotel room, make travel arrangements to and from the airport, and also take time zone and local traffic considerations into account. Contemporary networks look more like this scenario. In contrast, had this been IBN, the person would have simply asked an agent to arrange a travel on a given date with time and airlines preferences; relevant details would have been worked out by the agent behind the scene. Such an approach can potentially reduce the burden (and chances of errors) of concerned people. It is, therefore, unsurprising that IBN is gaining a lot of attention both in industry and academia. In future, the concept of IBN may as well be extended beyond networks to encompass intent-based systems (IBX), in general.

Be it industrial networks [5, 13] or enterprises, everyone can reap benefits from IBN. On one hand, data center management can be simplified. On the other hand, IBN can ensure that the critical constraints of latency and reliability are met in industrial networks. As we shall see later, IBN technology can help in reducing both capital and operational expenses involved in the network.

Motivated by these aspects, in this paper, we take a look at the contemporary picture of IBN. The primary objectives of this work are to identify important use cases related to industrial networks and to investigate how they can benefit from IBN.

The specific *contributions* of this work are as follows.

- Discussing potential industrial use cases of IBN.
- Proposing an architecture of IBN upon which future implementations can be based.
- Identifying enabling points and future directions for IBN.

The remainder of this paper is organized as follows. Section 2 presents a review of related work on IBN. We discuss an architecture of IBN in Section 3. Some of the potential use cases in the context of industrial networks are identified in Section 4. A short case study on intents is presented in Section 5. Section 6 discusses some of the challenges associated with IBN and how they can be coped with. Finally, Section 7 concludes this work.

## 2 RELATED WORK

Networks have witnessed automation for a long time in one form or other – be it simple shell scripts or advanced services such as, Puppet.[1] SDN [3], on the other hand, has brought forward a huge wave of innovation by simplifying the management of heterogeneous devices. Around the same time, advances in artificial intelligence (AI) and machine learning (ML) techniques has allowed to further simplify and automate network operations in the form of IBN.

[1] https://puppet.com/


*FICN'18, November 2018, New Delhi, India*

Kiran et al. [4] developed the Intelligent Network Deployment Intent Renderer Application (iNDIRA) system to enable intent-based networking. iNDIRA is designed to sit above SDN's control plane and interact via north-bound interfaces (NBIs). iNDIRA uses natural language processing (NLP) to allow users express their intents, which are converted into Resource Description Framework (RDF) graphs before getting translated into low-level commands. Primarily aimed for science and research networks, users can request iNDIRA for a service (for example, *file transfer between two end-points*) with desired constraints (say, 10 Mbps bandwidth); the system either accepts it or informs back to the user why a request could not be placed. Marsico et al. [6], on the other hand, considered the process of application-aware resource negotiation between users and the network. In case of resource unavailability, possible migration of non-critical services to alternative paths were considered in order to accommodate new application requirements.

Arezoumand et al. [2] addressed the problem of applying intents to multi-domain networks, where scalability is a big challenge. To overcome it, the authors took a hierarchical approach. The multi-domain network was represented as a multi-graph; an input intent graph (consisting of all the intents) at first underwent domain-wise decomposition so that all intents were localized within the respective domains and there were no inter-domain edges. In particular, transit domains with shadow end-points were added for non-peers. Subsequently, the decomposed graph was further processed to generate local intent graphs. The authors considered a few different types of intents, for example, virtual L2 WAN and service function chaining. Results indicated that the proposed hierarchical approach can offer about 30 times performance improvement as compared to when the network was considered as flat.

Tsuzaki and Okabe [12] proposed extensions to Network Modeling[2] (NEMO), a domain specific language bearing similarity with the Structured Query Language. While NEMO inherently provides pro-active modeling capability, the authors proposed to augment it with event-condition-action rules to make it reactive. The authors further discussed different use cases, for example, adapting routing path based on bandwidth limits, where intents expressed with reactive aspects can help in automate the network management.

Subramanya et al. [11] discussed the use of IBN as backhaul for 5G networks. In particular, the problem of duplicate packet removal network function was addressed, which becomes important when a given user is associated with multiple access points in a region, for example, a stadium. The authors, in an IBN environment, moved the said function to the backhaul, which also allowed for passing the traffic thorough other virtual network function (VNFs) such as deep packet inspection. The intents (such as, dynamic chaining) were represented with one or more virtual links. Results indicated that without IBN, network throughput fell down frequently and significantly.

Zhang et al. [14] discussed the development of an intent solver that can be used together with SDN controllers. The authors modeled different types of intents – application-related and network management-related – as optimization problems and used the intent solver to solve them. Compared to greedy mechanisms, such an approach allowed for resolving potentially conflicting intents,

[2]https://wiki.opendaylight.org/view/NEMO:Main

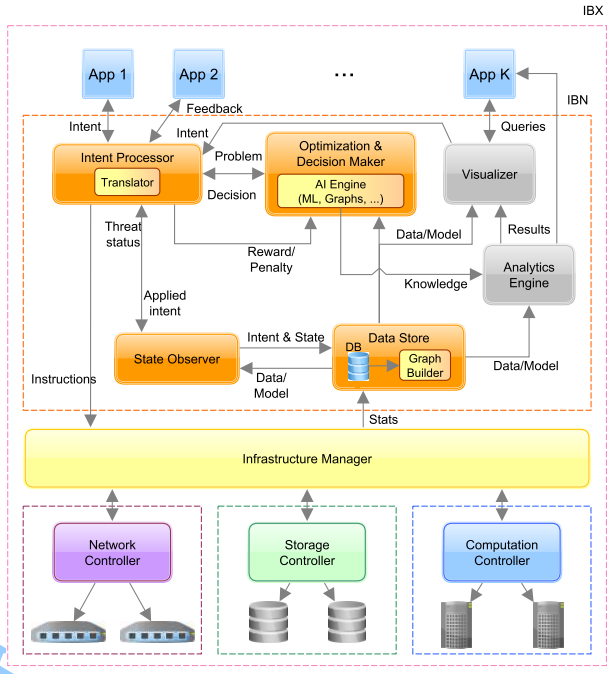



**Figure 1: Architecture of IBN showing its different components. In contrast to the common interpretation of IBN, "IBX" refers to the complete suite of intelligent applications, services, and operations that are supported by the network.**

and consequently, it led to better utilization. Abhashkumar et al. [1] discussed the development of Janus, a system to address a similar problem. In particular, Abhashkumar et al. extended the policy graph abstraction [7] model in order to represent diverse Quality-of-Service (QoS) policies (intents) that are relevant to a network. Such QoS policies are not necessarily static, but can vary with time giving rise to the dynamic nature of the graphs. Once a group of such policies were available, an optimization problem was formulated with the aim of maximizing the number of policies that can be installed in the network taking into account policy and resource constraints; simultaneously, the extent of corresponding path changes was aimed to be minimized.

## 3 AN ARCHITECTURE OF IBN

Before looking at how one can benefit from IBN, we present a general architecture of it, as shown in Fig. 1. It is loosely inspired by the Open Platform for Network Function Virtualization[3] (OPNFV) project. Its different components are discussed below.

**Intent Processor (IP)**: The IP is the core component of IBN/IBX, and it interacts with most of the others. In particular, the IP is responsible for accepting high-level intents expressed by users (applications) using a semantic language or through other appropriate user interfaces. Once an intent is available, the **translator** submodule translates that into a mid-level representation that can be understood by other modules. For example, IP can receive input in plain English or as a mathematical model; in turn, it can translate

[3]https://www.opnfv.org/. It is an ETSI backed project.

that into say, Algebraic Modeling Language, which can be easily understood by a mathematical solver software. When an intent is found to be feasible, IP communicates the corresponding instructions southward toward the devices. IP is also informed of any potential threat to the intended state so that it can take actions, if any, to protect the state.

**Optimization & Decision Maker (ODM)**: ODM is responsible for all decision making in IBN. ODM receives as input a mid-level representation of an intent in the form of, for example, an optimization [10] or decision making [8, 9] problem. After solving, it returns a feasible solution, if any, in the former case and the (relatively) best decision in the latter case. Any such solution or decision is conveyed back to IP. ODM essentially consists of one or more commercial or open source solver(s). The latter may not be physically located inside the network, but residing in an external cloud.

The **AI Engine (AIE)** sub-module of ODM applies AI- and ML-based techniques upon past intents (both installed and rejected), states and statistics to determine whether such actions were appropriate. In the process, IBX gains knowledge of what is correct in a given context and gradually learns to make better decisions in terms of feasibility and intent installation. Optionally, IP incentivizes or penalizes ODM depending upon whether or not a given decision generated by the latter was appropriate. Like ODM, AIE can also be cloud-based. However, in such cases, network latency involved in decision making should be critically tested.

**Data Store (DS)**: The DS acts as a repository of all relevant data generated in and collected from the network. Appropriate access mechanisms are provided to the other components of IBX so that they can store/use data in/from the DS. In particular, the DS is composed of the following two modules.

- **Database (DB)**: All concerned data are physically stored in the DB, which can be centralized or distributed. Various kind of data can be stored in the DB including, but not limited to:
  - Intents that are installed across the network together with representation of the corresponding intended states
  - Periodically collected structured data, for example, link and flow statistics, health monitoring data, and so on
  - Event-triggered unstructured data, for example, topology change information
  - Network inventory data – either entered by operator or discovered automatically
- **Graph Builder (GB)**: The GB module is aimed to provide a graph-view of the underlying network to others. Such a graph model can be used for optimization/decision making, generating analytics, and network visualization. For example, one may use the graph to solve the Traveling Salesman Problem with the aim of sending a message to each switch exactly once. In the absence of GB, the other IBN modules would need to repeatedly construct a graph view of the network by themselves based on data available in the DB. GB also acts as a proxy for a typical topology manager; it continuously updates the underlying graph in response to different events such as link up/down and bandwidth changes.

**State Observer (SO)**: After an intent is decoded and found to be feasible, the IP installs it in the network. Additionally, the SO is informed about the installed intent, which is subsequently stored in the DS. The job of SO is to continuously monitor the network, collect data and store them in DS, and inform IP in case it identifies any potential threat against an already installed intent. Such threats can arise in different contexts, for example:

- Topology change: Consider a user has installed an intent to establish shortest path communication with another endpoint. However, if one or more links along the path go down, the applied intent fails. In general, any change in the existing network topology can potentially threaten one or more intents applied in the network. Based on graph model available from GB, the SO continuously evaluates whether any active intent is violated by topology change(s).
- Link characteristics change: Given an active link, its bandwidth share for an application can decrease due to sudden surge in traffic. This can, too, threaten an intended state, for example, minimum bandwidth requirement for a video call.
- Other changes: A host can be overloaded with service requests leading to failure in serving a critical service. Alternatively, the host can completely fail. Such scenarios may threaten intents related to service provisioning.

**Visualizer**: Primarily, this module has two purposes – 1) to provide a visual outlook of the network, its status, and performance at any point of time using a graphical user interface (GUI) and 2) to allow users to express (some or all) intents via the GUI.

**Analytics Engine (AE)**: AE is responsible to give answers to simple as well as complex queries related to network performance. It can take into consideration raw monitoring data, detailed graph model, and advanced algorithms to generate results, which can be displayed by the Visualizer. Optionally, the AE can provide REST API to allow network monitoring by third-party applications.

It may be noted that the Visualizer and AE modules are non-critical, whereas the others constitute as core components of IBN.

**Apps**: In IBX, the apps are transparent to the underlying system and only interact via suitable Application Programming Interfaces (APIs). The system, in turn, provides the apps with relevant feedback depending upon the context. For example, the IP may inform an app that installation of a particular intent is infeasible at the given moment. Human user(s) in the loop can consider such information and take further decision appropriately.

**Network, Storage, and Computation Controllers (NC, SC, and CC)**: The three controllers are responsible for actually configuring the devices (such as switches, disks, and servers) according to a specified intent. The NC can be a network management system in a traditional network or a controller in the SDN context. Communication between the IP and each of these controllers allows users and operators of an IBN/IBX to efficiently allocate and virtualize resources as well as provision different services.

**Infrastructure Manager (IM)**: The IM[4] acts as an interface between the IP and the different controllers. It is also responsible for translating feasible intents into more "domain-specific" instructions related to the networking, storage, and computation. Consequently, the IP module can be transparent to any underlying controller. For example, IP can process an intent and request IM to allocate 50 GB

[4]The line between network/computation/storage controllers and IM, however, can be blurred. For example, OpenStack is an IM, but at the same time, it is also a storage and computation controller.

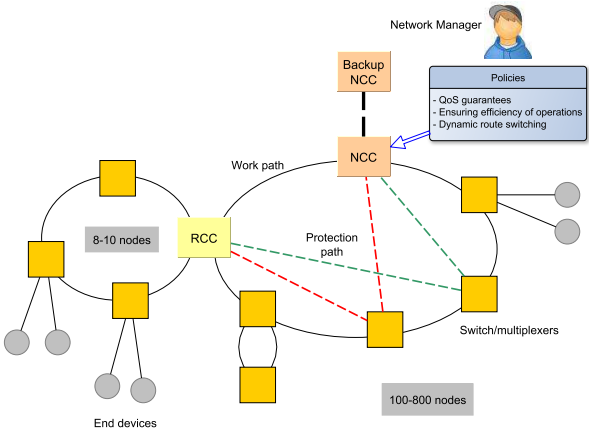



**Figure 2: Illustration of a typical "floral" topology used in utilities networks. The Regional Control Center (RCC) may consist up to 8–10 nodes, whereas the network as a whole may have 100–800 nodes. The National Control Center (NCC) is responsible for controlling the RCCs, which control their respective ring networks.**

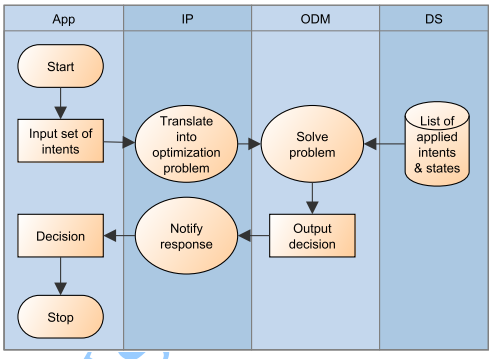



**Figure 3: Workflow to verify whether a given set of intents can be applied to IBN.**

storage, 5 CPU cores, and a P2P line between A and the place where the storage is. IM can process this in two steps – 1) at first, it talks to the storage and computation controllers, respectively, to reserve 50 GB space and 5 CPU cores and 2) it asks the network controller to establish a path. Moreover, IM is responsible for collecting various statistics from the controllers.

## 4 INDUSTRIAL USE CASES

Communication networks are used in various industrial sectors such as utility, oil and gas, mining, and power distribution. In some scenarios, like mining, wireless networks are used, while in many others wired networks is the technology of choice. Fig. 2, for example, shows a typical ring network used in utilities sector. Such networks usually have protection path(s) (could be multi-hop) that are used when the link(s) of the ring fail(s). The concerned network manager is responsible for enforcing various kinds of policies including QoS guarantees and dynamic route adaptation.

Here, we discuss some use cases where IBN can be potentially useful. They span different phases of a network life from bidding to commissioning and operations. We also discuss how these use cases can be realized with respect to the proposed architecture of IBN.

**Bidding and Commissioning**: Much before a network is actually deployed, one has to enter the bidding process and provide a meaningful cost estimate. To make things more challenging, project requirements are often specified briefly and vaguely. For example, a customer may say the he/she requires a network of $m$ devices providing $n$ services out of which $k$ are critical, and so on. IBN can be used here to provide a relatively more accurate price quote in a very short time as compared to a manual process. This is realized by having all the involved Apps submitting their constraints to the IP, which then converts them into an optimization problem. Subsequently, the ODM browses through the inventory of devices (not necessarily only those available in the current network) and their specification in the DS and chooses an optimal set of devices. It may even suggest devices that are not in the database if no sensible solution is found. Based on the cost of optimization, the bidding amount is computed and conveyed back to the App. Similarly, during commissioning, IBN can be used to generate optimum configurations for the devices in a quick and efficient manner, unlike a manual approach. IBN, therefore, can help in bringing down cost and time in both the scenarios.

**Planning/Accommodating Changes**: In an operational network, users and operators may want to introduce various changes. For example, changing the resource allocation in disaster recovery situation, while regularly adding new sensors or simply adding cables. In reality, such requests may often create potential conflicts; IBN can be used to decide whether or not those new flows can be accommodated. In particular, a new flow can be accepted if 1) enough bandwidth is available, 2) is of high priority (say, voice traffic and teleprotection), and 3) other traffic constraints are affected by tolerable measures. Alternatively, it can be accommodated by throttling/dropping low priority flows. IBN can make these optimal decisions without requiring an operator to do complex calculations. A closely related use case is that of planning, where a user merely want to "simulate" the effects of introducing a set of changes. In such a case, the computations have no effect upon the real network . Figure 3 illustrates this process.

**Protecting N:1 Systems**: IBN can play a significant role in the operations phase of a network. For example, in certain applications often networks have a single (pre-computed) protecting path supporting $N$ working paths. When one of these working paths fails, traffic is switched over to the protecting path. However, since there are still potential chances of failure, IBN, taking into account the changes in topology, can compute a new protecting path as backup. It may be noted that the new protecting would be there on a temporary basis. Although, such dynamicity is atypical for highly critical applications, it is useful for less critical systems that can benefit from high availability.

## 5 A CASE STUDY ON INTENTS

Different software environments have been used to evaluate IBN-related ideas discussed in Section 2. Emulation-based approaches

use Mininet[5] to generate the network with multiple switches and hosts. SDN controllers, for example, OpenDaylight[6] (ODL), Open Network Operating System[7] (ONOS), POX, and NOX, are typically used to implement the intent framework. NOX is the original SDN controller; POX is a Python-based implementation of it. However, both NOX and POX seem to be dated in terms of supporting the latest OpenFlow versions. ODL consists of the Network Intent Composition[8] project, where intents are used to express network policies that are communicated to the SDN.

ONOS provides several built-in intents to enable host-to-host contact, point-to-point communication, virtual network creation, and so on. In the following, we present a case study on ONOS' host-to-host intent and look at how it reacts to topology changes. For this purpose, a network of 10 Open vSwitches and two hosts were created using Mininet, as shown in Fig. 4 (a). Next, we enabled communication between the two hosts (identified with their respective MAC addresses) using the following command in ONOS:

```
add-host-intent 52:2F:37:08:44:9E/None
    B6:81:D6:47:14:4D/None
```

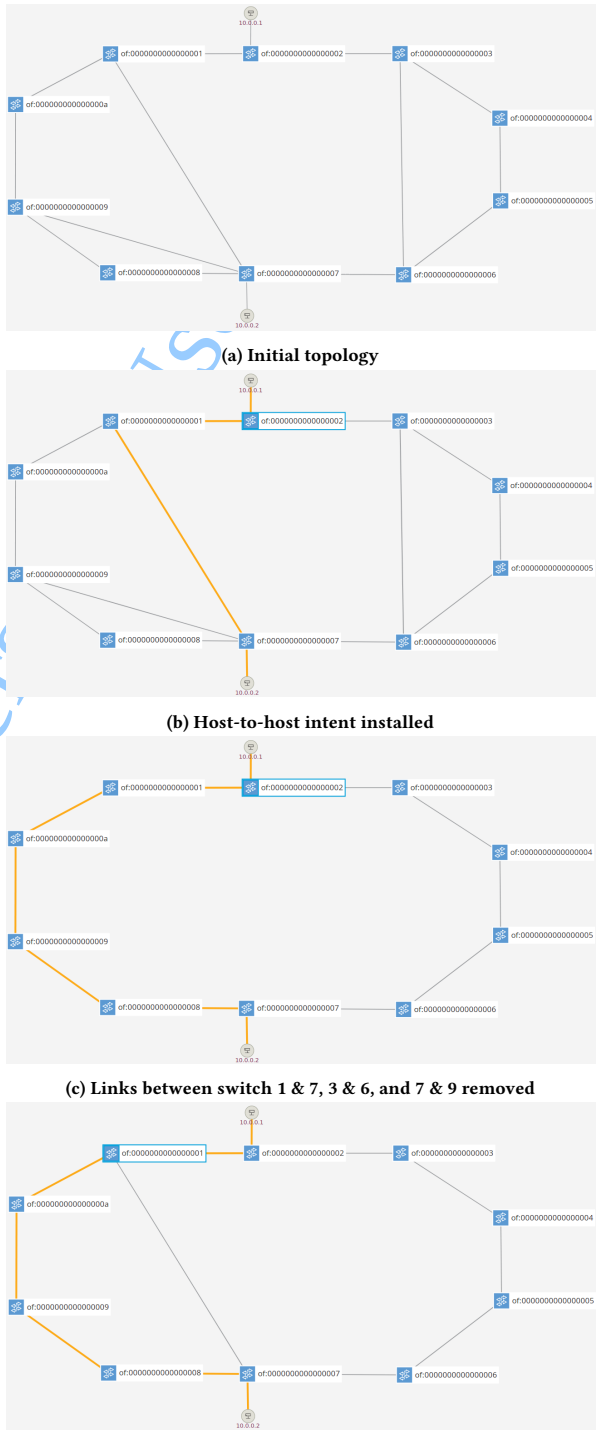
Fig. 4 (b) shows the traffic path (highlighted) between these two hosts under this scenario. Note that the hosts can reach one another via a 4-hop path. Next, we removed the links between switch 1 & 7, 3 & 6, and 7 & 9; Fig. 4 (c) shows the updated path. Finally, we let the link between switch 1 & 7 become active once again, as shown in Fig. 4 (d). Although, a new shorter 4-hop path between the hosts became available, ONOS was found to retain the old longer path. This could possibly be a conscious decision of not disturbing the path (and therefore, the intended state) as long as it is working because of switching overhead. However, when shortest path communication is intended, the concerned intent in ONOS would not be a good choice, particularly when the hop count between the shortest and the next shortest paths differ by a large extent. Nevertheless, this study reveals how powerful intents are, which work even in the absence of any intelligence at the routing and switching layers.

## 6 CHALLENGES AND FUTURE DIRECTIONS

Until now, we looked at the potential of IBN and how it can benefit us under different scenarios. However, are we ready to implement/roll out IBN immediately? Gartner[9] predicted that IBN is unlikely to become mainstream until 2020. In this Section, we look at different challenges that are associated with IBN's large-scale adoption and discuss how some of them can possibly be addressed.

**Language barriers**: One of the key challenges in IBN is the apparent lack of a standard but expressive language for specifying intents. IBNs are always expected to have humans in the loop, and therefore, having a high-level intent to express intents is desirable. Although NEMO and iNDIRA throw light along that direction, their capabilities may not yet be compared to that of naturally spoken languages, for example, English. Moreover, the development of NEMO does not seem to be active in the recent past.

In addition to the above, there is need for having multiple (three) languages. As noted in the architecture, users represent intents at a high-level, ODM processes a mid-level expression, whereas devices understand low-level instructions. Therefore, until a set of suitable



(a) Initial topology

(b) Host-to-host intent installed

(c) Links between switch 1 & 7, 3 & 6, and 7 & 9 removed

(d) Link between switch 1 & 7 restored

**Figure 4: Illustration of host-to-host intent in ONOS. Path between the hosts is updated in response to topology changes.**

[5]http://mininet.org/
[6]https://www.opendaylight.org/
[7]https://onosproject.org/
[8]https://wiki.opendaylight.org/view/Network_Intent_Composition:Main
[9]https://blogs.gartner.com/andrew-lerner/2017/07/11/intent-based-networking-faq/

languages are chosen, efforts on having them translated could not be initiated, which would affect in realizing a fully functional IP.

Independently, we also note that it might be a good idea to begin with in a small-scale. For example, initially one may have a simple GUI where users visually input their intents; serializing (say, using JSON) of such data is simple and can be easily converted into a structure understood by the ODM. Later, each of these layers could be progressively replaced with advanced components.

**Closed-loop verification**: It may be noted that the goal of IBN is not just expressing intents and having them installed. Unlike contemporary networks, IBN also aims to maintain the states enforced in the network by the respective intents. Such a process requires a closed-loop feedback, which has been illustrated in Fig. 1. However, what is meant by a "threat" in this context is yet to be addressed. Moreover, when a threat is perceived, what should be the appropriate response? For example, does the system attempt to fix it by allocating new resources, or does it block new intents and undo some existing ones to remove the threat. Perhaps a simple answer to this may not be possible; the response would depend on the kind of specific situation. In this context, one may classify the threats into potential categories say, mild, medium, and severe, together with prioritizing the intents. Such an approach may give better ideas on how to handle the threats in IBN.

**AI and ML**: It is expected that IBN would greatly benefit from advances in the domains of AI and ML. In particular, a set of service requests can often lead to constrained optimization problems, which can make use of AI-based heuristics to obtain a solution. On the other hand, as an IBN operates over time, it gradually starts to learn by itself. For example, when the network learns that a video call placed by a user was not satisfactory, it may attempt to reserve a stable and higher-capacity path in the future or inform the user that a good quality could not be achieved at that moment. However, there are a few issues involved here. First, a critical requirement for any ML algorithm is data. IBN, therefore, has to operate for some time to have such data available. However, any data collected from the pre-IBN deployment could perhaps be used until that time.

Second, AI can be a double-edged sword. On one hand it can imbibe efficiency; AI can be used for network analysis and intent implementation. On the other hand, it might be difficult to explain why an automated algorithm did something the way it did. This is particularly important for industrial networks, where, in the aftermath of an accident, the company has to provide explanations. Moreover, it is required that AI algorithms do not change/adapt the currently deployed and working networking connections in order to maintain the state of the network. So, in case of a failure, currently working connections should not be impacted while AI tries to fix the failure. Nevertheless, IBN will still retain human in the loop, which will help in verifying an action before it is actually taken. Additionally, it may be interesting to either have an AI-assisted procedure for humans or use some other technologies to track decisions.

**Legacy networks**: Industries – and enterprises, to some extent – often have legacy network equipment and systems. The aspects of bringing them under the IBN umbrella also need to be considered. In fact, one needs to look at integrating other networks as well, for example, SDN, MPLS, and MPLS-TP. The exact nature of communication would depend on the specific controller/network management system in use.

## 7 CONCLUSION

IBN is expected to change the way how networks are operated and managed. On one hand, the use of high-level intents would simplify its handling. On the other hand, integration of AI and ML algorithms would make it more efficient and robust. In this work, we proposed an architecture that encompasses these different elements. We also discussed several use cases of IBN and illustrated a workflow based on the proposed architecture. Finally, we also shed light on IBN's readiness for adoption and challenges that may be faced toward that goal.

In future, the scope proposed architecture can be further extended. In particular, as in ONOS, one would need to formulate the state machine of intents by capturing all transitions. Additionally, more use cases should be identified for wide acceptability. Finally, it needs to be evaluated how well the proposed architecture can perform in reality.

## REFERENCES


[1] A Abhashkumar, J.-M Kang, S Banerjee, A Akella, Y Zhang, and W Wu. 2017. Supporting Diverse Dynamic Intent-based Policies Using Janus. In *Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT)*. ACM, New York, NY, USA, 296–309.

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

[3] N Feamster, J Rexford, and E Zegura. 2014. The Road to SDN: An Intellectual History of Programmable Networks. *SIGCOMM Comput. Commun. Rev.* 44, 2 (April 2014), 87–98.

[4] M Kiran, E Pouyoul, A Mercian, B Tierney, C Guok, and I Monga. 2018. Enabling intent to configure scientific networks for high performance demands. *Future Generation Computer Systems* 79 (2018), 205–214.

[5] R Kumar, D Tandur, and M Kande. 2013. Performance evaluation of Field Device Integration (FDI) framework. In *2013 International Conference on Control, Automation and Information Sciences (ICCAIS)*. 178–183.

[6] A Marsico, M Santuari, M Savi, D Siracusa, A Ghafoor, S Junique, and P Skoldstrom. 2017. An interactive intent-based negotiation scheme for application-centric networks. In *2017 IEEE Conference on Network Softwarization (NetSoft)*. 1–2.

[7] C Prakash, J Lee, Y Turner, J.-M Kang, A Akella, S Banerjee, C Clark, Y Ma, P Sharma, and Y Zhang. 2015. PGA: Using Graphs to Express and Automatically Reconcile Network Policies. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication (SIGCOMM '15)*. ACM, New York, NY, USA, 29–42.

[8] B. K Saha and S Misra. 2016. Named Content Searching in Opportunistic Mobile Networks. *IEEE Communications Letters* 20, 10 (Oct 2016), 2067–2070.

[9] B. K Saha, S Misra, and S Pal. 2016. Utility-Based Exploration for Performance Enhancement in Opportunistic Mobile Networks. *IEEE Trans. Comput.* 65, 4 (April 2016), 1310–1322.

[10] B. K Saha, S Misra, and S Pal. 2017. SeeR: Simulated Annealing-Based Routing in Opportunistic Mobile Networks. *IEEE Transactions on Mobile Computing* 16, 10 (Oct 2017), 2876–2888.

[11] T Subramanya, R Riggio, and T Rasheed. 2016. Intent-based mobile backhauling for 5G networks. In *2016 12th International Conference on Network and Service Management (CNSM)*. 348–352.

[12] Y Tsuzaki and Y Okabe. 2017. Reactive configuration updating for Intent-Based Networking. In *2017 International Conference on Information Networking (ICOIN)*. 97–102.

[13] A Varghese and D Tandur. 2014. Wireless requirements and challenges in Industry 4.0. In *2014 International Conference on Contemporary Computing and Informatics (IC3I)*. 634–638.

[14] H Zhang, Y Wang, X Qi, W Xu, T Peng, and S Liu. 2017. Demo abstract: An intent solver for enabling intent-based SDN. In *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 968–969.