Visually-Represented, Intent-Based, Voice-Assisted Networking (VIVoNet)

Amar Chaudhari, Amrita Asthana
Atharva Kaluskar, Lakshay Karani
Sapna Gangwar
Interdisciplinary Telecom Program
University of Colorado Boulder
amar.chaudhari@colorado.edu

Dr. Levi Perigo
Interdisciplinary Telecom Program
University of Colorado Boulder
bettina.gluzman@facebook.com

Abstract—In the era of rapid digitization, the size of computer networks is increasing to a large and nearly unmanageable scale. The configuration, operation, monitoring, and troubleshooting of such networks is a cumbersome and time-consuming task for network administrators as they have to deal with the physical layer, underlying protocols, addressing systems, control rules and many other low-level details. This research paper proposes an Intent-Based Networking System (IBNS) with voice assistance that could abstract the underlying network infrastructure and allow the administrators to alter its behavior by merely expressing their intents via voice commands. This system will also display the real-time topology along with highlighted intents on an interactive web application which can be used for network diagnostics. It would be difficult to integrate a voice assistant in a traditional network environment and accomplish this functionality. However, Software-Defined Networking (SDN) makes it easier to configure the network behavior based on the intents received via voice commands and observe the changes in the path taken by network traffic via a user-friendly web interface.

Keywords- Alexa, Django, Floodlight, Intent-Based Networking, MariaDB, Nginx, OpenFlow, Real-time Visualization, ReST API, SDN.

TABLE OF CONTENTS
1) INTRODUCTION.............................................1
2) RELATED WORK...........................................1
3) SYSTEM SETUP..............................................3
4) RESEARCH TESTS AND RESULTS...........5
5) CONCLUSION...............................................6
6) FUTURE WORK.............................................6
7) ACKNOWLEDGMENT........................................6
8) REFERENCES..................................................6

I. INTRODUCTION

An intent can be defined as a high-level policy which determines the behavior of a network. An intent is ‘what’ the user wants from the network, without worrying about ‘how’ it is done. The time required to change such an intent in a traditional network is substantial. To overcome this, an Intent-Based Networking System (IBNS) allows the engineers to make the network infrastructure agile in limited time using Software-Defined networking [1]. An IBNS is a system that enables the administrator to define and implement the aim, behavior, and the policies enforced on a given network. In such a system, there is a platform to configure, monitor, and troubleshoot networks by altering its behavior without diving deep into the underlying networking technology. Such an environment can reduce human interaction with the system minimizing the chance of human-inflicted errors consequently saving time and money [2].

The VIVoNet system goes one step further by integrating IBNS with voice assistance and interactive web GUI. It comprises of four major components: SDN infrastructure with Floodlight controller [3], Amazon-Alexa as a smart Voice-Assistant, an Intent Engine, and an interactive Graphical User Interface (GUI). This integrated system enables the user to provide an intent such as least latency, least hop-count or highest bandwidth path between two locations and change the network traffic flow accordingly. For an organization with sites at multiple geographical locations, a system like VIVoNet can be efficient in finding the best path based on the user intent. This system enables the network administrators to delineate, build a state, and alter the network behavior via an automated management platform. The newness of this system exists in the voice-assistance capability and the intent engine application to compute and push flows into the network nodes so that the traffic follows a particular path between the two locations. The individual components of this system interact at various stages to process the intent and optimize the overall process of configuring flows in the network nodes. The execution of an IBNS with voice evades the manual configuration of network nodes, thereby, diminishing the possibility of human error. The real-time network visualization on the GUI is an aid to monitor and manage the network. With the inevitable growth of network size and dominance of smart assistants, the VIVoNet system can be an effective tool for network administration in the coming future. As a result of this research, the VIVoNet system has been developed comprising of Alexa, IBNS and web GUI.

II. RELATED WORK

A. Voice-Assisted Automation

Spoken Dialogue Systems (SDS) are intelligent agents that help users via spoken interaction to complete a task more...
efficiently [4]. SDS are being integrated into a variety of devices such as smart-phones, smart televisions, smart refrigerators and car navigation systems [5]. SDS can support numerous applications in enterprises, education, healthcare, entertainment, and government [4].

Amruta S et al. has proposed a wireless home automation system controlled by a computer. The proposed system automatically translates spoken words into text commands in MATLAB. The text commands are transmitted to a microcontroller which controls the household appliances via its corresponding relays [6].

Anoja R et al. has proposed a system to connect and control Internet of Things (IoT) devices using voice. The proposed system utilizes Alexa Voice Service to develop a customized skill that is used to connect IoT devices and publish them. The project utilizes Amazon Web Services (AWS) IoT service as a centralized management platform for various IoT devices and AWS Lambda triggers to process voice commands from the user. The project provides a use case of controlling IoT devices with voice using Alexa and suggests possibility of similar implementations for networking, home automation, and data monitoring [7].

The above efforts suggest a prospect of enhancing system usability by integrating voice assistance. This project utilizes Amazon Alexa which has the intelligence to detect user intents from the voice input. The VIVoNet system takes advantage of this smart voice-assistance to manage a company’s network.

B. Network Management

The Clean Slate project led by Nick McKeown at Stanford university discusses SDN as a new paradigm of networking which separates the control plane from the data plane [8]. This makes routing less complex and more manageable as the controller acts as the brain of the network and the devices do not need to perform computation. This allows the controller to have a complete view of the network and the administrator can alter the network from a single point.

S. Agarwal et al. also did an experiment in a setup consisting of an SDN and a traditional network. The SDN forwarding elements give the information of link state to the controller and the rest of the nodes follow hop-by-hop forwarding. As a result of their experiment, they found that network performance was improved and overall throughput was increased by adding only a few SDN forwarding elements along with the traditional network [9].

The above literature helped in understanding the importance of SDN to simplify the network management process. The VIVoNet system is designed to operate in an SDN environment to utilize features such as programmability, less complex routing, and centralized control to improve the network management.

C. Visual Representation

Due to the decentralized nature of traditional networks, dynamic visualization and monitoring becomes difficult. The concept of SDN allows the administrator to monitor and configure the network in a centralized and an automated manner. P. H. Isolani et al. have addressed the benefits of visualizing and understanding the network topology, amount, and the actual path of a network flow in an SDN environment [10]. Y. Watashiba’s research mainly aims to implement a visualization tool that allows network administrators to monitor and troubleshoot an actual SDN [11].

These research papers were useful in building an interactive real-time visualization of the network infrastructure which shows the traffic flow of an intent configured by the user. This would allow the network administrator to view the traffic flow on the web application and make educated routing decisions.

D. Intent-Based Networking System

Han et al. construed the objectives of an IBNS and the roadblocks to its development. They discussed intent composition, conflict checking, mapping and installation. They also reviewed REST API, ONOS SDN controller and express high-level requirement specifications. The study of this paper was important to understand the major hindrances that could arise while designing an IBNS. Also, the interaction of ONOS with REST was helpful in designing APIs for VIVoNet [12].

Cohen et al. discussed two core components of an IBNS - abstraction from the underlying vendor-dependent infrastructure and enforcement of specific behavior on the network infrastructure. They explained in detail the abstraction architectures, policies, and connectivity in an IBNS. This paper helped in understanding how a typical IBNS imposes intents onto the infrastructure and the connectivity between the two [13].

Cisco published a white paper and open source solutions for intent-based networking and a detailed interpretation of the concept. They discuss the advantages of an intent-based system - speed, agility, and reduction in human error. They construe the working of such a system which involves three steps:

- Translation: Comprehension of required business intent and its translation into code.
- Activation: Deployment of the intent at the required places.
- Assurance: Logging, monitoring and alerting.

These solutions were helpful in the comprehension of the goals of an IBNS and the typical stages in the implementation of an IBNS in the VIVoNet system. [14].

E. Research Novelty

After a study of the available literature, no paper was found which addresses network automation using voice assistance. This research involves the amalgamation of voice assistance and intent-based networking, which is a novel approach to control and manage network behavior. Moreover, there is little research and limited industry effort on an intent-based networking systems supported by voice assistance and real-time visualization.
III. SYSTEM SETUP

To build the IViVoNet system, a testbed was created which simulates a typical backbone network of a company with Point of Presence (POPs) in multiple U.S. cities (Refer Figure 1).

![Fig. 1. System setup](image)

All components of IViVoNet are virtualized on physical machines. The VMware ESXi hypervisor is used for virtualization. A Dell PowerEdge R430 machine is used in the design that consists of 12 core E5-2603 CPU, 32 GB RAM, and 400 GB of local storage. All VMs in the setup consist of 2 vCPUs, 2 GB of RAM, and 20 GB of storage.

The backbone network contains virtual switches as device nodes (P routers). These virtual switches are open-source Open vSwitch (OvS) (2.5.1) software with OpenFlow 1.3 [15] running on the servers. The IViVoNet system consists of a centralized SDN controller, Floodlight, running on a separate server that manages all the Open vSwitches [3]. The POPs that connect to the virtual devices are FRRouting (v4.0) instances. The POPs are connected in a full-mesh running eBGP and exchanging assigned prefixes.

IViVoNet system is developed using the Python programming language. Django (1.11.10) framework [16] has been extensively used to develop the front-end and the REST APIs. The Intent Engine and Speech Recognition System are Python-based modules developed for the IViVoNet system. The web server is an Nginx instance [17] that acts as a reverse proxy by accepting HTTP requests from the users and Alexa and forwards them to the Django application via Web Server Gateway Interface (WSGI). The database consists of an open-source MariaDB (10.3) [17] software running on a server.

To feed voice input to the system, Amazon Echo Dot (2nd Generation) was used. The Echo Dot was connected to a Wi-Fi network that has access to Amazon-Alexa Service via the Internet. A MacBook Pro running the latest Google Chrome browser was used to access the Web GUI via the Internet. DUO security service was used to provide two-factor authentication.

The following four components were developed to build the IViVoNet system which work in tandem to produce the desired results.

- Speech Recognition
- Infrastructure
- Intent Processing Engine
- Visual Representation

The administrator can direct the network behavior via voice commands. The Speech Recognition system accepts these voice commands via Amazon’s smart assistant Alexa. Amazon allows developers to extend Alexa’s capabilities by defining custom skills [19]. A custom skill called ‘IViVoNet’ has been created using the Alexa Skills Kit (ASK). The IViVoNet skill is a collection of intents, sample utterances to invoke those intents and slot types which help in getting more information from the user.

Figure 2 shows the flow of Alexa’s interaction with the user and the IViVoNet system. The user will first say a command to launch IViVoNet such as “Alexa, launch IViVoNet.” Alexa will analyze the user utterance and identify the corresponding intent (CreateIntent). Alexa will then continue the conversation in a funnel format with the aim of filling the configured slot values by asking additional questions.

![Fig. 2. Alexa skill flow](image)

Table I shows the utterances that will result in invocation of a particular intent. For example, if the user says “launch IViVoNet”, Alexa will first convert the speech into text and then match the text with all the sample utterances of the configured intents. If there is a match, Alexa will try to match user inputs with the slot-types and send a POST request to the API endpoint (vivonet/ask/alexa) with these slot-type values and the invoked intent.
Table I
SAMPLE UTTERANCES TO Invoke INTENTS

<table>
<thead>
<tr>
<th>Intents</th>
<th>Sample Utterances</th>
</tr>
</thead>
<tbody>
<tr>
<td>LaunchRequest</td>
<td>launch vivonet</td>
</tr>
</tbody>
</table>
| CreateIntent    | • vivonet setup a {intent_type} path from {from_city} to {to_city}  
                | • {confirmation}                                        |
                | • {intent_type}                                         |
                | • {from_city} to {to_city}                              |

Table II shows the configured slot-types used by the VIVoNet Skill. The {intent_type} is a custom slot-type with allowed values of least hop-count, least latency, and high bandwidth. Similarly, {from_city} and {to_city} utilize Amazon’s in-built slot-types of AMAZON.City which can detect any city name worldwide.

Table II
INTENT SCHEMA AND SLOT TYPES

<table>
<thead>
<tr>
<th>Slots Name</th>
<th>Slots Type</th>
<th>Sample Slots Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>{intent_type}</td>
<td>intent_type</td>
<td>least hopcount, least latency, high bandwidth</td>
</tr>
<tr>
<td>{from_city}</td>
<td>AMAZON.City</td>
<td>Denver, San Francisco (any U.S city)</td>
</tr>
<tr>
<td>{to_city}</td>
<td>AMAZON.City</td>
<td>New York, Chicago (any U.S city)</td>
</tr>
<tr>
<td>{confirmation}</td>
<td>confirmation</td>
<td>Yes/No</td>
</tr>
</tbody>
</table>

In the VIVoNet system, the Intent Engine is a Python application responsible for converting user-requested intents into translated network configuration. It creates appropriate ‘flows’, and the controller pushes them out to the switches. The switches are just dumb devices forwarding traffic depending on flows. Flows are directions from the controller to forward traffic that matches the required criteria. The controller talks to the applications via the Northbound Interface and to the switches using OpenFlow Protocol via the Southbound Interface. Each connection between a switch and a controller is called an OpenFlow channel. These OpenFlow channels have been encrypted using SSL. For persistent storage of end-locations, prefixes, live intents, and existing flows, a database is maintained on the front-end of the system. As soon as the flows are pushed to the infrastructure, the database is updated with the corresponding values. The Visual Representation application renders the network topology via REST API, retrieves the intents from the database, and highlights the paths as per the required intents.

Figure 3 shows the process flow of the Intent Engine. The Speech Recognition System provides the required intent, the source and the destination city to the Intent Engine. The Intent Engine retrieves the switch Data Path Identifiers (DPID) from the controller via REST API. It finds all available paths for the two DPIDs by querying the controller again. Taking the intent into consideration, it computes the best path amongst the available ones. It then constructs flows based on parameters like switch DPID, the input and output port, source prefix, and destination prefix. After constructing these flows for all the switches selected in the best path, it pushes them to the respective switches and verifies if the push operation was successful. If it was, it writes the intent and the flows to the database from where the Visualization System retrieves them and displays on the web interface. The final step in intent processing is to return a value to the speech recognition system; True for success and False for failure.

There are three intents supported by VIVoNet:
- Least Latency: Set the path with the minimum delay from source to destination.
- High Bandwidth: Set the path with the maximum available bandwidth between source and destination.
- Least Hop Count: Set the shortest path between source and destination based on hops.

The web interface of the VIVoNet system provides a real-time view of the entire network infrastructure. It uses REST API to extract the network topology in a JSON-based format. With the help of VIS.JS, a dynamic browser-based visualization library, the JSON-based network graph is converted into an aesthetic real-time network topology [20]. The Intent
Engine stores the configured path for the user specified intent in the database. The web application extracts the configured path using internal APIs and is populated on the web application. The user can then select any particular intent from the generated drop-down and view the highlighted intent path in real-time. The visual display of highlighted intents simply aids the user to verify and view the traffic flow based on the configured intents.

The web application also provides a fail-over scenario for specifying text-based intents as shown in the Figure 4 in case the Alexa service is not running. The user will specify the intent type, source, and destination in the drop-down menu. The selected values are then converted into a JSON-based format which simply emulates the voice-based command for the Alexa service. This JSON input is sent as a POST request to the Alexa service which invokes the particular skill and, the intent is configured on the network topology.

Since VIVoNet has the potential to control traffic engineering in the network infrastructure, it is important to protect the system from malicious users. The web application provides restricted access only to authorized users and network administrators to overcome security-related issues for the scenarios mentioned above. The users accessing the system must be authenticated using a login page before they can get into the system. The web application will validate the user credentials with the credentials in the authentication database. The users can then view the network topology information and modify the intents only if their credentials are successfully validated. We have also incorporated two-factor authentication to validate users when they request intents via voice. The OpenFlow channels have been encrypted with SSL to prevent eavesdropping.

IV. RESEARCH TESTS AND RESULTS

Four tests were performed to validate the voice-assistance and security features of the VIVoNet system by issuing voice commands to Alexa. The results of the four tests cases are shared in this section. Table III gives a summary of the test cases conducted and the results received.

<table>
<thead>
<tr>
<th>Test No.</th>
<th>Test Name</th>
<th>Test Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Configure an intent between two different locations.</td>
<td>Configured and path highlighted on the GUI.</td>
</tr>
<tr>
<td>2.</td>
<td>Configure an intent with the same source and destination city.</td>
<td>Not configured. Voice assistant will give feedback to the user</td>
</tr>
<tr>
<td>3.</td>
<td>Unknown user trying to configure the Intent.</td>
<td>Two-factor authentication via DUO Security is unsuccessful, hence intent is not configured</td>
</tr>
<tr>
<td>4.</td>
<td>Configure an intent between the two locations which already have an existing intent.</td>
<td>Not implemented in VIVoNet but possible.</td>
</tr>
</tbody>
</table>

Test Results Interpretation:

1) The configured intent and highlighted path is displayed on the GUI after the user expresses the intent and locations. This result implies that Alexa is able to invoke the correct skill. The intent engine processed the requested intent and OvS had the correct flows configured. The REST API calls were successful to provide data for visual representation.

2) When a user tried to configure an intent between the same source and destination location, Alexa gave a feedback that “Please give different source and destination cities.” To calculate the least hop-count, least latency or high bandwidth, at least one hop should be present between the locations. By giving the same location as source and destination, intent engine did not call the controller to push the flows, hence intent was not configured.

3) In the process of configuring an intent path, the DUO Security application sent a two-factor authentication code on the registered user’s mobile. For an unauthorized user, the verification was incomplete and Alexa terminated the process of intent configuration in the network. The user with authorization has the capability to approve or reject the request. This security feature bars any malicious network configuration attempt.

4) The user was unsuccessful to configure an intent path between two locations which had an existing intent. The VIVoNet system is scoped to allow only one intent configuration between a pair of locations. The flows have been pushed in the OvS based on the previous intent.

Test Summary: In case of a successful interaction amongst the different components of the VIVoNet system, the frontend obtained the values from the database using REST API calls and displayed the configured intent on the GUI with
a highlighted path as shown in figure 5 and in case of any discrepancy in the security or wrong input, intent configuration and its visual representation was unsuccessful and this is communicated via Alexa.

![Final result displayed on GUI](image)

The other results of this system are:

**Time Saving:** In today’s hyper-scale networks, agility is of utmost importance. Optimizing the time for deployments, scaling and debugging is critical. This makes time an invaluable resource for any enterprise.

On an average, the VIVoNet system required 0.016 seconds to create, push and verify flows on one OvS device for one intent. Alexa has a hard-timeout value of 8 seconds per request/response. The VIVoNet system has four request/response pairs. Therefore, the conversation with Alexa takes a maximum of 32 seconds. Hence, this research estimates a total time of 32.08 seconds to configure a network of five OvS devices via voice commands. Comparatively, the research conducted by Sourav J. et al. concluded that it takes approximately 260 seconds to manually configure a network consisting of five devices [21]. This project, by leveraging the power of automation and SDN, saves a considerable amount of time and effort as compared to networks that depend on RSVP-signaled LSPs for traffic engineering [22].

**Capital Efficient:** Most of the components used in the VIVoNet system are free and open source which reduces the cost of the overall solution and increases its market value. However, there are some vendor specific controllers also available in the market which can be used to setup an IBNS system. The price comparison for the controllers is given in Table IV:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost /year</td>
<td>$0</td>
<td>$1700</td>
<td>$40,293</td>
</tr>
</tbody>
</table>

**Flexibility and Automation:** SDN has various advantages that can be leveraged to automate management of large scale networks. SDN has a centralized management plane which can reduce overhead and downtime [25]. Traditionally, MPLS/RSVP-TE along with DSCP values have been used to segregate traffic. However, DSCP provides limited marking capabilities. VIVoNet system utilizes OpenFlow which provides extensive 40-tuple matching fields [26]. This type of granular traffic marking has been desired by the industry for several years but was not available [22].

**V. CONCLUSION**

In this research project, a novel state-of-the-art intent-based networking system using open-source solutions was developed that can apply intents into a large orchestrated network via voice commands with real-time visualization. With the technology pacing towards the voice and smart assistants, this system has the potential to simplify and automate network configurations, operations, and troubleshooting considerably. Owing to its voice assistance feature, the VIVoNet system has the capability to facilitate engineers with physical disabilities in network administration.

A user-friendly, precise, and appealing front-end was developed with a rich set of features that can give the administrator an insight into the network topology, visualization of various live intents, and modular information about the network infrastructure. Security was inculcated by implementing SSL connections in the software-defined infrastructure and introducing two-factor and login authentication.

**VI. FUTURE WORK**

This paper is a prototype for a traffic-based intent networking system. This system has a wide range of applications that could simplify network behavior patterns. This system could be used to:
- Orchestrate networks via Zero Touch Provisioning with voice.
- Retrieve infrastructure health metrics like CPU utilization, memory, link flapping, etc.
- Network monitoring by setting alarms.
- Traffic segregation at the Application Layer.

After adding scalability, maintainability, and monitoring, VIVoNet has plenty of potential to be of great assistance in managing the network of a large company.

**VII. ACKNOWLEDGMENT**

The authors would like to acknowledge the contribution of Dr. Levi Perigo for his constant support as the academic advisor and Dr. Kevin Gifford for his guidance as the course instructor. This research had valuable inputs from Ms. Bettina Gluzman, the industry advisor from Facebook.

**REFERENCES**


