University of Colorado
Interdisciplinary Telecom Program
TLEN 5700

Concept Definition Document

VIVoNet

Visually Represented, Intent-Based, Voice-Assisted Networking

1.0 Information

1.1 Approvals

<table>
<thead>
<tr>
<th>Name</th>
<th>Affiliation</th>
<th>Approved</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course Coordination</td>
<td>Dr. Kevin Gifford</td>
<td>CU Boulder ITP</td>
<td></td>
</tr>
</tbody>
</table>

1.2 Project Customers

Name: Dr. Levi Perigo
Address: University of Colorado Interdisciplinary Telecom Program, ECOT 312
Phone: (303) 735-5131
Email: Levi.Perigo@Colorado.Edu

1.3 Team Members

<table>
<thead>
<tr>
<th>Name: Amar Chaudhari</th>
<th>Phone: 720-292-3307</th>
<th>Email: <a href="mailto:Amar.Chaudhari@Colorado.Edu">Amar.Chaudhari@Colorado.Edu</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name: Amrita Asthana</td>
<td>Phone: 720-938-9617</td>
<td>Email: <a href="mailto:Amrita.Asthana@Colorado.Edu">Amrita.Asthana@Colorado.Edu</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name: Atharva Kaluskar</th>
<th>Phone: 720-257-3137</th>
<th>Email: <a href="mailto:Atharva.Kaluskar@Colorado.Edu">Atharva.Kaluskar@Colorado.Edu</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name: Lakshay Karani</td>
<td>Phone: 720-299-0025</td>
<td>Email: <a href="mailto:Lakshay.Karani@Colorado.Edu">Lakshay.Karani@Colorado.Edu</a></td>
</tr>
</tbody>
</table>

Name: Sapna Gangwar
Phone: 720-755-8119
Email: Sapna.Gangwar@Colorado.Edu
2.0 Project Description

The growth of network architecture in today's corporate and public domain is happening at a rapid pace. As the complexity of network design is getting more intricate, network management has become increasingly strenuous. Application overhead is on the rise, with multiple systems running on top of each other to administer massive networks. Device configuration and monitoring has also become tedious with increase in size. This calls for a smart application which can manage a network based on the administrator's requirements in an autonomous manner, without human intervention.

Today, the world is moving swiftly towards a generation of smart assistants with Apple, Google, Amazon, and Microsoft launching digital assistants to simplify day-to-day chores and let the machines do the job for people. The trend is to abstract the intricate details of the underlying technology and make it as user friendly as possible. VIVONet will be based on same concept to simplify network management.

VIVoNet is a system which shall be used to control an orchestrated network via voice commands. The system will accept instructions via mobile or home automation devices and process them to produce the desired output. Our application will visually represent the network architecture and the real-time transitions in the network based on the commands directed by the user. The voice APIs can be used to provide explicit intents and automate network functions.

The system shall provide an automated, flexible and scalable network. Minimal manual intervention will save time to configure the network, consequently making it economically beneficial. The Intent-Based networking system shall simplify operational tasks, improve the network availability and agility.

Level 1 – Network Infrastructure Design

The network infrastructure of VIVoNet shall comprise of a Software Defined Network (SDN) controller and a series of white-box switches running an SDN southbound protocol for communication between them. The application shall be running over the controller and shall communicate via northbound APIs.

Level 2 – Text-Based User Intents

In addition to complying with Level 1 criteria, VIVoNet shall automate the manipulation of the network infrastructure with text-based intents pushed through an application. VIVoNet shall transform these text-based intents into network parameters. The system shall then locate the relevant network elements and modify the network configurations to achieve the desired intent output.
Level 3 – Visual Representation

In addition to complying with Level 2 criteria, VIVoNet shall provide a visual representation of network architecture. The Graphical User Interface (GUI) shall imitate the simulated network based on the user intent. VIVoNet shall display the intent-based network modifications by highlighting the affected links the network topology.

Level 4 – Speech Recognition

In addition to complying with Level 3 criteria, VIVoNet shall use speech recognition API to dictate real-time transitions in the network. VIVoNet shall parse the speech to extract significant intents. These intents will be validated against justified test cases. The system shall then use the application to modify the network as per the validated intent and display on the GUI.

The deliverables for the project will include a software-based, orchestrated network, which shall be tested on three levels of network visualization, intents, and speech recognition. The initial level of testing will be based on accurate simulation of the user defined network on the GUI. The network shall comprise of interconnected nodes configured with required routing protocol. The second level of testing shall be based on application of text-based intents in the simulated network. The success and failure shall depend on the network manipulation/operation as per the text-based user intent. The final level of testing shall be based on the speech recognition via speech/text conversion engine, followed by validation of intent against the intent algorithm. The intent framework will be visible on the GUI.
VIVoNet is a comprehensive networking system, which will be used for real-time intent-based networking with a visual representation of the entire network infrastructure. The system will be driven by voice commands using a smartphone, computer, or a standalone product, similar to Google Home or Amazon Echo. VIVoNet will be portable and could be hosted on virtual environments like a Docker container or a virtual machine. The SDN controller will have control of the system while simultaneously running Speech Recognition and Intent-Based Networking applications.

VIVoNet comprises of four sections:

1. **Network Infrastructure:**

   The Network Infrastructure will be a series of hybrid switches capable of performing traditional and software-defined networking. In a real-time scenario, the network infrastructure could be a city-wide network used for disaster recovery and/or public safety or commercial businesses and datacenters. For practical purposes, the system will be a comprehensive collection of hybrid white-box switches to mimic a real-time environment.

2. **Intent-Based Networking System:**
Intent-Based Networking System will accept text inputs consisting of intent attributes and Policy information from the Speech Recognition System. The Intent Engine will create flows with the help of certain algorithms, which will be sent to the underlying infrastructure using automated configuration and orchestration platform.

3. **Visual Representation:**

Visual Representation comprises of open source APIs, which will display the underlying network infrastructure in real time. It will also highlight the intent-based route/path that will be created using Intent Based Networking System.
4. **Speech Recognition System:**

The Speech Recognition System will accept voice input from the user, which will be converted into a text format. It will then be compared with a Command Configuration Database, which includes the configuration details for setting up an intent network. If a match is found, the system will send configuration attributes to the Intent Based Network System via the SDN controller or else it will send a “Match not found” speech response back to the user. An open source API will be used for speech recognition and create a custom command configuration database consisting of intent-based networking operations.
**Technical:**

**CPE.1.1** Underlying network infrastructure should be active and robust. A network model consisting of an SDN controller, white-box OpenFlow switches and user-nodes should be redundant and fault-tolerant.

**CPE.1.2** Visual representation APIs should be able to modify and display the network infrastructure in real-time. The real-time transition should be visible on the GUI with minimal latency.

**CPE.1.3** Intent-Based Networking System. The intent engine should be able to validate the text and voice based intents to compute a command for the SDN controller to manipulate the network operation. The algorithm should be accurate in computing the correct command for the controller, or it should generate an error response. In no way should an incorrect intent be applied to the network.
CPE.1.4 Speech recognition API should accurately convert speech to text and match the input with the command configuration database. The speech parsing has to be error-free and exact, to obtain correct inputs in the intent engine.

Logistical:

The VIVoNet Team must:

CPE.2.1 Gather all the necessary APIs, Software, and Hardware required by VIVoNet.

Financial:

CPE.3.1 Voice-assisted automation devices required to achieve intent-based networking. VIVoNet shall be using open source software, and the hardware is available in the telecom lab. The system shall require Google Home or Amazon Echo Plus for Voice Inputs which will cost around $150.

3.0 Design Requirements

Functional requirements (FNC.X) as well as design requirements (DES.X.X) for VIVoNet are specified below to improve project clarity.

FNC.1 The system shall orchestrate the underlying network infrastructure using an SDN Controller.

DES.1.1 The required network shall be orchestrated as per a pre-defined topology on the datacenter infrastructure.

DES.1.1.1 The datacenter infrastructure shall consist of servers running hypervisors.

DES.1.1.2 The network topology shall consist of containerized or virtual devices running on the hypervisors and the links between the devices shall be connected automatically using virtual network interfaces.

DES.1.2 SDN controller shall communicate with the network devices using a southbound protocol.

DES.1.2.1 The system shall have an application running along with the SDN controller which shall communicate with the Intent-Based Networking and Speech Recognition Systems.

DES.1.2.2 The application shall then push the required changed through SDN controller to the devices using southbound protocol.
DES.1.3 The SDN controller must include automated validation functionality that verifies proper network configuration and expected operations.

DES.1.3.1 The system shall check configuration syntax before pushing them to the devices.

DES.1.3.2 The system shall also validate the generated configuration against the expected operation before pushing them to the devices.

FNC.2 VIVoNet shall receive inputs for functional intents from user via text/voice.

DES.2.1 The speech recognition system shall have a voice API engine for voice to text-based conversion.

DES.2.2 The system shall consist of a speech recognition application which will translate the commands directed by the user into a standard input to be fed to the application running on the controller.

DES.2.2.1 The speech recognition system shall have a command configuration database acting as comparison baseline for the voice commands.

DES.2.3 The system shall have a failover mechanism which will notify the user in case any system fails.

DES.2.3.1 The system shall have an application to send/receive keep-alive messages to all the components of the system.

DES.2.3.2 In case the speech recognition system fails, the system shall switch over to text based inputs for the intent-based networking system.

FNC.3 The input will be translated into corresponding configurations and the required function will be implemented.

DES.3.1 A speech recognition system will convert the voice inputs into a machine-readable format.

DES.3.1.1 A voice-based API shall be used to perform this operation. It can be any open-source API like Google API or Amazon Alexa API.

DES.3.2 The text output provided by the API shall be validated.

DES.3.2.1 The output will be matched with a database containing all valid commands/functional capabilities of VIVONet. If the command is valid, its intent will be treated by the processing engine. If not, the system will provide an error on the GUI or voice.
**DES.3.3** Processing Engine

**DES.3.3.1** The brain of VIVONet - the processing application – will intelligently convert the received inputs into flows which shall be installed on every switch in the network topology via OpenFlow.

**DES.3.4** Post-configuration validity check

**DES 3.4.1** Once the processing engine completes the configuration process, the system shall perform verification of the pushed configuration and report if the system failed to do the same on any of the concerned devices.

**FNC.4** The system shall display the network architecture with the specified intents on the GUI after being processed.

**DES.4.1** The system shall consist of an application that will visualize the complete network topology.

**DES.4.2** The GUI shall have the functionality to choose from the available devices in the network to perform the task between the devices.

**DES.4.3** The GUI shall highlight the path taken by the intent provided by the user.

**DES.4.3.1** The GUI should highlight the parameters related to path selected for the intent.

**DES.4.4** A text output should be present in the GUI which will tell user the steps taken by the controller to perform the task.

### 4.0 Key Design Options Considered

In order to start viewing different design options for VIVO Net, the various design requirements must be considered with respect to the SDN controllers. Figure 5 below, broadly classifies the SDN Controllers into two main categories: Vendor-Neutral and Vendor-Specific.
The team chose five primary controllers:

![Types of SDN Controllers](image)

**Floodlight**

The Floodlight Open SDN Controller is a Java-based enterprise controller which runs OpenFlow. It is a complete open-source project which was first initiated by a community of developers and engineers from BigSwitch Networks. It is well-known for being simple to build and run with minimal dependencies.

Some of the features that make it a worthy candidate for VIVONet are [6]:

- Support for a broad range of both virtual and physical OpenFlow switches.
- Multithreaded from ground-up. High performance guaranteed.
- Support for OpenStack Cloud Orchestration Platform.
- Apache-licensed.

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contains a comprehensive GUI</td>
<td>Knowledge of Java required to troubleshoot issues</td>
</tr>
<tr>
<td>Support for QoS, Security and Traffic Monitoring Capability</td>
<td>Limited configuration management</td>
</tr>
<tr>
<td>Good documentation available</td>
<td>Lacks a high availability mechanism</td>
</tr>
<tr>
<td>Very simple to build and run compared to other controllers</td>
<td></td>
</tr>
<tr>
<td>Integration with REST API</td>
<td></td>
</tr>
</tbody>
</table>
Open Daylight (ODL)

Open Daylight (ODL) is a modular open platform for automating networks of any size. ODL was built with the goal of simplifying network customization and programmability. The core of ODL platform is the Model-Driven Service Abstraction Layer (MD-SAL). In OpenDaylight, the underlying network devices are represented as objects or models whose interactions are processed by the SAL. The SAL is a data exchange and adaptation mechanism between YANG models representing actual devices and applications. The YANG models provide generalized description of the devices and application capabilities without knowing the specific implementation details.\[7\]

ODL includes support for broadest of protocols in any SDN platform including the following\[8\]:

- OpenFlow 1.3
- REST API
- Traffic Monitoring
- Security (ACLs)
- BGP, MPLS-TE
- Segment Routing (PCEP)

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extensive TE Support</td>
<td>Java Based – Resource Intensive</td>
</tr>
<tr>
<td>Extensive REST API Support</td>
<td>Most addons are not backward compatible</td>
</tr>
<tr>
<td>Good Integration with OpenStack</td>
<td></td>
</tr>
<tr>
<td>Open Source Development</td>
<td></td>
</tr>
<tr>
<td>Modular Architecture – Easy addition of new functionality via addons</td>
<td></td>
</tr>
</tbody>
</table>

Ryu

Ryu is a python-based SDN framework. Ryu supports APIs which help in creating network management and control applications. It supports OpenFlow versions 1.0 to 1.5. The following figure depicts how a Ryu controller fits in an SDN environment:
Using the built-in Ryu apps, one can implement the following functionalities:

- Switching Hub using OpenFlow
- Traffic Monitor - Monitors the OF switch statistics.
- Firewall – Firewall can be set using REST.
- QoS - QoS functions can be set using REST.

Ryu controller supports NETCONF and OF-config network management protocols. Ryu apps are single-threaded entities.

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python-Based</td>
<td>Single-threaded, hence low performance</td>
</tr>
<tr>
<td>Extensive Documentation</td>
<td></td>
</tr>
<tr>
<td>OpenStack Integration Capable</td>
<td></td>
</tr>
<tr>
<td>Open-Source Development</td>
<td></td>
</tr>
<tr>
<td>SDN Flow Based TE</td>
<td></td>
</tr>
</tbody>
</table>

**Cisco Apic-Em**

Cisco Application Policy Infrastructure Controller is an SDN controller mainly used for enterprise networks. It is a policy-based automation platform which is used to simplify and abstract the intricate network infrastructure. It can host multiple SDN based applications that use REST APIs as the northbound interface to provide network automation.\(^\text{[12]}\)

APIC-EM\(^\text{[12]}\) has many benefits like:

- Intelligent, open, programmable network with open APIs.
- Transforms business-intent policies into dynamic network configuration.
- Single point for network-wide automation.
Cisco APIC-EM\(^{[12]}\) provides some important features such as:

- **Network Discovery**
  
  Scans the network to build a centralized inventory database using Cisco Discovery Protocol (CDP), Link Layer Discovery Protocol (LLDP), Simple Network Management Protocol version 2 and version 3 (SNMPv2c, SNMPv3)

- **Policy Management**
  
  Used to define intents with the help of Access Policies. It is basically used to a set of rules for interaction between two or more endpoints.

- **Role Based Access Control (RBAC)**
  
  Users can be mapped one of the four predefined roles for added security.

- **Public Key Infrastructure (PKI) certificate**
  
  The PKI service provides an integrated authentication server for automated key management. It greatly simplifies authentication and trust in the network.

### Pros

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>24x7 Cisco support</td>
<td>High Cost</td>
</tr>
<tr>
<td>OpenStack Integration</td>
<td>Requires vendor specific devices</td>
</tr>
<tr>
<td>Extensive Documentation</td>
<td></td>
</tr>
<tr>
<td>QoS Capability</td>
<td></td>
</tr>
<tr>
<td>Extensive REST API Support</td>
<td></td>
</tr>
</tbody>
</table>

### Cons

- High Cost
- Requires vendor specific devices
- Requires using Multi-protocol Lable Switching (MPLS).\(^{[13]}\)

#### Juniper Contrail

Contrail controller is the Juniper proprietary controller which has not adopted Open Flow protocol yet. Juniper uses Xtensible Messaging and Presence Protocol (XMPP) to connect virtual routers in servers to Contrail Controller. The tunneling of Network traffic between virtual routers and hypervisors is done using Multi-Protocol Lable Switching (MPLS).\(^{[13]}\)

Juniper Contrail supports features like:

- Routing, bridging,
- Load Balancing,
- L3 and L4 firewall capabilities
- Active/Active redundancy
- Analytics service
- REST APIs to write customized application as per user need.
### Table

<table>
<thead>
<tr>
<th><strong>Pros</strong></th>
<th><strong>Cons</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Overlay SDN so use tunneling mechanism.</td>
<td>Cost</td>
</tr>
<tr>
<td>No need to do tenant segregation because of tunneling</td>
<td>Vendor specific support</td>
</tr>
<tr>
<td>Simpler to implement</td>
<td>Because of Tunneling, less control on traffic engineering</td>
</tr>
</tbody>
</table>

### 5.0 Trade Study Process and Results

VIVONet is a multi-tiered system. The interfacing between multiple applications running inside this system is a key aspect of this design. The language that the controller is written in and the familiarity of the team with that language is also critical. Cost is always a tilting factor when it comes to engineering designs. The below trade study has been done taking into consideration all these crucial aspects which will lead to a feasible and practical design.
<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
<th>WEIGHT</th>
<th>SCALE (1-5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LANGUAGE</td>
<td>The programming language in which the controller code is written.</td>
<td>0.25</td>
<td>1-Complex to develop 5-Simple to develop and integrate</td>
</tr>
<tr>
<td>API SUPPORT</td>
<td>The features that have in built support with APIs.</td>
<td>0.25</td>
<td>1- Few applications supported (1-5) 5- Large number of applications supported (5-10)</td>
</tr>
<tr>
<td>COST</td>
<td>Cost of buying the applications</td>
<td>0.25</td>
<td>1- Indicates that the controller is unaffordable. 5- Indicates that the controller is free or the cost is negligible.</td>
</tr>
<tr>
<td>OF VERSION SUPPORT</td>
<td>The maximum version supported by the controller</td>
<td>0.1</td>
<td>1- Specific versions supported 5- All versions supported</td>
</tr>
<tr>
<td>QoS SUPPORT</td>
<td>Applications to implement QoS rules and monitor traffic</td>
<td>0.15</td>
<td>1- No support 5- Full support with APIs</td>
</tr>
</tbody>
</table>

Table 1 Trade Study Parameters, Weights, and Reasoning

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Weights</th>
<th>ODL</th>
<th>RYU</th>
<th>Floodlight</th>
<th>APIC-EM</th>
<th>Contrail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language</td>
<td>0.25</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>API Support</td>
<td>0.25</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Cost</td>
<td>0.25</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>OF Version</td>
<td>0.1</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>QoS Support</td>
<td>0.15</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Total</td>
<td>1.0</td>
<td>3.8</td>
<td>4.4</td>
<td>3.8</td>
<td>3.8</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 2 Trade Study for VIVoNet Design Concepts
6.0 Selection of Baseline

6.1 Selection of components

The Ryu Controller was the chosen SDN controller based on its score in the trade study having a number of qualitative advantages. It is entirely written in python programming language which is considered to be highly simplistic and flexible. Python is considered to be the primary language for network programming. Python is a light-weight tool which is platform independent, highly modular, and easy to understand.

The API support for Ryu Controller scored better than the rest because Ryu has an extensive API support for a variety of functions. The APIs are designed using REST framework which makes it easy to integrate with external applications. The APIs of the Ryu Controller support traffic monitoring information including per port statistics, link bandwidth and delay which are important for the VIVoNet system to find a better path to a destination specified by the user. Furthermore, Ryu also supports QoS implementation through the APIs which can be leveraged by the VIVoNet system to design a high priority path in the network.

Ryu is an open flow controller and because of this, it does not have any license cost attached to it which was the key factor to select RYU controller as the baseline of the project. Also, Ryu supports the creation of new applications for network management and control so it will be added advantage for the project.

Vendor specific controllers like Juniper Contrail and Cisco APIC-EM have limitations to operate on specific devices while Ryu being an open software can work in the heterogenous environments. Organizations can customize their deployments according to their specific needs and developers can modify the components to meet the requirements of the underlying architecture.

Ryu supports most of the OpenFlow versions i.e. 1.0, 1.2, 1.3, 1.4, 1.5. OpenFlow is considered to be one of the first SDN standards. It is one of the widely used southbound protocol standard in any SDN environment. All the devices in the network infrastructure must support OpenFlow in order to establish communication with the controller. It provides a high level of abstraction and granularity for packet forwarding, routing, and traffic manipulation.

The QoS support of the Ryu Controller is equal to the ODL Controller because both the Controllers provide extensive Layer 2 and Layer 3 QoS Support. The ODL does happen to be a bit slower and complex in the implementation, mainly because the addons for QoS are developed in Java. On the other hand, Ryu Controller has a multi-threaded
Python-based implementation which not only supports Class of Service QoS at Layer 2 but also supports DSCP marking for the Layer 3 QoS. The other solutions CISCO APIC-EM and Contrail have a more optimized solution for QoS but only work on Cisco Nexus Switches and Juniper Routers. Ryu Controller provides VIVoNet a more flexible, easy to integrate and a vendor-neutral solution for the QoS support.

Now that the Ryu controller has been selected as the baseline design, the VIVoNet team will move forward to implement this idea. Although the overall system has been selected, many options remain as to which controller can be successfully integrated with the Intent-Based Networking and Speech Recognition Systems. The Ryu Controller has an extensive API support and the controller is developed in Python which makes it easy for the VIVoNet team to add more functionality to it. However, in case the Ryu controller does not work as expected with other systems in compliance with DES.3.3.1 and DES 3.4.1, we propose to failback to the ODL controller which satisfies all the baselines requirements.

7.0 References


