International Journal of Innovative Research and Advanced Studies (IJIRAS)
Volume 3 Issue 5, May 2016

ISSN: 2394-4404

Evalution Study And Demonstration Of Openflow Overlay Application Interface Usage Using Autonetkit

Kehinde Adebusuyi

Hilary Ezea

Intelligent Systems & Communication Networks Research Group,
Department of Electrical and Electronic Engineering,
Federal University Oye-Ekiti,
Ikole-Campus, Nigeria

Abstract: ***We explain the notion of software-defined networking (SDN), whose southbound interface may be implemented by the OpenFlow emulation application such as Autonetkit. We evaluate the practical operation of Autonetkit and the Overlay application Interface Usage. We give an overview of existing OpenFlow-based Overlay applications and the proposed model with network graph. Finally, we point out model design choices for SDN inter-domain routing using OpenFlow evaluation and discuss their performance implications.***

Keywords: Software Defined Network Interconnection, OpenFlow, Autonetkit

I. INTRODUCTION

Recently, Software Defined-Network – [SDN] controller has been identified by researchers as the enhancement to the de-facto legacy Internet Protocol networking architecture and a paradigm shift from rigid control and data Plane which the current next generation internet fabric is built on.

The issue of management, troubleshooting and security are hard fought as manual configuration of several knobs are still great issue. Stefano et.al [1] proposed an Open Source Hybrid IP/SDN {oSHI} model which combines Quagga for OSPF routing and open vswitch for open flow based switching on Linux platform. In their experimental comparative study between the emulation on minimet and distributed SDN testbeds, Oshi was suitable for large scale data center as demonstrated by the authors in [2] and a methodology of reducing the route oscillations was suggested where the BGP sample transactions were observed at various outage time. The current demand for high speed at the backbone is a major driver for an efficient and reliable backbone that can deliver uninterrupted internet applications services such as on-line gaming, multimedia communication and transport networks services anywhere, anytime. In a recent 5 year report by a major player in the telecommunication industry in the European region – () in September 2015, reveals that the future internet applications growth is forecasted to grow at 70% rate. This is attributed to the development of a high speed backbone that is scalable to cater for the growing needs of the service providers and network operators. This paper evaluates the networking emulator networks in experimental support using Autonetkit Overlay Application. The remainder of this paper is arranged as follows: Section 1 gives us an overview of the Software-Defined Networks and the current need for an emulator to test SDN experimental ideas. Section 2 relates the introduction with the Motivation for Autonetkit Overlay applications. This section discusses the trend in Data Communications, the challenges in traditional Networking and the evolution of programmable networks. Section 3 is the continuation of section2 which gives a bird eye-view of the network model, algorithm for the architecture and framework of Autonetkit Overlay within the academia community. Section 4 is the comprehensive evaluation study to analysing issues related to the concept of Abstraction, the types of Abstraction. The prospect of a programming language in the NOS and a focus on Python as a high-quality programming language for this work and its application in measuring Large-

Scale Networks. Section 5 of our work concludes this paper which is another step towards extending the value proposition of Autonetkit Overlay on inter-domain level or multi-domain level. We propose a model to experiment and explore maximally redundant techniques for fast re-routing on the IP layer and the possibility of an autonomic approach to Managing Cognitive Hybrid Software defined Networks. The promises of Autonetkit Overlay uses network graphs within domains.

II. BACKGROUND AND RELATED WORK

We propose an emulation network that thrives on network graph creation using open source tools such as minimet. Quagga, Autonet kit, networkx, and programmable networks developed with python languages. …et .al [2] developed a model and emulation framework for BGP Evolution, in their model, Multi-domain SDN was proposed to be outsourced to an external contractor to provide inter-domain routing services facilitated through a Multi-As Network Controller.

The need for a self-aware systems was justified by Gelenbe, E.[3] in his work on software-defined self-aware network proposed a Cognitive Packet Network (CPN) which is able to improve the quality of services at the Backbone Network.

A. MOTIVATION FOR AUTONET KIT OVERLAY APPLICATION

The main motivating factor for Autonet kit Overlay Application, is the ease of configuration in networks that eliminates poorly understood interaction between local policies, poor convergence and lack of appropriate information hiding, non-determinism and poor overload behaviour.

Rather than an engineer doing all the horrible scripting, one of the motivation from SDNi is that of an obstructions on the industry part and it philosophical one of them is better programming instructions which our proposed model really drive and the mechanism to push those scripts out which makes thing horrible even though the API Push those scripts out. There is a bit of control being able to describe the network with high level to build what you want the network to look like. The management plane will handle separate responsibilities to fulfil the main conceptual idea of Software Defined Network. This SDN partitioning will secure each domain on its own thereby contributing to privacy across the network.

These new issues drive the research on new routing methods. Our goal is to adhere to this imminent drivers in the software defined next-generation network experimental research domain as it appears as closely as possible by experimenting with Autonet kit to develop models. The Autonet kit in fig 1.0 consist of a Cisco Virtual Internet routing Lab (VIRL) with a user mode of LINUX running Quagga routing. It occupies a 32MB RAM size of three (3) virtual machines.

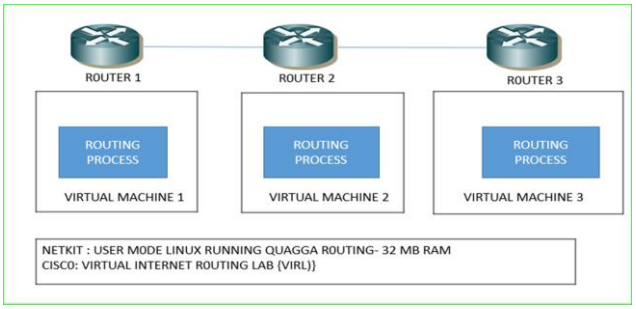


Figure 1.0: Cisco Virtual Internet routing Lab (VIRL)

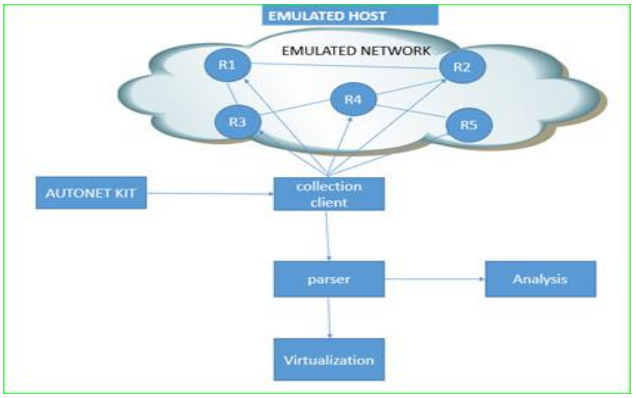


Figure 2.0: Emulated Network with Autonet kit

III. COMPREHENSIVE SURVEY: BOTTOM-UP APPROACH

We consider a use case in measuring large-scale networks, where we can have two networks built on python scripts communicating with each other. Essentially the protocols running inside one network is slightly different from that of the other in another network and the protocols running between the borders and configurations need to cooperate to include information such as business relationships. It is a lot of complexity once we start building them into bigger networks, so what we propose is a model to break the networks into smaller networks. This is slightly small in terms of the bigger networks with 40 or 50 routers with plenty of networks on the internet. We propose a model where the configurations of the devices with low level vendor specific syntax is utilized. This is similar to the work of ….. et. al where he discussed abstractions as a solution to the simulation of complex networks . He however proposed the breaking down of complex networks into smaller ones . This is essentially fairly similar between each devices but pretty tedious and takes time to repeat the process across different network devices. For router exchanging messages within the same network and different across the network. All these configurations is tedious, time consuming and boring and does not scale when across large networks.

In term of trying to do the configurations [], one approach is to have a database push into templates and the problem becomes how do we specify these properties in the database because we have raised the abstractions and from typing into each devices, we can push it out into the network albeit getting the device level properties across the network is a big challenge. We start by looking at an idea of a simple network

design where the networks are in autonomous systems as seen in figure 3.0 []. The AS1

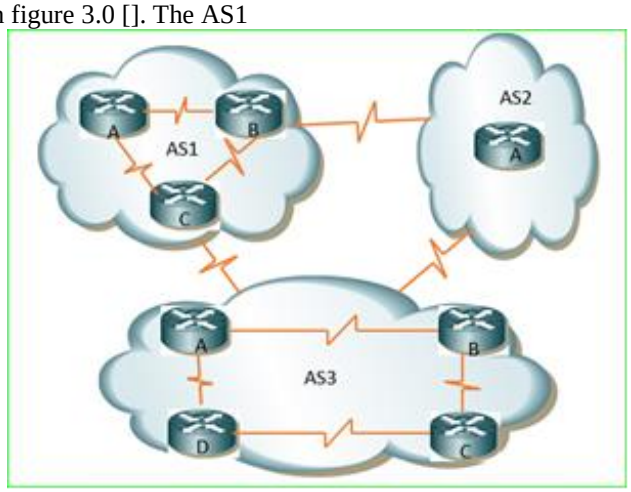


Figure 3.0

A. APPLICATION USAGE

Autonetkit

B. NETWORK MODEL

The figure 4.0 represents the schematic digaram of the overlay showing the input graphML with the loader which set the defaults for the compiler with input from various routing protocols to create the overlay for the device configurations module because the network is configured device by device. The overlay module is booked in the condensed graphs for the abstract network module level which is pushed out to the various devices and generate functions in Table 384884: We have graph attribute output from ibgp, ebgp which is pushed out to API Like netcomf , SNMP, 0nePK, saltstack, puppet, etc

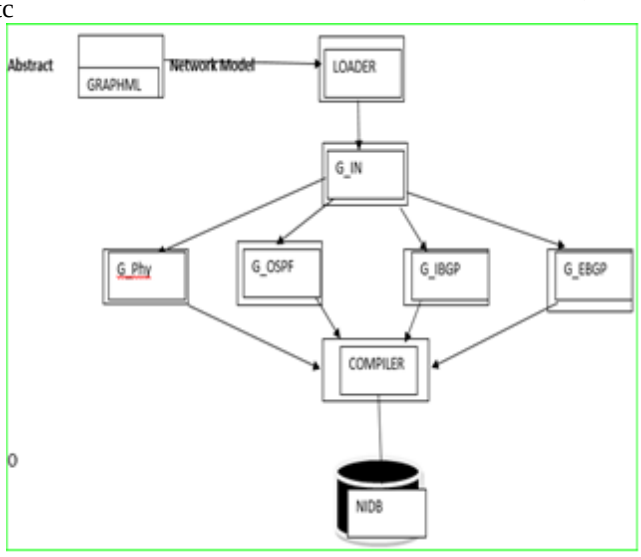


Figure 4.0: Autonet Kit Network model

The input graph model G_in with individual inter and intra routing protocol (IBGP & EBGP) and the compiler computes the parameters for packet flows into the NIDB

C. EBGP GRAPH CREATION

We consider Ebgp as a use case in the network model. Ebgp is the protocol that routes between autonomous systems. It works inversely to ospf which is adding edges, if source is not the same. The following network graph shows the overlay from the loader that pushes packet flows into the ebgp edge graph module.

```
; g_in = anm [ ‘ input’ ]
  Print  [n for  n  in  g_in if n.asn ** 1]
 Print [e for e  in g_in.edges () if  e.src.asn (= e.dat.asn)
```

```
g_ebgp = anm.add_overlay {‘’ebgp’’ g_in}
g_ebgp.add_edges_from {e for e in g_in.edges{}
            if e.src.asn ** e.dat.asn}
```

If the source asn is equal to the destination asn, the the input graph is equal to the anm input.

IV. EVALUATION AND DISCUSSION

We evaluate this study with autonet kit running web server in single-user mode client connected from 10.0.0.1. Figure 5.0 reveals a sample Autonetkit in a network. The Autonetkit application sends hyper transfer text protocol (http) post to the web server 10.0.0.1 through the individual connected web sockets d3.js

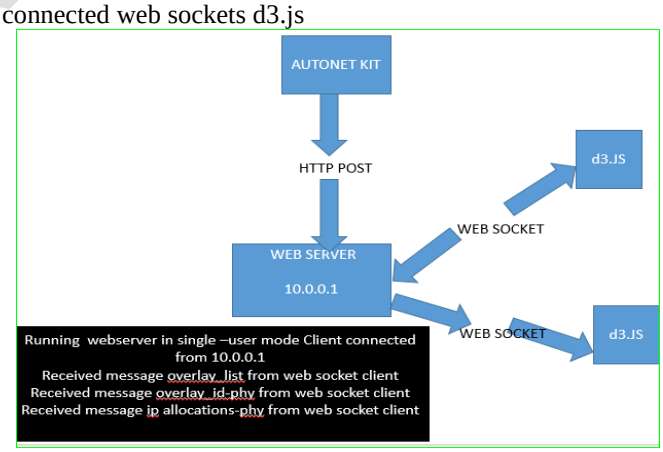


Figure 5.0: A sample Autonetkit in Network

The physical graph creation consists of links with soure, speed and target at different rates. The node received the asn with identification number 3. for each coordinates in a device type “router” as follows:

```
‘’phy’’:  {
“Links”: {
{ ‘’source’’:  0, ‘’speed’’  10,  ‘’target’’:  11 },
{ ‘’source’’:  0, ‘’speed’’  10,  ‘’target’’:  12 },
{ ‘’source’’:  1, ‘’speed’’  10,  ‘’target’’:  11 },
}
‘’node’’:  {
{ ‘’asn”: 3, “device_type”;  ‘’router”,
     “id”:  “11”,  “x”;  251.0,   “y”:  349.0 },
{ ‘’asn”: 3, “device_type”;  ‘’router”,
```

"id": "10", "x"; 250.0, "y": 201.0 },
{ ''asn": 2, "device_type"; ''router",
"id": "11", "x"; 554.0, "y": 47.0},

In terms of doing some basic network creation using API, we considered using open Shortest Path First {ospf} as it runs within autonomous Systems similar to Border Gateway Protocol. We took node from the input graph and install on routers on the network and we added the edges from the input graph such as to all the routers within network and adding the edges from the input graph if the source is same as the edges. We draw the links to the networks as shown in figure 7.0

What we used is router configurations basically. Figure 1.0 shows the representation of a router packed into each other with routing protocols running on them and they communicate to each other and find a path. We built them into a big network, and essentially form the internet. By grouping this routers into different properties such as autonomous systems AS1221 Telstra and AS7575 AARNET. We can form different configurations as shown in Figure 6.0.

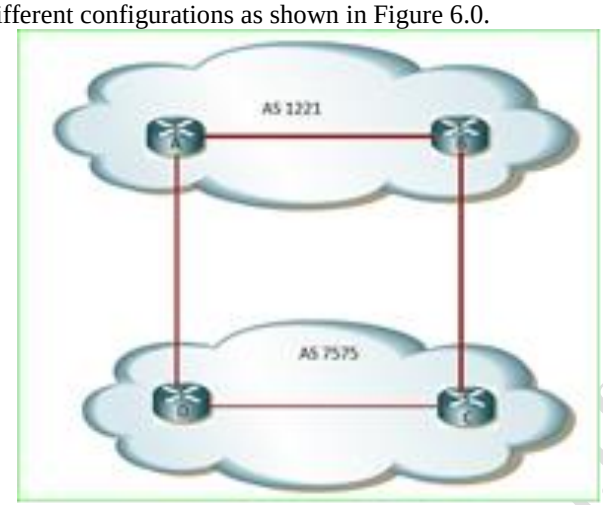


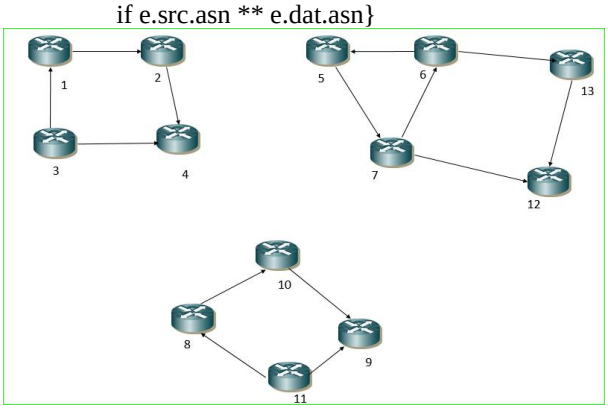
Figure 6.0: Autonomous systems AS1221 Telstra and AS7575

A. OSPF LINKS AND NETWORK GRAPH

The network in Table71.0 was drawn in Yed Diagramming Application and import directly into networkx and see the attributes and nodes [Table 1.0] . We iterate over the nodes

In [4] :

```
g_ospf = anm.add_overlay {''ospf'' g_in}
g_ospf.add_edges_from {e for e in g_in.edges{}
        if e.src.asn ** e.dat.asn}
```



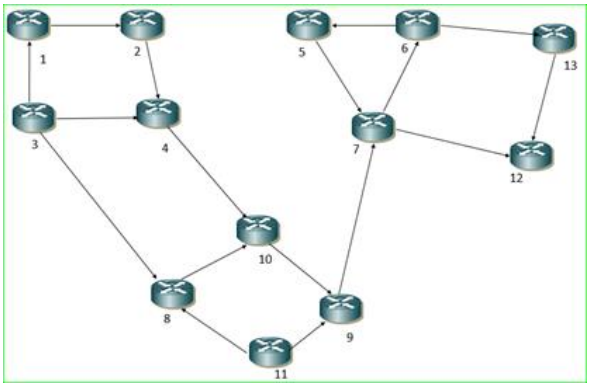


Figure 7.0: A sample OSPF Autonomous Systems in Network in Yed diagramming

B. ALGORITHM

In using AutoNetkit, we created an algorithm that imports from autoNetkit, load to build the network

Using AutoNetkit

```
Import autonetkit
From autonetkit import load, build_network
Input_graph = load.graphml.load_graphml
('demo.graphml' )

anm = autonetkit.ANM( )
anm = build_network.initialise (input_grpah)
```

In [2] [1, ,3, 2, 4]

[input: (10, 4), input: (3, 8), input: (2, 5), input: (7, 9)]

PHYSICAL GRAPH

In [3]:

```
g_phy = anm[ 'phy ']
g_phy.add _nodes_from [g_in, retain = ['asn', 'label',
'route_reflector' ']
g_phy.add_edges_from[g_in.edges{}]
print list [g_phy .nodes{}]
print list [g_phy .edges{}]
```

[11, 10, 13, 12, 1, 3, 2, 5, 4, 7, 6, 9, 8]

[phy: {11, 9}, phy: {11, 8}, phy: {10, 9}, phy: {10, 8},

```
g_ibgp = anm.add_overlay {''ibgp'' g_in}
g_ibgp.add_edges_from {{s,t,} { for g in g_ for t in g_in
}
                if s.asn = t.asn}
```

phy: {10, 4},

[phy: {13,12}, phy: {13, 6}, phy: {12, 7}, phy: {1, 3}, phy: {1, 2},

[phy: {3, 8}, phy: {3, 4}, phy: {2, 5}, phy: {2, 4}, phy: {5, 7}, phy: {5, 6}, phy: {7, 9}, phy: {7, 6},

The physical graph above in Figure … is readable taking into consideration the topology in Figure [].. we could printout the path for the node list and edge list. When building up multi queries, it became quite difficult. A visualization system of the scripts with d3 is seen in figure 7.0.

C. IBGP NETWORK GRAPH

The IBGP is the protocol within a network to communicate with another external network. An example is creating a full mesh topology and generate all the nodes and edges in the source network are the same as the nodes and edges in the destination. The ibgp are running in full mesh is going to be border end as seen in the script in table 2.0 .In pretty large neworks,,route reflector works better which simply states that rather than every peer connecting to each other. Every routers and nodes connect to a central source and build as shown in Figure ….. In terms of expressing with graph operations, the syntax if we look at Autonetkit , we could look at the nodes in previous graph and group them by asn and give us a list of all the asn property and the nodes that have the asn in a set of routers and compare with another asn in a set of routers.

In [6] :

In [7] : g_ibgp = ans.add_overlay{"ibgp" g_in, retain = "route_reflector'

For asn, nodes in g_ibgp.groupby {'asn'} . items { } :

rrs = {n for n in nodes if n.route_reflector}

if len {rrs} :

// setup rr hierarchy

Clients = set {nodes} = set {rrs}

Edges = { { s,t } for s in rrs for t in clients }

g_igbp.add_edges_from { edges}

In [7] : g_ibgp = ans.add_overlay{"ibgp" g_in, retain =

With this comprehension, looking at all the list of nodes in that asn dictionary if they have a route reflector set, if there is a link there, then we can do a set of operations….by determining, which direction to follow, which node is set and which node is done and creating a full mesh between the route reflectors and the those edges for the nodes {figure ---0000}without the set attributes as shown in Table …

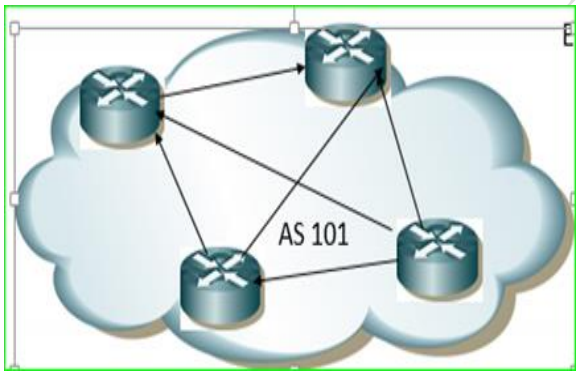


Figure 8.0

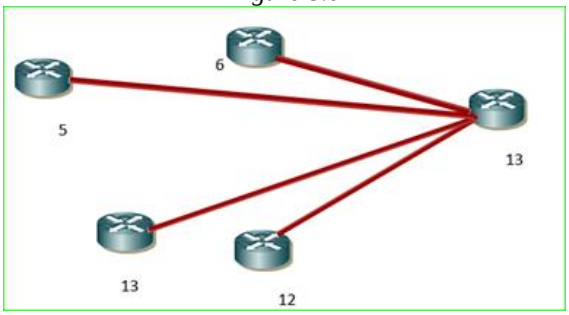


Figure 9.0

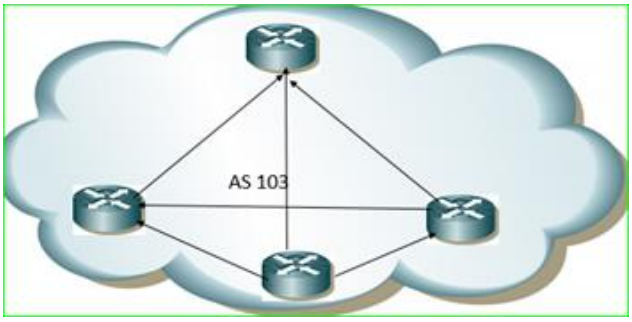


Figure 10.0

V. CONCLUSION AND FUTURE RESEARCH

The design approach in Table088 gives the templates which you can specify the compiler to use either as quagga, cisco or other routing vendor. We used Mako which is similar to other device template languages, defined the template, add iteration conditions with variable substitutions systems with similar device configuration syntax and network labels and use the set and get attributes on the nodes.

REFERENCES

[1] S. M. Metev and V. P. Veiko, Laser Assisted Microtechnology, 2nd ed., R. M. Osgood, Jr., Ed. Berlin, Germany: Springer-Verlag, 1998.

[2] J. Breckling, Ed., The Analysis of Directional Time Series: Applications to Wind Speed and Direction, ser. Lecture Notes in Statistics. Berlin, Germany: Springer, 1989, vol. 61.

[3] S. Zhang, C. Zhu, J. K. O. Sin, and P. K. T. Mok, "A novel ultrathin elevated channel low-temperature poly-Si TFT," IEEE Electron Device Lett., vol. 20, pp. 569–571, Nov. 1999.

[4] M. Wegmuller, J. P. von der Weid, P. Oberson, and N. Gisin, "High resolution fiber distributed measurements with coherent OFDR," in Proc. ECOC'00, 2000, paper 11.3.4, p. 109.

[5] R. E. Sorace, V. S. Reinhardt, and S. A. Vaughn, "High-speed digital-to-RF converter," U.S. Patent 5 668 842, Sept. 16, 1997.

[6] M. Shell. (2002) IEEEtran homepage on CTAN. [Online]. Available:http://www.ctan.org/tex-archive/macros/latex/contrib/supported/IEEEtran/

[7] FLEXChip Signal Processor (MC68175/D), Motorola, 1996.

[8] "PDCA12-70 data sheet," Opto Speed SA, Mezzovico, Switzerland.

[9] Karnik, "Performance of TCP congestion control with rate feedback: TCP/ABR and rate adaptive TCP/IP," M. Eng. thesis, Indian Institute of Science, Bangalore, India, Jan. 1999.

[10] J. Padhye, V. Firoiu, and D. Towsley, "A stochastic model of TCP Reno congestion avoidance and control," Univ. of Massachusetts, Amherst, MA, CMPSCI Tech. Rep. 99-02, 1999

ISSN: 2394-4404

[11] Matlock, H., and Reese, L.C., 1960, Generalized solutions for laterally loaded piles., Journal of Soil Mechanics and Foundation, 86(5), 63–91.
[12] Nayak, G. C., and Zienkiewicz, O. C., 1972, Convenient forms of stress invariants for plasticity, Proc. ASCE, 98(4), 949-953.
[13] Noorzaei, J., Viladkar, M. N., Godbole, P. N., 1995, Influence of strain hardening on soil-structure interaction of framed structures, Computers & Structures, 55(5), 789-795.
[14] Owen, D. R. J., and Hinton, E., 1980, Finite elements in plasticity-theory and practice, Pineridge Press, Swansea.
[15] Pise, P. J., 1982, Laterally loaded piles in a two-layer soil system., J. Geotech. Engrg. Div., 108(9), 1177–1181.
[16] Poulos, H. G., 1971, Behavior of laterally loaded piles-I: Single piles., J. Soil Mech. and Found. Div., 97(5), 711–731.
[17] Reese, L. C., and Matlock, H., 1956, Non-dimensional solutions for laterally loaded piles with soil modulus assumed