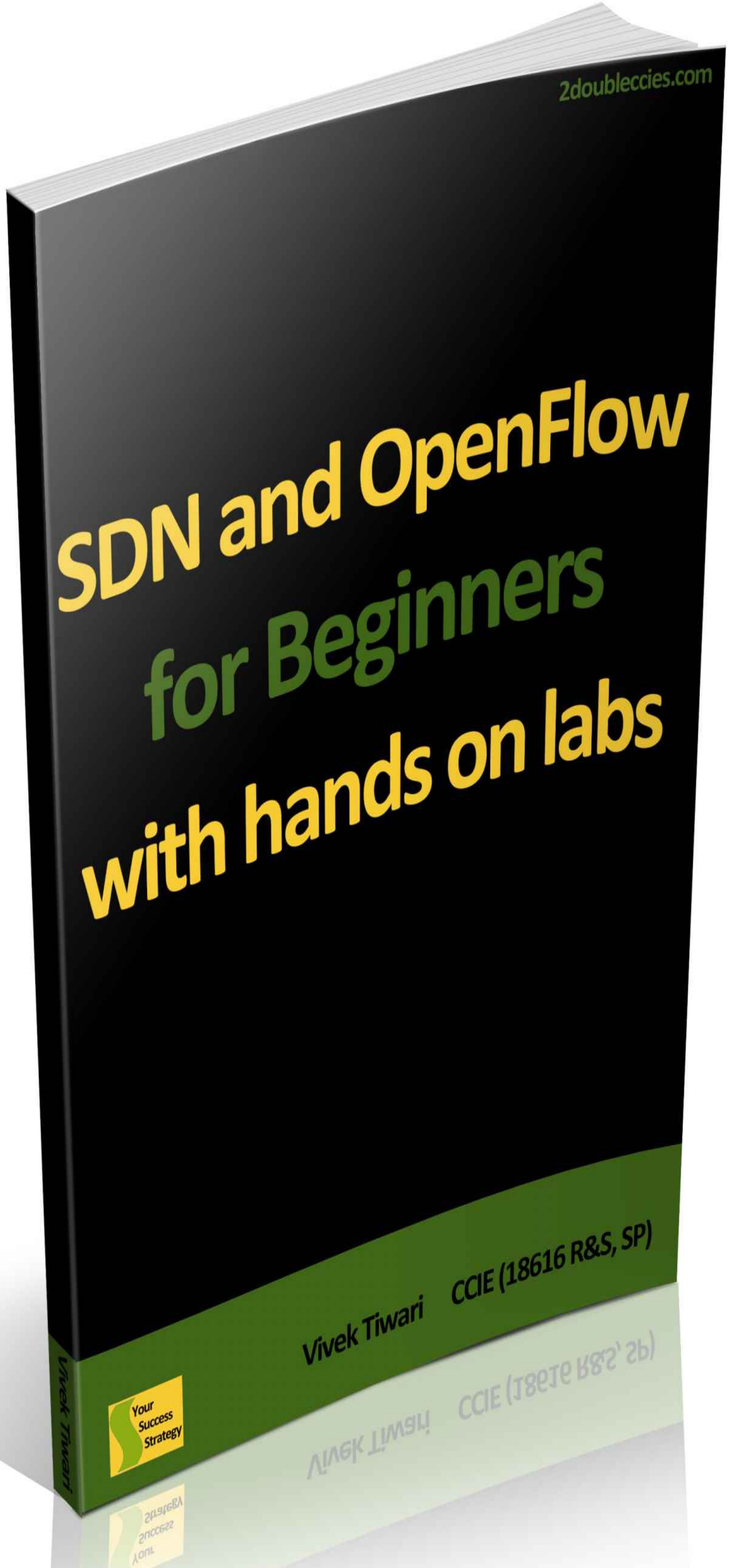


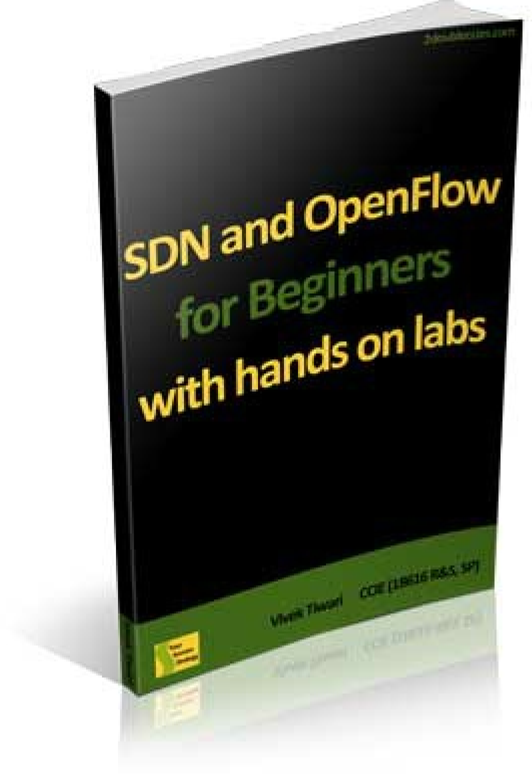
2doubleccies.com

SDN and OpenFlow for Beginners with hands on labs

Vivek Tiwari CCIE (18616 R&S, SP)

Vivek Tiwari

Your Success Strategy


2doubleccies.com
SDN and OpenFlow
for Beginners
with hands on labs
Vivek Tiwari
CCIE (18616 R&S, SP)

SDN and Openflow for beginners with hands on labs

Vivek Tiwari

COPYRIGHT, WARRANTY AND DISCLAIMER

contact@2doubleccies.com

Warranty and Disclaimer

This book “SDN and OPENFLOW FOR BEGINNERS WITH HANDS ON LABS” and information herein are provided “as is,” without any warranties or representations, express, implied or statutory, including, without limitation, warranties of non-infringement, merchantability or fitness for a particular purpose.” A sincere and comprehensive effort has been made to make this book as complete and as accurate as possible. MMDD Multimedia LLC. And the authors shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book. All pictures and illustrations are copyright of the author except for the picture of the genie which has been made by someone unknown to the author.

Trademark Acknowledgments

All trademarks are the property of their respective owners. MMDD Multimedia LLC., is not associated with any product or vendor mentioned in this book. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark. MMDD Multimedia LLC. Cannot attest to the accuracy of this information.

Cisco, CCIE, Juniper, GNS3, Dynagen, Dynamips, Openflow, VMware, Java are trademarks or registered trademarks of their respective companies and/or its affiliates in the United States and certain other countries. The author does not represent Cisco in anyway. This book and its contents have not been reviewed, approved, or endorsed by Cisco.

About The Author

Vivek Tiwari CCIE # 18616

(Routing & Switching and Service Provider)

Vivek Tiwari holds a Bachelor's degree in Physics, MBA and certifications from multiple vendors including Cisco's CCIE. With double CCIE on R&S and SP track under his belt he mentors and coaches other engineers. Vivek has been working in Inter-networking industry for more than twenty years, consulting for many Fortune 100 organizations. These include service providers, as well as multinational conglomerates. His six plus years of service with Cisco's Advanced Services has gained him, the respect, and admiration of colleagues and customers alike. His experience includes and not limited to network architecture, training, operations, management and customer relations; which made him a sought after coach and mentor, as well as a recognized leader.

Vivek is also the author of

Your CCIE Lab Success Strategy the non-technical guidebook.

Stratégie pour réussir votre Laboratoire de CCIE: Le guide non-technique.

Your CCNA Exam Success Strategy the non-technical guidebook.

Your CCNA Success Strategy Learning by Immersing.

Acknowledgements

I would like to thank Donna Menna and Avantika Tiwari our editors for converting my engineer-speak into an understandable and printable language.

I thankfully acknowledge the helpful review and suggestions from:

Kai Wang

Sean Garrett

Harish Dommathamari

Dean Bahizad

Lastly, I would like to thank everyone else who assisted and encouraged me with this work.

Reviews

Kai Wang CCIE # 6130

This book was an easy read and will be very accessible to novice SDN and OpenFlow learners. A great introduction for anyone that would like to learn about this new technology!

Sean Garrett CCIE # 11390

This book reads well and does a great job in breaking down SDN fundamentals into easily understand analogies. If the reader is contemplating using SDN at the enterprise or service provider level, this is an excellent resource to read. It was good to see the concepts expanded into a broader scale, spreading across disparate networks and protocols. To the casual reader and someone not very familiar with SDN, this book will help to explain the technically challenging aspects, and gain a better understanding SDN fundamentals in the network.

Harish Dommathamari CCIE # 9891

SDN is one of the most talked about industry terms today and this book is the definitive read on getting to understand SDN and OpenFlow. Well-structured and simple to read, combined with hands on labs on SDN using OpenFlow this book serves as a good beginner's guide for anyone who is interested to learn about SDN.

Dean Bahizad CCIE # 18887

Before reading this book, I really had no idea where to start with Software Defined Networking (SDN), just a vague definition. The easy to follow format and concrete examples really helped me understand SDN more. (The Borg example is really good and easy to relate to).

I also found the step-by-step instructions on how to set up the hands-on lab really helpful. Great book for anyone new to SDN—worth every penny!

DEDICATION

I would like to dedicate this book to all those network engineers who took their time to answer my questions, even the ridiculous ones. Their deep understanding of networking, patience and the willingness to impart knowledge has helped me to be what I am today.

CHAPTER 1 - WHY THIS BOOK?

I had been hearing about SDN for quite some time. This was supposed to be a technology disruption that was much needed for the networking market. Then came July 23, 2012: A company called Nicira was acquired by VMware for $1.25 billion. They spent such a hefty amount on a startup because Nicira does the same for network virtualization as VMware did for server virtualization.

This $1.25 billion weight behind the technology made SDN mainstream. The buzz going around the technology world was that Nicira is going to threaten giant companies like Cisco and Juniper. SDN created a lot of debate and discussion throughout the networking world. It came out of the geeks realm and became a buzzword. The Nicira spotlight brought SDN into focus for the technology savvy CEO's and CTO's. In short, all this commotion put SDN on the tech horizon of a whole lot of companies.

As I was finishing my other book, news articles and write-ups were clogging my inbox and I was being asked about SDN by friends and customers alike. I looked around and there was a plethora of information; but, like many other things on the internet, there was a big pile of *jumbled* information. There was a lot of information that was already out of date and then there were articles which were biased towards or against SDN. On one end of the spectrum, there was too much information by people who didn't know much about SDN, which sometimes translated into misinformation. On the other end of the spectrum, there was too much technical information, which became OHT (overhead transmission) and put you to sleep.

I looked at YouTube and there were hundreds of videos explaining SDN from individuals explaining their view of SDN to live product demonstrations from different players in this game.

Note: *Interestingly a Swedish company www.tail-f.com did a survey and found out that 87% of NA enterprises see SDN as more important than cloud or virtualization, yet only 51% know what SDN is!* ***The key here is not the numbers but the fact that SDN needs much more awareness.***

I jumped on to Amazon, and lo and behold, there was just one book on SDN. I was surprised, but, at last, I thought there was something I could find that would have relevant information in an ordered fashion. A 400-page book of technical reading was not exactly what I was looking for; but even that was a pre-order and was not going to be available for 90 days or so.

That is when I decided to write this beginners guide. I wanted to keep this short and to the point: something that you can read in three or four sittings. A book that gives you an idea about SDN, helps you speak intelligently about it, enables you to ask intelligent questions, and even gives you the sense if this is something that you would want to pursue further.

If you are hands-on engineer like me then you learn more by getting your hands dirty in the sandbox (your lab). That is the reason I have added the hands on portion to this book also. This will get you on your journey to the next realm of networking.

I also wanted to stick to the basics and be vendor neutral. Since I am basing this book mostly on version 1.0.0 of OpenFlow (version 1.3.3 is the latest) this book should stay relevant for anyone who wants to test the waters. Most of the vendor products now support OpenFlow version 1.0.0 and support for higher versions is being added.

Some of you may find this book very basic while others may also find this insightful. It all depends how much you have read and heard about SDN and from whom. Whatever you feel about the book, I am happy to hear from you. Your comments and suggestions to this book will be valued immensely. I do not claim to be an expert on SDN and OpenFlow. There are many others who know more than I do and I will be mentioning the blogs and websites that I have seen of great value to learn about SDN at the end of this book in the resources section.

This book will give you the concise knowledge that you need to start on SDN in a step by step fashion which will save you hours and hours of searching and sorting of online articles and blogs which may have become too old to be relevant, be colored by views of people who are for or against SDN or views that may be there to promote one company product or the other.My knowledge of SDN came from reading hundreds of blogs and informative articles.As a result, this book is the condensation of this online knowledgebase and my personal knowledge and experience.

Part I: What is SDN?

CHAPTER 2 - WHAT IS SDN?

That is the first relevant question to ask. What is SDN?

You will hear a lot of statements like:

- SDN decouples the control plane from the data plane.
- It allows us to change network behavior dynamically.
- SDN (Software defined networking) is exactly what it says, network behavior defined by software.
- It opens the door to network intelligence so that you can get lot more value from your network.

Like everything else in networking, the answer is, "***IT DEPENDS***."

For a **Network Engineer**, it is the next cool technology which you will need to keep a close watch on. If this technology meets or exceeds the expectations, then this is the next big evolution of networking and you will need to be prepared to ride the wave.

For a **Technical Sales Engineer**, it is not only the buzz word that you can use every now and then, but also something that you have to be aware of and ready to answer questions to customer network Architects and CTO's. They will need to address questions like, how is this technology relevant to the customer and how is their current hardware compatible with this technology? What are the added benefits to them?

For a **Network Manager**, it is something still on the horizon and may affect you three to five years from now. Those at the select few organizations that like to be on the cutting edge will think different for sure as they may have SDN in the labs or even a pilot somewhere. However, like any other new technology, there will be the initial technology absorption pains. In the end, this will make the job easier as configuration management and traffic flow within the network will be controlled in a more granular and centralized fashion. This means more control for the network manager.

For a **CTO,** it means that the overall hardware costs will decrease while they can roll out better services for their organizations. These services will be flexible and scalable which will be easier to operate.

For a **Programmer,** it means creating API's that allow software to dynamically control network devices at a depth down to the ASIC level on the ports of a switch.

For **Software Developers,** this open field will grow as much as their imagination can. All the current applications can take advantage of this new flexibility of the applications talking to the network infrastructure and the network responding dynamically to its needs.

So what the *&^%$ is SDN?

One word that came to my mind by reading about SDN was “BORG”. For those of you who have not seen *Star Trek*, the Borg’s are an alien race that can assimilate you and then control you from a central hive mind. I am sure some Star Trek fans can dig deeper into this and start picking apart the difference between SDN and BORG's but if I have to explain SDN to someone who is a *Star Trek* fan in one word, the BORG collective will explain how the central SDN controller will make use of the data plane of switches. Besides the fictional race of the Borg’s let me explain SDN with a real world personal example.

Many of us, unfortunately, are used to getting into a traffic jam on a daily basis on our way to work.

My office used to be five miles off of the freeway and it used to take more time to travel those five miles than the ten miles I travelled on the freeway. One day I was running a little late for work, but to my surprise, I had actually arrived at work about 5 minutes early. When it happened again the next day, I realized that all the traffic lights (see diagram below) had been timed such that if you are travelling at the speed limit you will get a green light all the way through. The city had not only coordinated and timed the lights, but had also increased the times of the North-South green light by about 15 seconds while reducing the East-West light times by 15 seconds during peak hours of traffic.

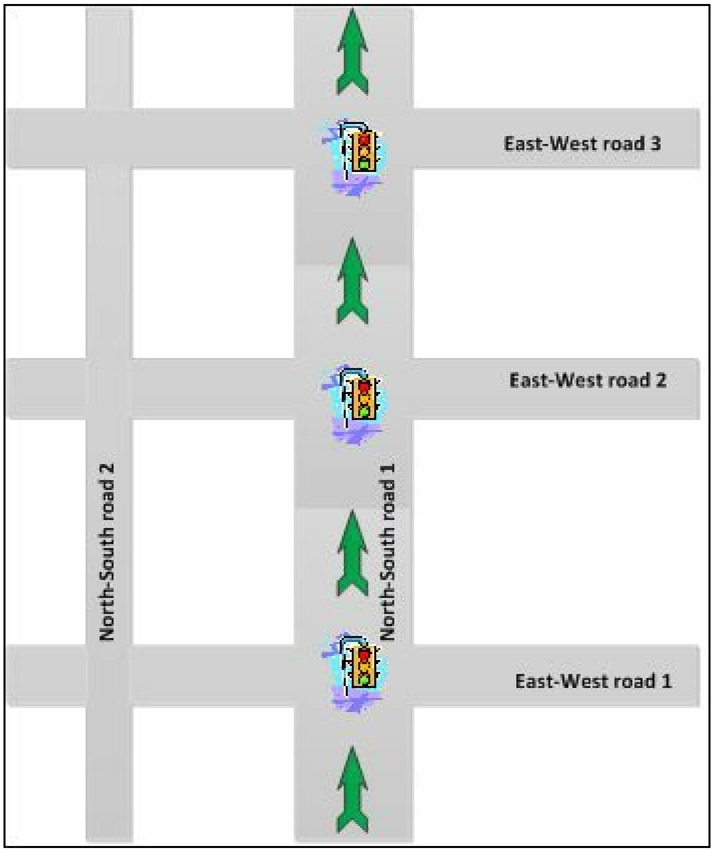


Figure 2.0

This was great for me and other drivers. It did not have any negative impact because the East –West traffic was minimal during peak hours on week days.

All was well until people noticed that the same rules applied on a Saturday or a Sunday or any other holiday. As a result, the East-West traffic was bad in the morning and evening on holidays and weekends because the lights were giving priority to the North-South traffic even when there was minimal traffic.

About two weeks after this change in lights, there was an accident during morning rush hours on the corner of E-W road 2 and N-S Road 1. (See diagram below) Because of the accident, the N-S road traffic was diverted to the side road to bypass the stretch that had the accident.

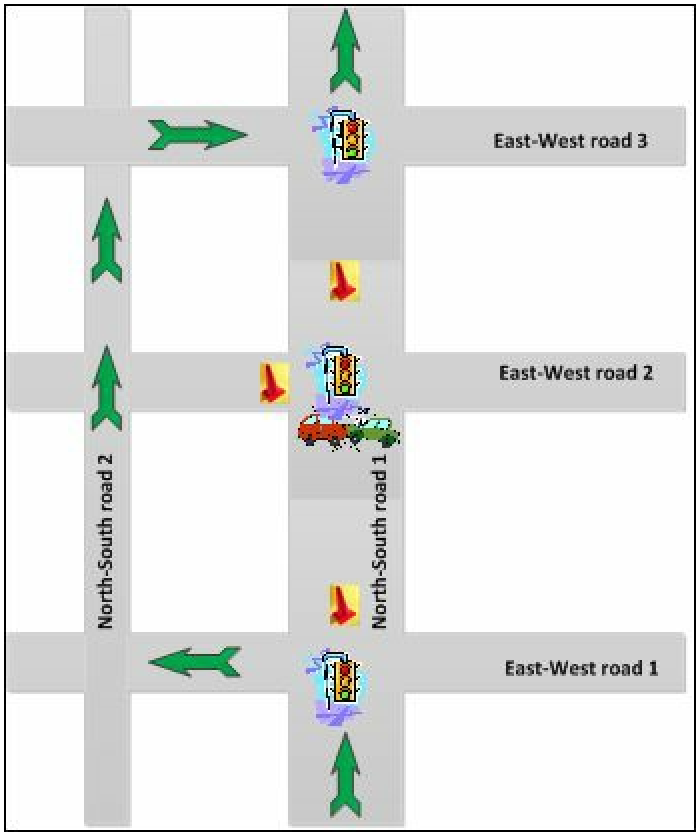


Figure 2.1

This created a lot of problems because the diversion took the traffic from E-W road 1 to NS Road 2, and then back on the main road using E-W road 3. This entry back to the main road was slow by design and all the drivers were late by at least an hour or more that day.

What if they had cameras installed on all these traffic lights, they could see that the traffic light on EW Road 3 was the traffic choke point? There were no cars coming on the N-S road. That is something I could only wish for because this is a huge infrastructure investment. You need to install cameras at all the intersections and then have someone or an application monitor those cameras and change the duration of lights according to traffic.

So what am I saying with this story?

If you are thinking in terms of the following then you understand the concept of SDN.

Cars and **Trucks** = **Packets** on the network

Roads = **Data plane** of the network

Traffic **light controller** = **Control plane** of the network

Me, the **driver** in the car = the **application** (because this road infrastructure is there to facilitate my transportation)

SDN is an architecture under which the applications (me sitting in that traffic jam) can ask the management plane (traffic light controller) for a particular treatment (priority, least hops, lowest latency) across the network. This management plane has the central view (cameras installed at each intersection) so that it can dynamically direct traffic depending on the congestion and multiple paths.

SDN formal definition

I thought that it would be wrong if I do not give you a formal definition of SDN. However this turned out to be no trivial task. There are numerous definitions out there but not all the players agree on one definition. Since Open Networking Foundation (ONF) (www.opennetworking.org) now manages SDN, I will give you their definition.

"Software-Defined Networking (SDN) is an emerging architecture that is dynamic, manageable, cost-effective, and adaptable, making it ideal for the high-bandwidth, dynamic nature of today's applications. This architecture decouples the network control and forwarding functions enabling the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services."

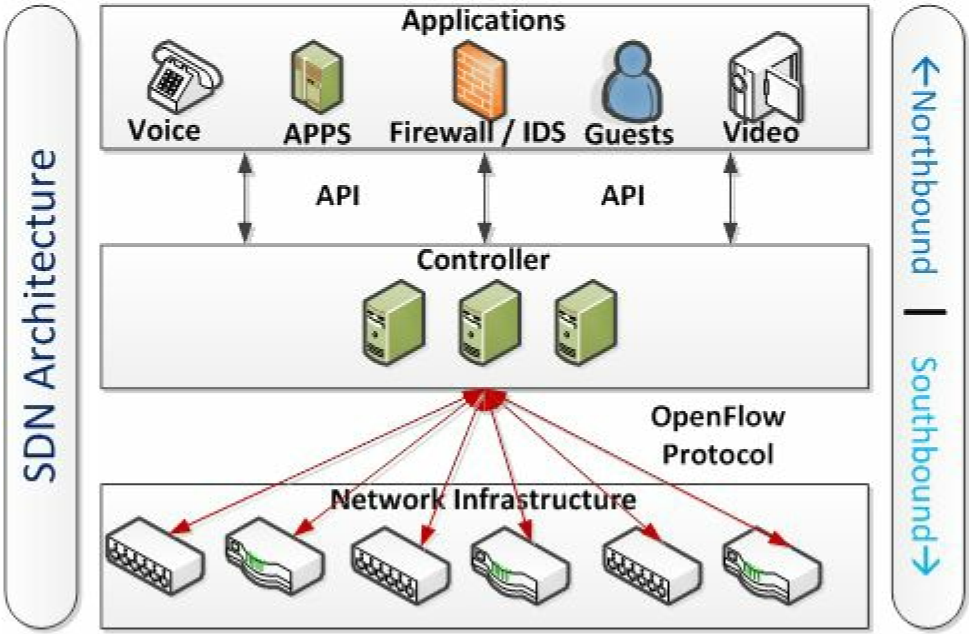


Figure 2.2

Network Infrastructure – This encompasses the network devices such as **routers and switches**; both **physical** as well as **virtual**(hardware or software only).

Controller – This encompasses the software based on a centralized controller which could be on a **server** that talks to all the devices in the network infrastructure using open API's. This also means any controller will be able to control any network hardware.

Note: *A centralized controller does not mean a single controller. It can be logically centralized controller which exists in multiple datacenters with all the redundancy that we have for critical servers.*

Applications – This encompasses the variety of applications for which the network exists. This includes voice, video, enterprise applications, network services like guest access and intrusion detection. These applications can talk to the controller using open API's to give them what they want. For example, voice traffic may ask the controller to have it treated with least latency while an enterprise backup server may tell the controller to give it bandwidth whenever it is available.

***Note:** Many blogs and articles also refer to **northbound** and **southbound** API's in SDN. The conversation between the Controller and the Applications is referred to as Northbound and conversation between the Controller and the Switches is referred to as Southbound.*

For the sake of simplicity, in server virtualization, there is a big bank of memory, a bank of CPU processing power, and a huge amount of storage on the network (SAN). A virtual server can be either statically or dynamically assigned each of these resources at the appropriate level to provision and bring a virtual server up in a matter of minutes. Another good thing is that once this server has done its job, it can be decommissioned just as easily, and then you can make the CPU, memory, and storage space available for reuse somewhere else instantly.

Any network device today consists of the data plane, control plane, and the management plane. Right now all these planes are in the same device. Simplistically speaking, a supervisor module has the control and management plane while the switch fabric or back plane will be the data plane.

Imagine a situation where the data plane is separated from the control plane. Can you think of an example of this? Many of you may say that this is already used in the wireless world where a central controller controls all of the wireless access points; and you are correct. Network virtualization is already there amongst us now, but how about extending this to the routers and switches?

In the SDN vision, we are taking the intelligence out of the switch to make it a dumb device like the newer AP's being used with wireless controllers. All of these dumb switches will be controlled by a central controller. Since this controller will have an overall view of the network, it can dynamically adjust flows of traffic according to layer 2, layer 3, layer 4 or any other attribute in the data packet.

Furthermore, if we are using an open protocol, then this switching hardware need not be manufacturer specific. As a result, we will have dumb silicon that will have ASICS whose job will be to send packets across at wire speed as per the commands from the controller. It will become very easy to add/remove/maintain hardware to the network.

By now, the name software defined networking should make much more sense to you. It is the software which resides on a server, which is also the control plane, will define how the data plane (ASICS on the switch or the switch fabric) forwards traffic.

The protocol that is used between the controller (control plane) and the dumb switch (data plane) is called "OPENFLOW." (Figure 2.2) This is not the only protocol that is being used for this purpose but this is the one that is an open standard and is maintained by openflow.org.

I have seen SDN and OpenFlow being used synonymously and that would be wrong. Although very popular, OpenFlow is just one of the protocols that make SDN possible. Cisco's onePK is another that has tight integration with Cisco's hardware.

The key here is that you now have much greater control over traffic as it is seen as flows. As a result you can treat the flow of video, voice, email, IM and a screen being shared from the same computer (same IP and MAC address) differently from one point to the other on your network in a dynamic and secure fashion.

Allow me to give you another example to fortify what I have said.

Last summer, I was going from Alexandria, VA to Woodbridge, VA to meet an old friend. I hopped on to highway 495 and the ride was smooth since it was 11 AM on a Sunday morning. I was testing my new GPS which can also receive traffic information. As soon as I got on to highway 95, I started seeing the red tail lights of cars stopping in the distance. To my surprise, my time to destination did not change in the GPS. I slowed down and turned the radio on to check the traffic report, but I heard the GPS asking me to take the nearest exit. This was even more surprising as I knew that this was not the way to my friend's house. The radio told me that the location of the accident was about 3 to 4 miles ahead on the freeway. It seemed like I was about to waste another 20 minutes of my life in a traffic jam.

I was thinking about turning the GPS off and returning it as it was pointing me to the wrong exit; but, since I was testing it, I decided to play along. I took the exit, and after few minutes of zigzagging on side roads the GPS got me back on the freeway ahead of the accident. As a result, I was able to reach my friends house in time.

I was just in awe of this new GPS. It had intelligently seen the traffic jam and guided me to a side road which I never knew anything about, and made me bypass the accident entirely.

To me, that GPS was the SDN controller that was using its intelligence to direct my car (the data packet) across the path of least latency. Since I (the application) told the GPS to take me through the fastest route it made a dynamic decision to direct me towards a path that is traditionally slower than the freeway but was faster in that particular situation.

I (the application) could have alternatively chosen the shortest path or the toll free path to the destination and the GPS (SDN controller) would have made a dynamic decision accordingly and directed the car (data packet) based on my request.

Note: I used these real world examples of traffic lights and GPS to explain the concept behind SDN but this does not mean it can only be used for QoS. Preferential treatment of packets is just one of the many uses. SDN can keep backup traffic on underutilized links only and critical

application traffic on the fastest links. Traffic engineering can be easily done by directing flows to the designated path on the network. SDN can be used to perform all the functionality of current networks and much more.

Continuing on my examples, if applications can talk to the SDN controller, then the application needs can be served dynamically on an ongoing basis. However, this will be done within the constraints of the rules set for the controller. The controller has a finite amount of resources such as bandwidth and queues which it has to juggle in order to fulfill the application's demands.

For example, the application can tell the controller that this data traffic should not leave the building or should stay on the internal network only. Applications like voice and video can also ask for more bandwidth and better treatment (QoS) for their packets.

I am voice call and I need least jitter and delay
Make sure that you tube video gets priority at 9:30 AM today as we are launching a new campaign
Sure, I can do that!!!
Make sure data traverses our internal network only
This video broadcast is for in country customers only.
Make sure this message does not traverse the networks of country X, Y and Z.

Chapter 3 - A brief history of SDN / OpenFlow

SDN is so new that the word history doesn't even rhyme with it. However, things move at a lightning pace in the IT world. Things that are as young as two to three years old can be referred in a historical perspective.

The concept of separating the control plane and data plane is not new; this has been used in the telephone industry and has been kicked around since the 90's for networking. Here are a few interesting facts for you:

- Ipsilon networks proposed GSMP (RFC 1987) General switch management protocol in 1996. This had a controller and was a flow based model for ATM switches. This company was not very successful and eventually bought by Nokia.

- In the year 2000, ForCES (forwarding and control element separation) was there and you can find a couple of RFC's on this.

- In 2004 RCP (Routing control platform) was proposed to overcome the fully meshed nature of iBGP. The proposal was to have a centralized platform separate from the forwarding plane that can collect information and performs route selection on behalf of the routers and presents to them the best routes.

- Ethane came in 2007 which coupled simple flow based switches managed by a central controller which controlled the flows.

OpenFlow is an open source project that was the result of a six-year research collaboration project between Stanford University and the University of California at Berkeley. Here are some interesting events related to OpenFlow.

- Developed at Stanford and OpenFlow ver 1.0 was announced in Dec 2009

- March 2011 Open network foundation (ONF) was launched.

- Oct 17-19 2011 Stanford University hosts the first Open networking summit

- Feb 2011 OpenFlow ver 1.1

- October 2011 First Open networking Summit

- Feb 2012 OpenFlow ver 1.2

- April 2012 Second Open networking Summit

- June 2012 OpenFlow ver 1.3

- April 2013 Third Open networking Summit

- ONF has the support of 100 members and is continuously growing. The interesting thing is that the members are not only the vendors who are developing this, but are also users like Google and Facebook.

- ONF is doing plugfests to make sure that the controllers and the switches from different vendors work together. The third plugfest was in June 2013.

- ONF has also launched an OpenFlow conformance testing program which lets the vendors demonstrate their compliance to the OpenFlow specification.

CHAPTER 4 - CAM, TCAM AND OPENFLOW

Before we go deeper into SDN, you need fully understand the TCAM which is a better version of CAM. So, let us start with CAM. It is very important that you have intimate knowledge of TCAM because this is where the SDN controller will write to, to affect the flow of packets in the data plane.

RAM and CAM

Traditionally, content is stored in RAM (Random Access Memory) at a unique location. If an application wants to access that content, it is given the unique location of that information to retrieve content when needed.

Input for a memory operation is the location of the content that we are interested in and Output is the content at that location.

In other words, it is like the X on a treasure map, where "X" marks the location of the treasure. When you (a computer application) look for the treasure (stored information), you look at the map marked with an "X" (unique memory address). You go to that "X" (memory address) and get your treasure (content stored at that memory address).

Similarly a hyperlink that directs you to the PDF version of the introductory chapters of this book is another example. You click the link (which is your unique address) and you retrieve the information (First few chapters of this book in pdf format).

CAM is something that works in an opposite way. You input the content (mac address) and the output is a unique location (port number associated with that mac address).

Input for a CAM operation is the content that we are interested in and output is the unique location of that content.

For example, if you want to search for this book "SDN for beginners" on your favorite search engine it will point you to certain book sites who are selling this book. If there are multiple booksellers selling this book (which I hope will be the case) then your search engine will use its ranking algorithm to give you the best search on top. You can then go that particular website and look up the book and get the details such as author, price, and date of publication etc. and buy it. If a search for content in CAM leads to multiple matches, then the built in priority encoder sends out the first match as the outcome. This is explained in more detail below.

CAM

A CAM (content addressable memory) table is part of the memory of a switch, which keeps track of which mac address is received on which port. This memory is extremely fast, power hungry and expensive. For that reason there is a limited amount of CAM in a switch. CAM's are fast because they complete a lookup in just one go (one clock cycle). Let me illustrate this with a simple example. (If you already know how a CAM table works in a layer 2 switch please proceed to the next section of TCAM.)

Consider the network below.

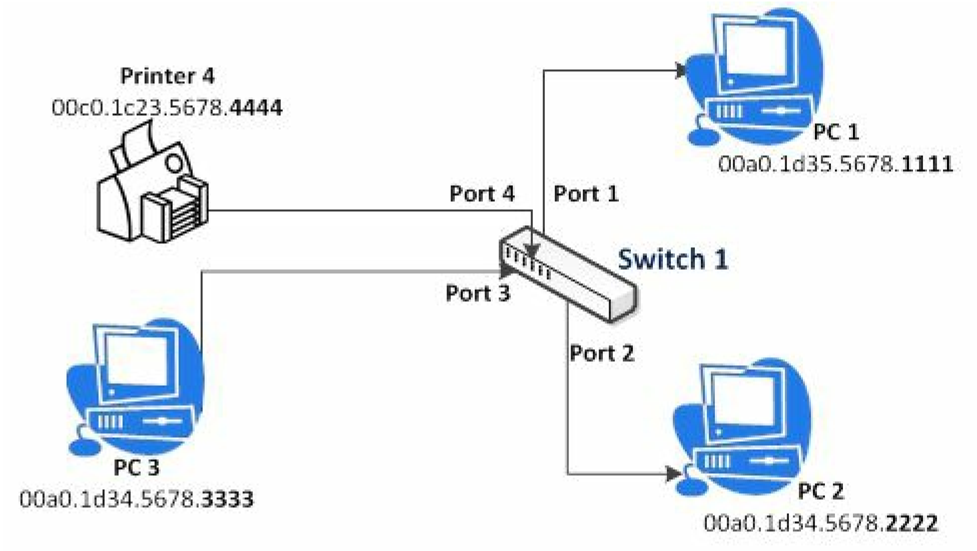
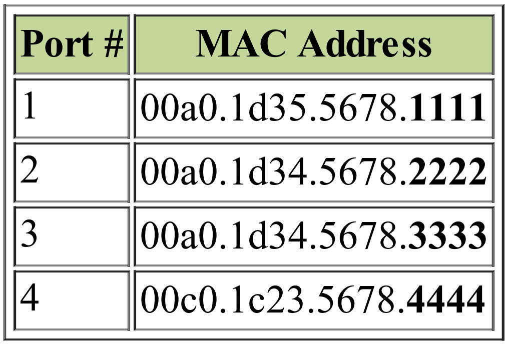


Figure 4.0

The fully populated CAM table for the switch will look something like this.>



Port #	MAC Address
1	00a0.1d35.5678.**1111**
2	00a0.1d34.5678.**2222**
3	00a0.1d34.5678.**3333**
4	00c0.1c23.5678.**4444**

Figure 4.1

The switch has learned the MAC addresses of all the connected devices and associated the respective port numbers from the Ethernet packets that were received on its switch ports from these devices. If PC3 wants to print on Printer 4, it will send packets with a destination MAC address 00c0.1c23.5678.4444. The switch will look up this MAC address in its MAC Address Table/ CAM table and will see that this MAC address is on port number 4. This lookup will result in the packet being sent on port number 4. Something similar happens when PC1 wants to talk to PC2 and so on. There are other actions that are also taking place such as updating the timers to expire the cam entries and looking up vlan numbers, but I have kept it at a high level only.

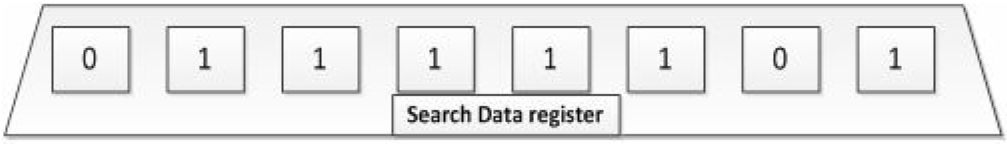
The CAM that I have been talking about is a binary CAM. It is called binary because it understands 0's and 1's only. All these values go in a table which is looked up at one go and results in either a match (the packet being forwarded to the right port) or a no match (it is flooded to all the ports).

TCAM

For the sake of understanding, I would say that binary CAM is good for exact lookups. What happens when there are wild card bits in a lookup like in an ACL which is matching on 10.1.x x with a mask of 0.0.255.255? As you already know, the number 255 is the 'do not care' part in the mask, and this is where Ternary CAM or TCAM (I think of it as TRI CAM because of the third state) comes in. TCAM can store three states which are 0, 1 and *. The last state of "*" is called the mask which means it can be of any value 0 or 1. This corresponds really well with the mask on an ACL.

The easiest way of explaining how a TCAM is with an example.

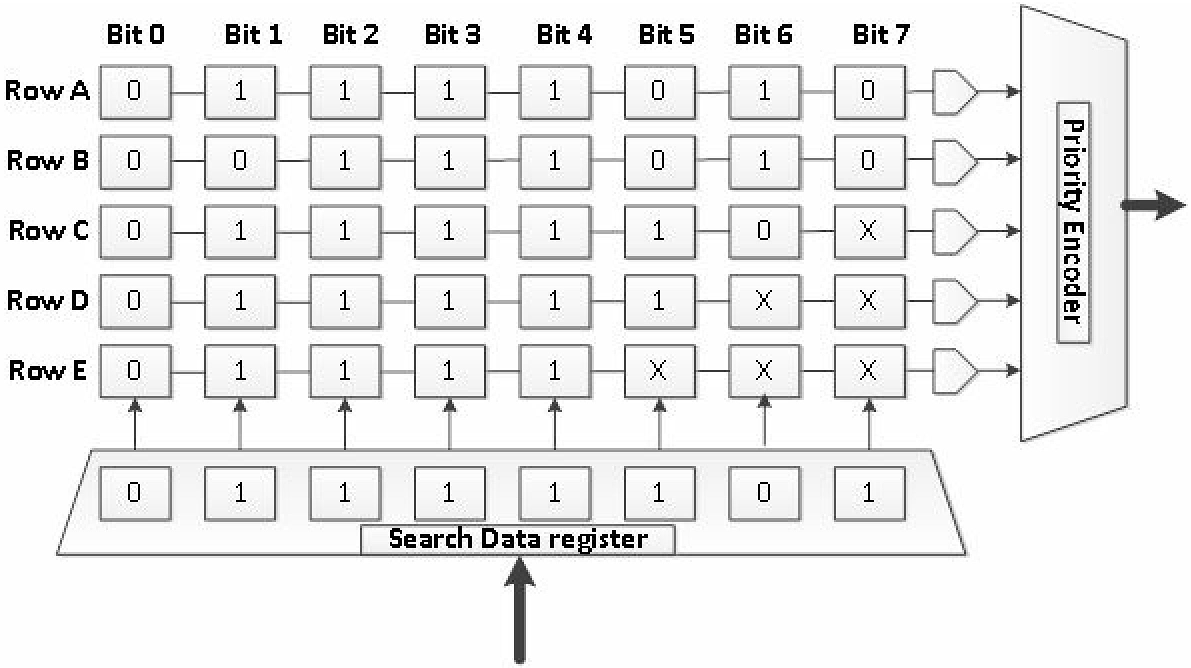
If the input to the search register is



0	1	1	1	1	1	0	1

Figure 4.2

As this lookup starts (see figure below) all 5 lines (Row A to Row E) going to the priority encoder (on the right) are set to high (this is why it uses so much power). All the rows are looked up in one clock cycle (that is the reason it is so fast).



	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Row A	0	1	1	1	1	0	1	0
Row B	0	0	1	1	1	0	1	0
Row C	0	1	1	1	1	1	0	X
Row D	0	1	1	1	1	1	X	X
Row E	0	1	1	1	1	X	X	X

0	1	1	1	1	1	0	1

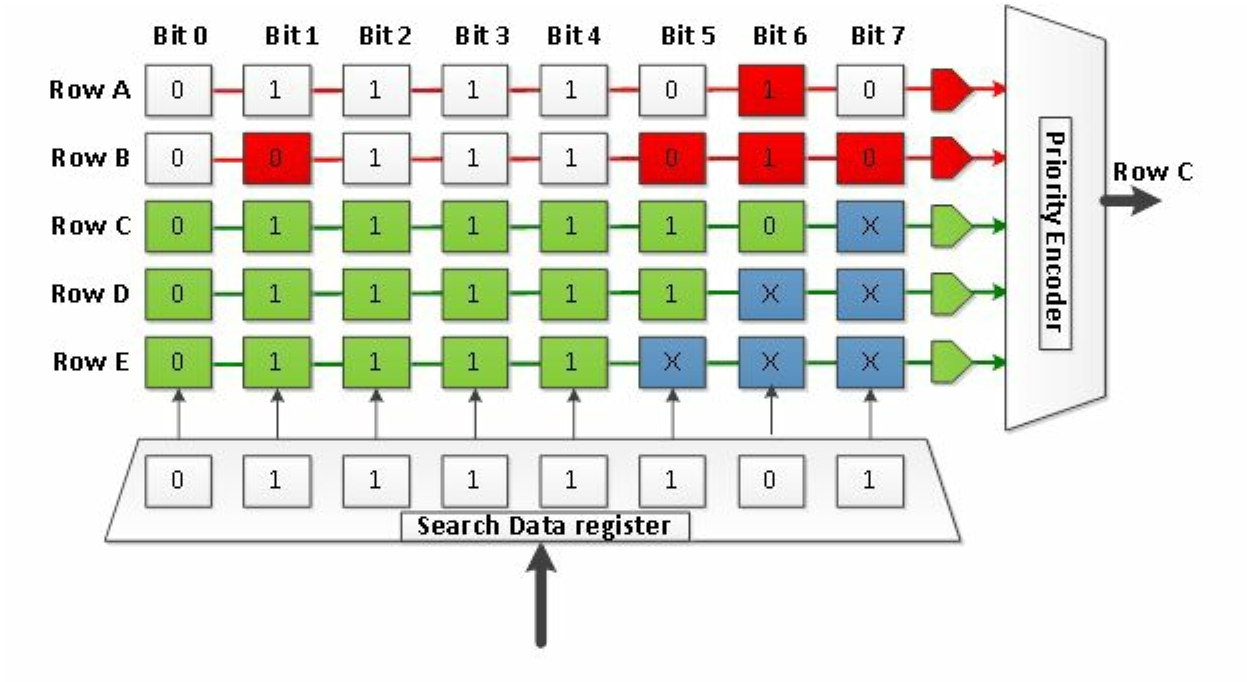
Figure 4.3

This is how this process works.

Row A has a mismatch on **Bit 6**. (See figure 4.3 below) Even one mismatched bit makes the line go from high to low (non-conducting state).

Row B has a mismatch on **Bit 1, 5, 6 and 7**. So this line also goes low (non-conducting state)

Row C, D and E are a match because all values match except for that of "X" (don't care values).



	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Row A	0	1	1	1	1	0	1	0
Row B	0	0	1	1	1	0	1	0
Row C	0	1	1	1	1	1	0	X
Row D	0	1	1	1	1	1	X	X
Row E	0	1	1	1	1	X	X	X
Search Data register	0	1	1	1	1	1	0	1

Figure 4.3

Since "X" is the joker that represents both 0 and 1, all three lines stay high. The priority encoder now gets these three inputs and outputs the **first match**, which is **Row C**.

If you are thinking that this is exactly how the ACL's work, you would be right. The first matching rule of an ACL is applied to the packet and the next action of dropping or sending it out is done. The rest of the lines in the ACL are not used. That is the reason TCAM is used extensively for matching ACL's for things like QOS, Policy based routing etc.

Now you know how the latest devices can do wire speed forwarding in spite of long ACL's. The ACL's are crunched and programmed into the ASICS, which use TCAM to make ultra-fast forwarding decisions.

TCAM is expensive, so there is a limited amount of it in a device. That is the reason you need to use the SDM template (switch database manager) on a Cisco 3750 switch to set the template according to the role of the switch. The TCAM for a Cisco 3750 switch is shared between ACL's, QOS ACL's, VLAN ACL's, L3, and L2 forwarding tables. If you know that your switch will be used for L2 only then all the TCAM space allocated by default to L3 entities is taken away and reallocated to L2 entities.

TCAM and SDN

You must be wondering how this intimate knowledge of TCAM helps you in understanding SDN.

In a traditional network device like a router, there is a RIB (Routing information base) and a FIB (forwarding Information base). Complex algorithms are run on the RIB to get the best route in the FIB. So, for example, if the best route to a network is using a specific next hop, then the FIB entry corresponding to this RIB entry will have the outgoing interface computed along with all the data needed for a layer 2 encapsulation already there. This facilitates wire speed forwarding. RIB usually stays in software while the FIB is put in the fast hardware on the device.

In case of SDN, the controller is a separate entity, which runs the complex algorithms to compute the best path. The results of these computations are directly sent to the TCAM, which populates the FIB (forwarding table) to facilitate the fast flow of data. TCAM is the how the SDN controller influences the hardware in the switches for data forwarding.

Since the controller has a central view of the network it can dynamically redirect traffic in case of congestion, link interruption or other factors like user policies or even on the request of an application.

The controller is like that GPS, which tells you to take the most efficient route and dynamically changes the route in case there are any accidents, road construction or road closures.

This deep understanding of TCAM is very essential for SDN to work, so that is the reason why I dedicated a chapter on CAM and TCAM.

CHAPTER 5 - FLOW TABLE AND FLOWS IN OPENFLOW

This chapter has a lot to digest, so tighten your belt or loosen it if you have to :-)□.

So what is a flow in SDN land?

At a basic level, you can say that flow is packets going between a source and destination pair. A collection of flows is a flow table.

Flow Table

A flow table has three basic components.

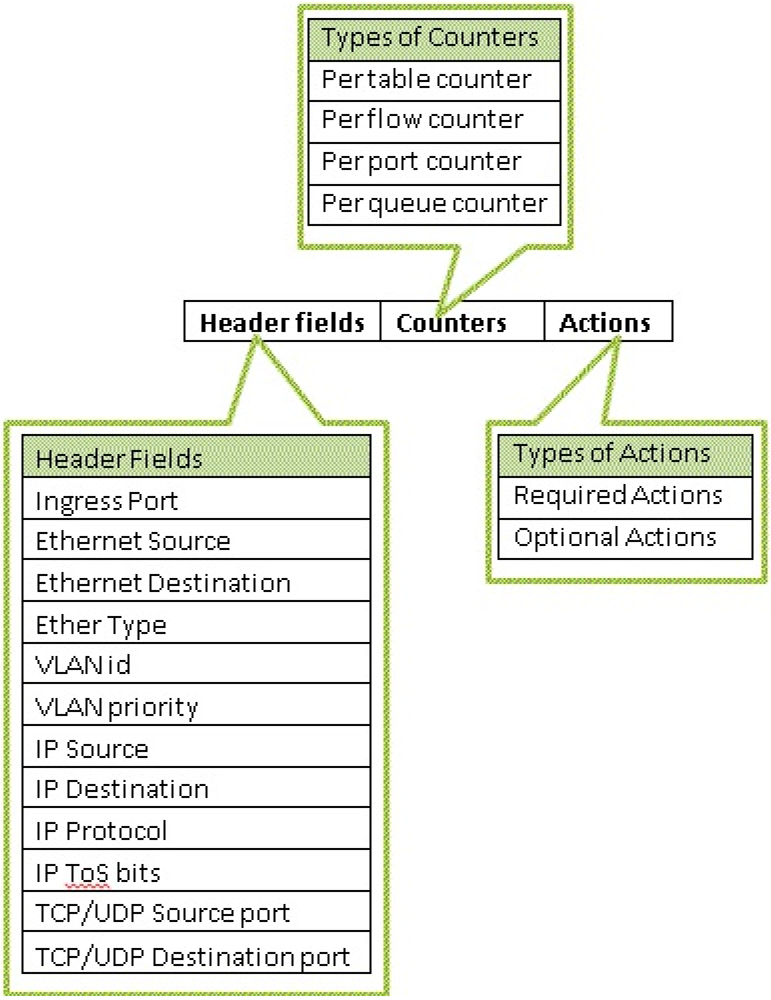


Figure 5.1

- **Header fields**: to match packet flows
- **Counters**: updated with matching packets in that particular flow

- **Actions**: taken on the packet flow based on the match in the header

Header fields, as you may have already guessed, are the VLANID, Source MAC Address, Destination MAC Address, Source IP Address, Destination IP Address etc. There are 12 tuples that can are used to match incoming packets with flows according to OpenFlow specification 1.0.0

Header Fields	Size in Bits
Ingress Port	Depends
Ethernet Source	48
Ethernet Destination	48
Ether Type	16
VLAN id	12
VLAN priority	3
IP Source	32
IP Destination	32
IP Protocol	8
IP ToS bits	6
TCP/UDP Source port	16
TCP/UDP Destination port	16

Ingress Port is implemented depending on the manufacturer starting with port number 1.

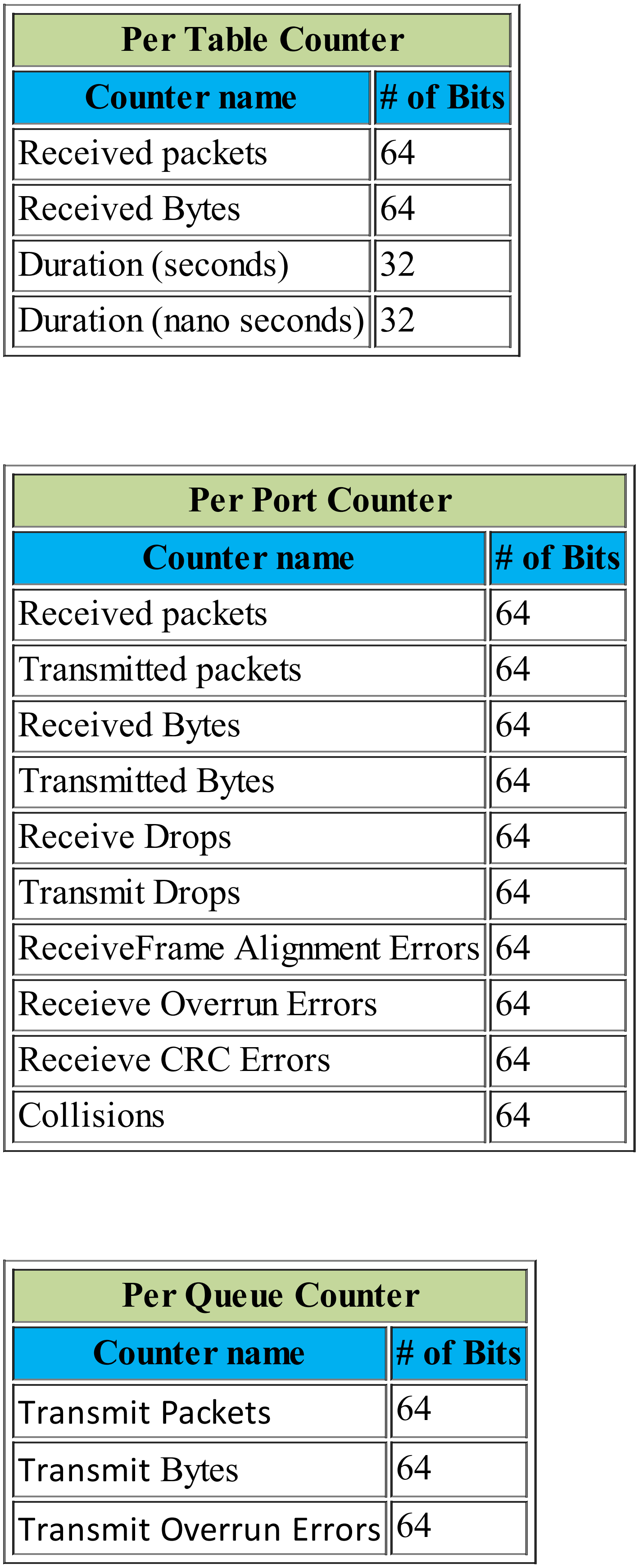
Figure 5.2

Flows can match one or more than one fields in the flow table.

Note: *A* ***Tuple*** *is a list in a specific order.*

Counters are maintained per table, per flow, per port and per queue. These counters are maintained in software and are required for a device to be OpenFlow compliant.

Per Table Counter	
Counter name	**# of Bits**
Active Entries	32
Packet Lookups	64
Packet Matches	64



Per Table Counter	
Counter name	**# of Bits**
Received packets	64
Received Bytes	64
Duration (seconds)	32
Duration (nano seconds)	32

Per Port Counter	
Counter name	**# of Bits**
Received packets	64
Transmitted packets	64
Received Bytes	64
Transmitted Bytes	64
Receive Drops	64
Transmit Drops	64
ReceiveFrame Alignment Errors	64
Receieve Overrun Errors	64
Receieve CRC Errors	64
Collisions	64

Per Queue Counter	
Counter name	**# of Bits**
Transmit Packets	64
Transmit Bytes	64
Transmit Overrun Errors	64

Figure 5.3

Actions

Each inbound packet goes through the process shown (Figure 5.4) below. This is something that we all are familiar with, but this simple process has far-reaching effects.

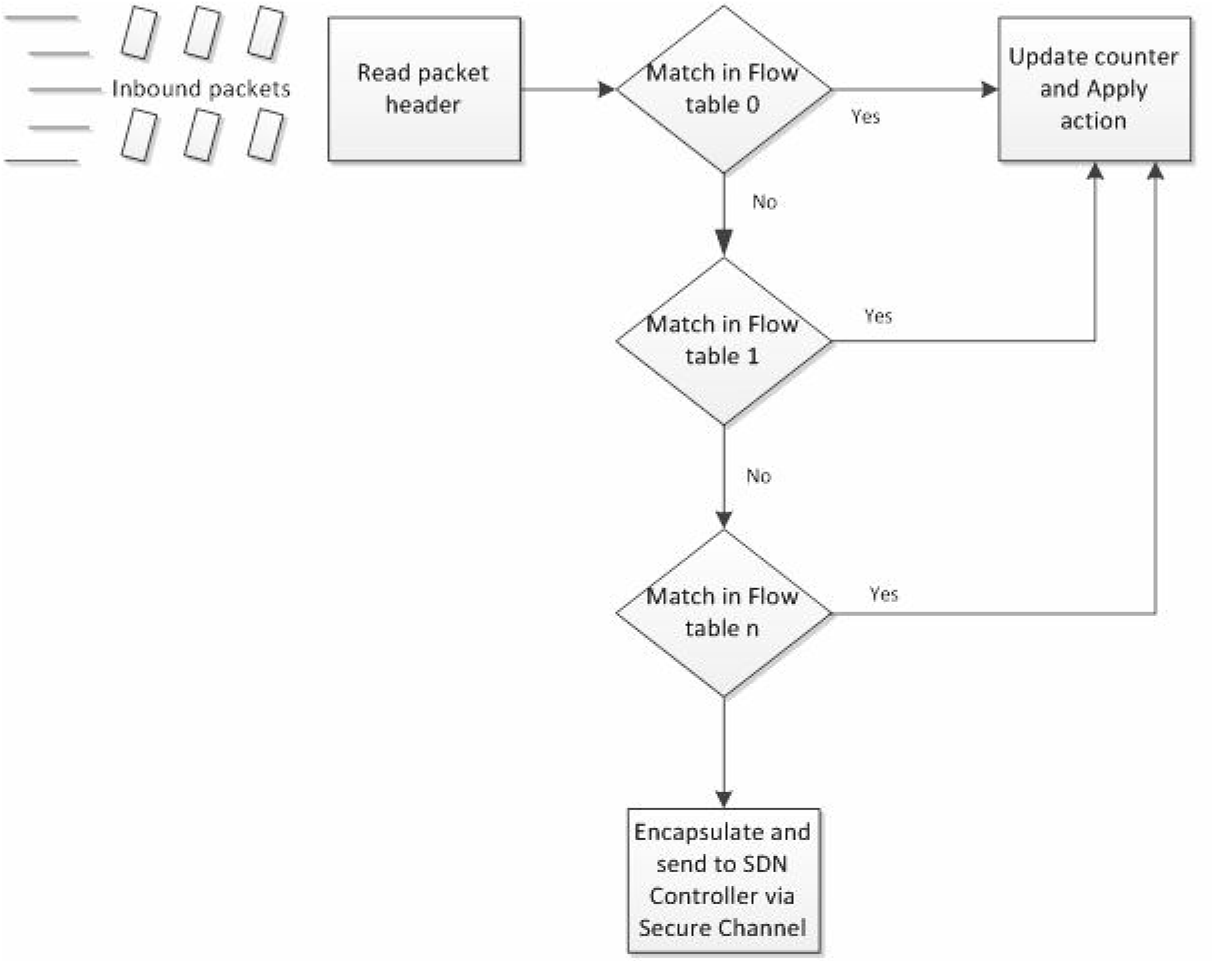


Figure 5.4 (this reflects OpenFlow 1.1 behavior)

For example, if you wanted to add a firewall to your network you will have to:

1. Choose, evaluate and buy the hardware.
2. Test it in your lab for suitability to your network environment.
3. Find a suitable change window to put this in your network.

If you have an SDN environment, all you have to do is add a list of actions on the controller and you are done. My mind is just saying WOW!

In SDN, every inbound packet is matched in the flow table using the Ethernet source and destination fields. The rest of the lookup depends on the packet type. For example, if the packet is ethertype 0x8100 (VLAN) then the VLAN id field and the VLAN priority fields are looked at. Similarly, if the ethertype is 0x806 (ARP) the lookup may include the source and destination IP Address fields.

***Note**: A flow table value of **ANY** matches all flows.*

A flow entry, which is an exact match with no wild cards, has a higher priority than the one with wildcards. Each wildcard entry has its own priority. If multiple entries with the same priority are matches then the device is free to choose the order of matching.

For every match two actions take place

- The counter for that particular match is updated.

- The action associated with that particular match is taken.

Every matched flow entry has zero or more actions associated with it.

Note: No forwarding** action means that the packet is **dropped

Port#	Src. MAC	Dest. MAC	Ether Type	Vlan ID	Vlan Pri	IP Src.	IP Dest.	IP Proto.	IP ToS	L4 Src.	L4 Dest.	Action
*	*	*	*	*	*	*	1.2.3.4	*	*	*	*	Port 1/1

Routing (*all packet to 1.2.3.4 are sent to port 1/1*)

Port#	Src. MAC	Dest. MAC	Ether Type	Vlan ID	Vlan Pri	IP Src.	IP Dest.	IP Proto.	IP ToS	L4 Src.	L4 Dest.	Action
*	*	*	*	*	*	*	*	*	*	*	21	Drop

Discarding traffic (Firewall:*all packets with 21 as destination port are dropped*)

Port#	Src. MAC	Dest. MAC	Ether Type	Vlan ID	Vlan Pri	IP Src.	IP Dest.	IP Proto.	IP ToS	L4 Src.	L4 Dest.	Action
*	*	*	*	*	*	4.2.3.1	*	*	*	*	*	Port 1/1,4/1

Mirroring *(traffic from 4.3.2.1 is sent to uplink port 1/1 and monitoring port 1/10)*

Figure 5.5

Actions can be of two types

- Required Action

- Optional Action

Required Actions

1. Forward packets to physical ports and these virtual ports

All – Send the packet out all port except for the incoming port

Local – Send the packet to a local port

Controller – Send the packet to the controller

In_Port – Send the packet out the incoming port

2. Drop any matched packet that has no action associated.

Optional Actions

1. Forward packets to these virtual ports

Normal – Process the packet using the traditional L2, VLAN, L3 processing.

Flood – Flood the packet along the minimum interfaces as specified by spanning tree.

2. Enqueue – Forwards packet through a queue associated with the port. This is used to provide basic QOS.

3. Modify Field – This is an optional action but it increases the usefulness of OpenFlow many fold. Using this action, you can change the VLAN or the Source/Destination Mac Addresses, ToS bit, TCP/UDP port numbers etc.

***Note**:The name "Normal" for forwarding action above is given for the traditional packet processing. This may be normal for today but would be old / traditional in a few years when OpenFlow is the new normal. I suppose this would be changed in the coming years.*

OpenFlow hardware

Based on the above actions OpenFlow hardware comes in two flavors, **OpenFlow-enabled** and **OpenFlow-only**.

OpenFlow-enabled hardware supports all the required actions and the optional actions. This means that the network device is a hybrid device that supports the present day traditional manufacturer's specific way of forwarding and the new OpenFlow way of forwarding packets. This also means that we will be seeing these devices in the interim for the coming years till everyone is convinced that SDN is the way to go and the world changes over to this new technology.

OpenFlow-only hardware supports only the required actions. This new hardware is coming out now, is based off the present OpenFlow specifications, and supports that only. We will be seeing this in our labs initially and then may be on small parts of our network.

OpenFlow Modes of operation

OpenFlow can populate the TCAM in two different modes.

Reactive Mode

This mode should be familiar to us network engineers. The first packet is sent to the controller and as a result, the controller inserts an entry into the TCAM, which results in a FIB entry. The subsequent packets have a match in the flow table now and are switched accordingly. This seems synonymous to first packet being process switched and the subsequent packets being fast switched.

Proactive mode

I am sure your mind is already thinking ahead about the proactive mode. You are right if you thought that in proactive mode you have to prepopulate the flow tables with all the network information. This could be done if you know the subnets, VLAN's, ACL's and topology information. This would eliminate the need for the slow packet lookup that is used in reactive mode.

Note: *To have your own OpenFlow switch please see* ***Appendix F*** *for instructions on how to make your own OpenFlow switch.*

CHAPTER 6 - OPENFLOW

OpenFlow and SDN are sometimes interchangeably used in the same sentence. This is because of the two being much intertwined to make SDN architecture possible. However, just to clarify, SDN is architecture (see Figure 6.1 below) while OpenFlow is the protocol that is used between the controller and the network infrastructure as seen in the diagram below.

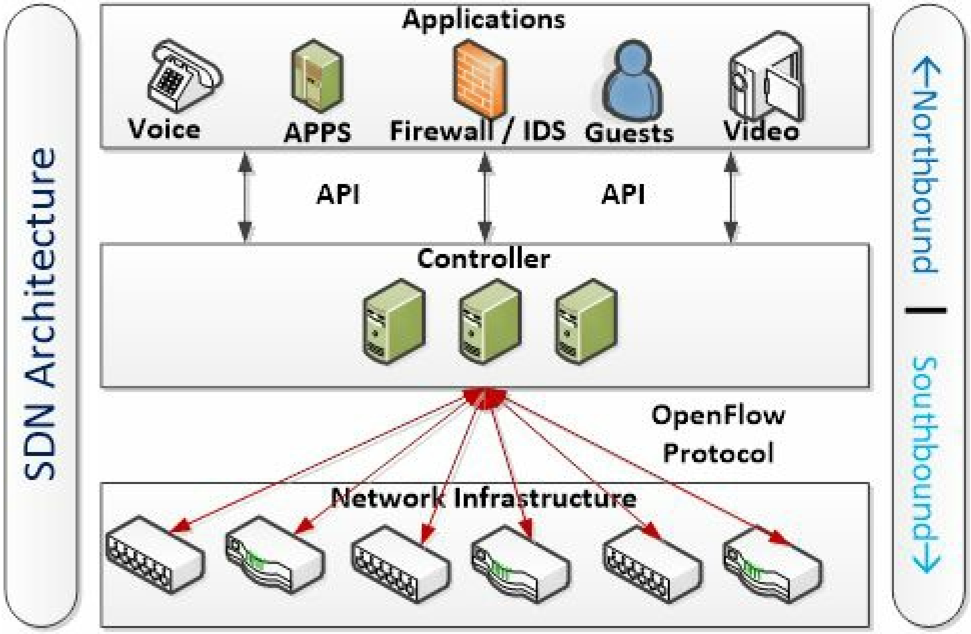


Figure 6.1

To put it into simpler terms, a controller speaks this language (OpenFlow protocol) with compatible network hardware. It is like an Android app that will work on all android devices whether it's from HTC, Samsung, Motorola, any other known or unknown manufacturer.

OpenFlow is an open network protocol that is designed to manage and direct network traffic centrally for routers and switches, which can be from different vendors.

OpenFlow protocol details

Like any other language, OpenFlow protocol has its own vocabulary which is used between the controller and the network infrastructure. However, this much easier to learn as compared to learning new languages that we humans speak.

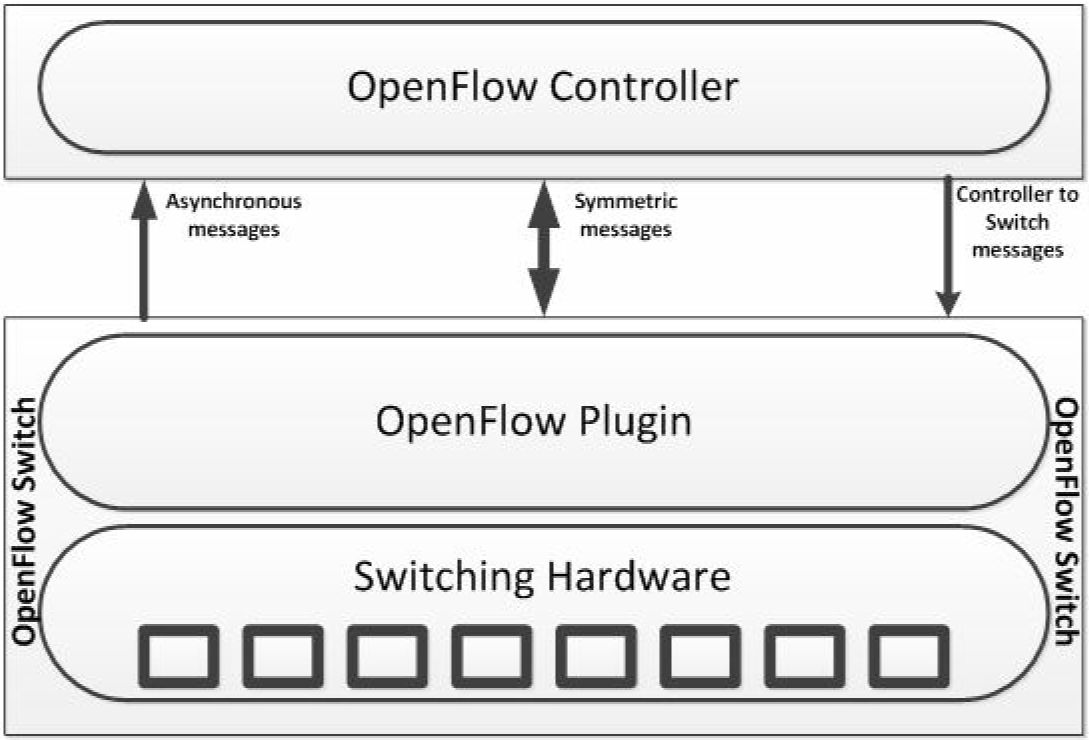


Figure 6.2

All the messages exchanged in the OpenFlow protocol can be divided into three major groups.

- Controller to Switch messages
- Asynchronous messages
- Symmetric messages

Controller to Switch messages – These messages are initiated by the controller and used to manage the switches, or to get information from the switches. These are of six different types.

Read State: Used by the controller to read the counters from the per flow, per port and per queue counters as mentioned in the last chapter Figure 5.3

Modify State: Primarily used to add, delete, or modify flows in the flow tables on the switch.

Send packet: Used for sending packets out of a particular port on the switch.

Barrier Request/Replies: Used for receiving notifications of completed operations from the switch. If a barrier request is received from the controller on the switch, the switch must complete all the pending tasks it has before it received the request. It creates a barrier in the switch so that it can jump to the next task only when it has finished what it has on hand. As soon as the switch completes its tasks, it sends a barrier reply back to the controller and starts working on its next tasks.

Features: This is a feature request to the switch after a secure connection is established between the controller and the switch. The switch replies only to the controller with the features and capabilities it can support.

Configuration: Used by the controller to get and set configuration parameters in the switch.

Asynchronous messages – These messages give the switch a chance to speak freely to the boss (controller) without asking permission to speak. This is where the switch can inform the controller of dropped packets, or an interface going down. These are of four different types.

Port Status: Any change in the status of the port is sent the controller. Status change includes not only the port going down but a change in 802.1d (spanning tree) status.

Packet in: This may seem a bit confusing, but this message is sent to the controller only if there is a packet coming into the switch that does not match any flow entries (which means it has to be sent to the controller) or the packet matches the rule that the packet has to be sent to the controller.

Flow removed: Any flow entry added to the flow table has an idle timeout and a hard timeout value associated with it. So whenever a flow is removed which could be because of idle timeout, hard timeout, flow modification message a “flow removed” message is sent by the switch to the controller. Something to be aware of here is that a flow modification message can specify if the controller wants to know about this flow being removed or not.

Error: As you can guess this message is used by the switch to inform the controller of any errors it sees.

Symmetric messages – These are bi-directional messages that are sent by the controller or the switch to each other without any solicitation. For example, hellos or echo requests can be sent without any solicitation and these are replied by the receiver. These are of three different types.

Hello: Like many other network protocols, hello messages are exchanged between the switch and controller on startup.

Echo: We are all familiar with Echo request and Echo reply messages. Both of the sides can initiate these types of messages and are obligated to reply if they receive one. These messages give an idea about the latency of the switch – controller connection.

Vendor: A vendor messages provide a standard way for switches to tell the controller about the additional functionality that they offer. This message is also used for trying out new features.

Chapter 7 - OpenFlow in action

Now that you are aware of all the messages that make this protocol work, we are ready to see it in action and checkout how this protocol makes it all work.

New Boss in Town

Connection between the controller (the boss) and the switch is established using a known fixed IP address and is secured using TLS (transport layer security). The connection is initiated by the switch (you have to go up to the boss and say hello□) on startup to the controller, using the fixed IP address and TCP port number 6633. The switch and the controller authenticate mutually by exchanging certificates, one from each side.

Who Are You?

As soon as the connection is established, hellos are sent from both the sides. The hello message also contains the highest version of OpenFlow protocol that each one of them can support. This leads to them operating at the lowest version compatible with both sides. Just in case the lowest version supported on one side cannot work with the other, an ERROR message is sent with the details is sent and the connection is terminated.

What Can You Do For Me?

Once the protocol version (the language that the Boss and employee can understand) is established, the boss asks the employee about their capabilities. A "**feature request**" packet (see details in chapter 6) is sent to the switch and it replies back with its features and capabilities. This is followed by a "**Get configuration request**" through which tells the controller about the current switch configuration. There are other requests like "**Stats request**", "**Port status**" and "**Vendor**" can also be exchanged. You will see this in one of the labs (see lab xxx in the hands on section) where you will capture and analyze a Wireshark packet capture of this communication.

Where Are You Located?

The last thing the controller (boss) needs to know is the location of the switch so that it can direct traffic accordingly; this is done using LLDP (link layer discovery protocol). The controller creates an LLDP packet and asks the switch to replicate it across all its ports. There is a rule already set by the controller that all LLDP packets are to be sent to the controller. As the new switch sends out the LLDP packets, the neighboring switches receive those packets and send them back to the controller.

Thus, the controller knows the topology of the network right away.

After this is done, there are echo request and echo reply packets between the switch and the controller.

The Boss is Un-Reachable

In case the connection between the switch and the controller is interrupted because of an echo request not being replied or a TLS session timeout or any other reason the switch tries to contact one or more backup controllers. The order in which the contact is made is not specified in OpenFlow.

If attempts to contact the controller fail, the switch must reset the TCP connection to the controller and go into emergency mode. In this mode, the switch flushes out all entries from the flow table except for the ones that are marked with an emergency bit.

Once the switch establishes its connection, back with the controller the emergency flow table is still being used. Now it is up to the controller to flush the flow table completely and add all new entries or it can just add to the emergency flow entries.

***Note**: When the switch starts up, it is in an **emergency mode** until it establishes a connection with the controller.*

The Boss is Back

As soon as the switch sees the controller, it is ready to work based on what it gets from the controller. The controller may remove all the emergency flows and install new flows or it may choose to add on top of what is already there.

Let us assume that a new switch comes on the network and after the initial packet exchange as described earlier there are no flows installed in this switch. Now, a packet comes in from a connected host. As soon as that packet comes in, it is matched to the flow table. Since there are no flows it results in a table miss and this packet is sent to the controller (hey boss, I don’t know what to do with this). The switch sends the packet encapsulated in OpenFlow to the controller. The controller, in return, initiates the ARP process and if there are no matches in the table it is considered a table miss and the packet is sent to the controller as “Packet in”. A regular ARP request-reply follows and then the flow is put in the flow table.

CHAPTER 8 – OPENFLOW VERSIONS

I have added this chapter here for the sake of completeness of this book. The different versions with the detailed specifications are at www.opennetworking.org. I have also added links below for your convenience.

OpenFlow Switch Specification 1.3.2 (Apr. 25, 2013 – currently under ratification)

OpenFlow Switch Specification 1.3.1 (Sept. 6, 2012)

OpenFlow Switch Specification 1.3.0 (June 25, 2012)

OpenFlow Switch Specification 1.2 (Dec. 2011)

OpenFlow Switch Specification 1.1.0 (Feb. 28, 2011)

OpenFlow Switch Specification 1.0.0 (Dec. 31, 2009) | Errata v1.0.1

Just to give you an idea, the latest versions of OpenFlow now has support for IPv6. It also supports three types of ports (Logical, Physical and reserved). There is provision for tunneling which can be used in datacenter deployments or VPN deployments. Features like bandwidth management, QoS and per flow metering have been added.

The default behavior in OpenFlow 1.0 is to send all unknown packets to the controller. This has been changed to drop the packet unless specific rules are there to send the packet to the controller.

ONF has decided to freeze the major version at 1.3 right now so that industry can catch up to it. It wants to give the industry a stable target that the manufacturers can aim for.

CHAPTER 9 – SDN ADVANTAGES

SDN technology promises many advantages for the current networks.

1. Centralized control and management of networking devices.

2. Vendor neutrality.

3. Very agile and flexible network that can change using automation by the use of common API's. These API's hide the underlying network details by abstraction of the network layer from the applications that are using it.

4. Greater reliability because of centralized management and automation of management of network devices. This will result in lesser configuration errors, faster implementation of network wide changes and will result in a consistent network policy being enforced.

5. Very granular network control, as you now have the ability to inspect and affect singular flows if needed. Flows can be controlled and influenced by their layer 2 through layer 7 attributes.

6. Making a network that adjusts to the user's needs instead of users adjusting to the network. This results in a greatly improved user experience.

7. Much faster innovation and improvisation, as this will done at the software level and will be vendor independent. This will result in more enterprises, individual developers and software vendors using the common API to bring innovative services and drive revenue.

8. Very easy to deploy complex features like custom load balancing using special user defined algorithms because it will be a software plugin or software modification only.

9. Faster deployment of new features because you will not have to update/upgrade every device on the network individually. There will be no dependence on the hardware manufacturer for release of newer features.

10. Centralized and detailed view of the network also results in more efficient utilization of network resources.

11. All of the above advantages will feed on each other and will let us build networks at a scale that can't be imagined now.

Each of the above will be utilized in different ways by enterprise networks, datacenter networks and service provider networks.

CHAPTER 10 - SDN FOR THE ENTERPRISE NETWORK

SDN will have a great impact on enterprise networks. It will solve many problems and will make it much easier to tackle others. There is a lot that can be said here, but I will keep this concise like other chapters in this book.

Problem: Each enterprise network has different parts of the business with different needs. For example, the office side of the network may have emphasis on collaboration tools like WebEx and Sharepoint, the ecommerce side may have a total emphasis on web development, security, the design side of the business may need those 40Gig and 100 Gig connections to share huge CAD files on the fly and the call centers will be all for the application serving customers and voice traffic.

Solution: Network slicing. SDN allows you to slice your network into pieces which can have different needs and policies coexist with each other without any interference. Flows for each of these business units can be separated in a granular fashion and given the desired treatment.

Problem: Use of expensive resources.

Solution: Multiples flows can be directed to the same expensive resource and with slicing you can maintain the separation. Now you can have production and development traffic go through the same network hardware without fear of one affecting the other.

I have seen this happen in many networks that traffic is backhauled inside the datacenter for intrusion detection. Flows can do this much easily at the edge itself.

Network Access Control (NAC) can also be done easily by applying rules to flows. The first flow will go to the controller which can then decide using AAA or any other criteria if this host is:

- allowed access as a guest
- allowed full access
- no access allowed

The controller can then push a flow on to the network devices accordingly.

NAT will be just putting a rule that will change the IP address for traffic going out of certain interfaces and do the reverse when traffic comes back.

Ease of deep packet inspection coupled with detailed visibility on your unified wired and wireless network will easily automate how backup traffic, voice traffic, video traffic or traffic that needs encryption will be treated. SDN controllers can dynamically redirect flows for each of the above situations and make changes in case of congestion, packet drops or errors on a network

Traffic segregation that may be required by law or enterprise policies can be easily achieved by making sure that the flows of specific traffic types never mix.

Problem: Standardization of configurations across the enterprise.

Solution: Because of the centralized controller being at the heart of SDN, there will be standardization of configuration which is very hard to achieve and even harder to maintain.

Problem: High operations cost.

Solution: SDN is not only centralized but also vendor neutral. This means that the operations team does not have to learn multiple vendor platforms as it will have a common look and feel for different services like load balancers, firewalls, wan acceleration etc. All of this will result in a smaller team being able to manage much larger networks thus reducing costs further.

Centralized view and control of the network brings many other advantages. For example, if you want to modify your QoS policy, the change would be implemented centrally on the controller which would be an easy fix instead of modifying the configurations of hundreds of devices.

Similarly for doing maintenance you can take one piece of hardware or even a part of the datacenter out of service with ease because flows can be proactively redirected to alternative paths. Once the maintenance is complete the controller can normalize traffic with equal ease.

Adding or removing services like firewall, load balancing, encryption etc. will become easy too as you have adjust the flows to follow the path through the firewall or load balancer. Taking this further you can easily add or remove a chain of services like these with ease.

You might be thinking that the above merits are applicable not only to an enterprise environment but for any other environment and you would be right. It is the same features that will be used in different ways depending on the environment.

CHAPTER 11 - SDN FOR SERVICE PROVIDERS

The demand for bandwidth is increasing for service providers. While the CAPEX (capital expenditure) is declining for wired services, it is increasing for wireless services with the overall wholesale margin also shrinking 9. Mobile data has already surpassed voice data traffic. Mobile data doubled between first quarter of 2012 and first quarter of 2013. It is set to double again this year 10. While this growth is predicted to go on at this pace for the next 5 years the growth for voice traffic which has been the so called bread and butter of Service providers is growing only by 4 percent. As a result the demands for giving new services are increasing while the revenue is not.

Note: *By service providers I mean companies like AT&T, Verizon, Deutsche Telekom etc.*

Another big attraction for service providers is the reduction of OPEX (operational expenditures) because of centralized control which will not be platform dependent. This centralized control gives service providers the ability to automate highly orchestrated environments at an unprecedented scale. They will not have to have vendor specific experts that take care of one part of the network.

Besides reducing the CAPEX and OPEX, SDN enables them for more efficient utilization of their existing bandwidth capacity. Having an overall view along with programming ability of the network gives the service providers the ability to divert traffic from saturated links to underutilized links.

The traditional ways of planning, building and deploying infrastructure is proving to be slow for the market. It is also very expensive and is not able to change according to the market needs. Service providers realize that they will need to launch new services much faster to differentiate themselves from the others. Those who cannot will fail. They need to adjust their networks according to the customer's application needs.

By having a central control of the network, which would not be vendor specific, would allow the service providers to freely innovate to provide its users a big variety of services that can generate revenue for them. Service providers will be able to increase their average revenue per user (ARPU) and may eventually charge for data services like you pay for electricity. You pay for what you use. This will be possible using SDN.

Introduction of added services like security or load balancing will just be a matter of redirecting of flows through that application specific device. For example, if you are having product campaign and expect 100 times more traffic for the initial two weeks then you can easily spin up the virtual infrastructure of virtual servers along with network capacity, load balancing, firewalls and then slowly bring it down as the demand stabilizes to normal level. Can you imagine doing this in the traditional way of procuring, configuring and putting extra switches, routers, load balancers and firewalls in a datacenter and then tearing it down after a the campaign? Usually you will be spending about six months of effort for that 2 week launch. Now it can be done much faster. Faster also means much lower costs and more flexibility.

SDN also gives unprecedented insight into network traffic that has not been available before. This insight is not only the knowledge of all layers of the OSI model for network traffic, but this along with time of day, geography, amount of traffic enables service providers to further provide more customized and specific services their customers based on

- Cost
- Load
- Latency
- Security
- Time of day
- Customers traffic policies
- Bandwidth

Each of these factors above provides value to the customer and this means additional revenue for the service providers.

For example, a customer can use a self-service portal from the service provider to choose low latency service for their voice and telepresence calls, additional internet bandwidth during chosen times of the day and more bandwidth on the lowest cost link for its backups. This ability to pick and choose has only been seen on the server side.

If you add this flexibility to a cloud service provider, then the possibilities are endless. Imagine that one day you wake up to see that your whole network, servers etc. are unusable. It could be due to natural disasters, virus/worm attacks, hackers etc. The good news is that you can also imagine that all you need to do is restore your data to cloud service providers who can spin up all your essential services in a matter of hours. You can bring up your essential services like DNS, DHCP, Active Directory, Email, Web services and much more to get you up and running.

Multi-tenancy for service providers will go to the next level as they will be able to provide customized services to each of their tenants while keeping absolute insulation between them.

CHAPTER 12 - SDN FOR WAN

Wide area networks have been the expensive part of the network. The cost of these links is coming down but not as fast the utilization is increasing. One of the biggest cost saving can be achieved by using these link more efficiently.

Big global companies have multiple links between continents. If this connectivity is between 10 cities in US to 5 cities in Europe and 5 Cities in UK, then it will be impossible to load balance traffic across all these links. I say this because if each of these links will have a backup link, which is on two separate routers, so it will involve 40 routers (20 sites x 2 routers each)

Each of these routers will use their best path algorithm to send traffic over the best possible path. However, if one of the links has errors the router cannot look at the other routers and send traffic over an alternate link.

SDN will have a controller that will have the overall vision of the network and it can redirect traffic from one link to the other. Therefore, if your CIO is having his Monday morning video conference with his counterparts in London and Frankfurt, you can tell the controller to give video traffic from New York, Frankfurt and London high priority. The controller can redirect Video traffic to the best reliable link while email and employees watching YouTube (of course they are watching something that will help them in their job) can still keep going on the other links. Email traffic can always bear the packet retransmissions but video cannot.

In a similar fashion, the controller can redirect and distribute traffic in a way that all links are utilized and there is least amount of idle capacity.

The best example I have found is Google. They made their own app and are now utilizing their wan links at 95%.

CHAPTER 13 - SDN FOR DATACENTER

Datacenter is the piece of the network that has already seen virtualization. Computing power (CPU and memory), storage (SAN) have already been decoupled from hardware. The next logical piece to be virtualized is the network.

I used to watch the TV Series "Mad Men" which is about advertising during the 60s in US. In a few episodes they show the phone room where in the operators are connecting calls physically by plugging and unplugging cables. This is how it used to be. However, was that model scalable? Can you imagine someone plugging a connection for you whenever you make a call? It may not be humanly possible but that is exactly what happens when you make a call today. A virtual connection is made automatically for you. Since this is being done by software, on demand by the application (you making that call) it is scalable and fast. Now, take that analogy to a datacenter. Each datacenter has patch panels that connects switches, servers and many other devices to the each other. Whenever we want to connect one server to another server we create a VLAN for that. However there is a limit to the number of VLANs. It is a manual process too. SDN promises us that we can have those connections made by inserting flows on demand. SDN not only plumbs the connection for you from one end of the data center to the other but also un-plumbs that once the connection is not being used. Through software this can be done much faster and this will result in mega datacenters with many thousand VM's connecting to each other.

This automation of provisioning the network along with the automation on provisioning virtual machines will surely revolutionize IT. This will result in a flexible Datacenter that can help complete backups when the demands are less and give priority to application traffic when there is high demand.

It is common to hear that X website crashed due to too much demand. This has happened to leading news website, movie streaming website, and tax fling website just to name a few. Can you imagine a datacenter that will respond by claiming network, compute and storage resources to the increased traffic for an event (A movie premier, a special product sale event, the World Cup final, a breaking news story) and then return those resources back to the pool when not needed?

This automation of provisioning and management will be a prime factor in getting SDN inside the datacenter.

If you still have your doubts, you will feel more confident in SDN when I tell you that Google is already using this. Google's internal traffic between datacenters is growing much faster than the user facing traffic. This growth is too expensive because networks don't get cheaper with scale. By using SDN Google could buy generic hardware based on its requirements and make changes to it using software which is much faster. Any change in hardware design for new features usually takes a long time which could be a year or more. However software can be updated regularly and more changes and flexibility can be achieved by Google using SDN. Using an SDN controller give Google a centralized view of the network which makes it much easier to manage.

Chapter 14 - What does SDN mean for Companies like Cisco and Juniper?

The SDN market is in its nascent stages right now. Almost any vendor you can think of has a SDN strategy. Some have a very aggressive approach to it while some have a defensive approach.

If you look at Open network Foundation that is the non-profit organization, taking care of the OpenFlow standard it has 98 members (8) at the time of writing this book.

These members include

- Hardware device vendors like
  - Cisco, Juniper, Arista, HP, F5, Brocade, Extreme etc.
- Network users like
  - Google, Facebook, Yahoo etc.
- Hardware vendors like
  - Ciena, Spirent , Texas Instruments, Intel etc.
- Software vendors and many more.

Are all vendors using an external controller and OpenFlow?

With standards being developed by the Open Network Foundation, many companies like HP, IBM, Big Switch, Dell and NEC have created implementations that use an OpenFlow controller, while there are others like VMware, who are not using an external controller at all. VMware is using software it acquired from Nicira to be used in its vSwitch without the use of a controller. Cisco's SDN architecture which is called Cisco ONE (open network environment) is trying to embed the controller intelligence in its hardware and network management software.

Cisco acquired Cariden for $141 million. Interestingly Cariden, is a software company which drives the adoption of software that can ease the transition to SDN. Cisco supports the OpenFlow controller but also has onePK (Cisco ONE platform kit). While OpenFlow has limited functionality, onePK can expose a very high number of API's that can use the intelligence built into Cisco hardware ASICS (Application Specific Integrated Chips). This strategy not only supports OpenFlow but also adds a little proprietary spin to it. This, in turn, keeps the customer buying Cisco hardware which the company would like to save as it has enormous investment in that. There is always a possibility that Cisco can develop more protocols which will work with or parallel to OpenFlow.

Cariden software is a part of the Floodlight controller which you can download and play with. See **appendix B**.

Cisco is also funding a separate company by the name of Insieme which is developing programmable hardware and software. You may find it amusing that Insieme is being run by the same team of three Cisco engineers who gave Cisco the SAN switches and then the Nexus switches.

It is natural for Cisco to save its hardware business, but by embracing SDN, it is playing in this market. This will make sure that it can ride the wave rather than miss the boat.

Juniper also acquired Contrail Systems for $176 million which makes controller software for SDN. Interestingly, the CTO of Contrail Systems was the chief architect of JUNOS the operating system for Juniper hardware.

Juniper has more of a software spin on SDN and even announced a licensing business model. However, Juniper like Cisco, is also emphasizing its own hardware ASICS. Like Cisco, Juniper also sees layer 4 to layer 7 services as virtualized application running in SDN infrastructure.

Both of them are have controllers but are promoting a hybrid control plane which is distributed.

Cisco has will have SDN compatible products out in 2013 while Juniper has promised this in 2014.

Juniper is initially going after the service provider edge and the datacenter market and plans to expand to its traditional market of service providers from there.

Cisco is attacking the enterprise, service provider and datacenter market all at once by developing API’s that will make IOS, IOS-XR and NX-OS programmable using the Cisco ONE platform.

Both these companies see SDN as a disruptive technology but are having a very different approach to it.

Although there are many other players in the SDN market that are shipping hardware today but I have not mentioned their names. This is intentional because the theme for this chapter is to see how the big two of networking are reacting to SDN.

Chapter 15 – Hype or Reality

After all the pages I have written about SDN: Is it just hype, or will it turn into a reality? Will it remain in the realm of academia and experimental networks or will the service providers and enterprise customers adapt this? The answer to this lies in the future.

I personally feel that SDN is here at the right time. Data center virtualization is maturing and organizations are adapting it globally. Inside a datacenter, the compute and storage instances have been decoupled from hardware--- the only piece left is the network. **An elastic pool of compute, storage and network resources that can give you an on demand datacenter which can be scaled by adding more resources does not sound like a hyperbole.** Virtualization has gained the trust and confidence at every level of the IT organization. With this in the background network virtualization should be easier to adopt.

IDC the leading provider of market intelligence, announced in May 2012 that SDN market would grow to $2 billion (6) by 2016. They revised this number to $3.7 billion (7) by December 2012. SDN is in Gartner's (5) top ten critical tech trends for the next 5 years. This can lead to a significant disruption in organizational networks.

By now, I hope that you have enough understanding of SDN to start asking the right questions. Like

- Do you need SDN/OpenFlow?
- Are present day network devices ready for this new technology?
- How are the current implementations of SDN working?
- Will SDN scale?

In my view, SDN is like any other Client server application (controller being the server) and this will have the same scalability issue as that of any such application. The good thing is that we have experience in scaling that architecture.

The other issue I see is convergence. Although we can have backup flows or multiple flows installed in the flow tables for a faster convergence but if the switch down message from the switch to the controller has to be less than the most time sensitive application on the network like Voice.

There is also the issue of having a limited amount of TCAM in present day switches. As a result, there's a limited number of flows you can install in the TCAM. The next generation chipsets will address this for sure but for now we do have this problem and the current hardware is not going to be phased out for the next seven to ten years. This could mean that SDN could be implemented first on the edge devices, which usually talk to the core/distribution and not to other edge devices. The number of flows and the change in flows per device are low as compared to all the flows that have to be handled by the core devices. You can argue to have granular flows at the edge while having coarse flows at the core in these kinds of situations and you would not be wrong. It all depends on the size and requirements of your network.

If you look into the specifications of current hardware that support OpenFlow you will see that many features are supported only in switch software and not hardware which you know is much slower.

All these big companies that are selling dedicated hardware for application like Firewall, Load balancing, VPN etc. will be reduced to just software companies as all these will be just an application that can talk to the controller to control the flows.

Security is another concern for experts for implementing SDN.

As I said in the first few sentences in this book, this debate has been going on for the past two years and will not stop for a few more. However, I believe that before iPhone there was hardly a market for apps but now you can get an app for anything you can think of and that too for free or very low price. Open API’s means that there will be many applications available before you know it.

There will be a big choice of controllers as well as protocol alternatives to OpenFlow (see appendix A) and there will be proprietary implementations until the market settles.

How does all of this play out is to be seen? The way I see it, it is history in the making and I am glad that you are also a part of it now.

CHAPTER 16 – THE FUTURE

One question that comes to everyone’s mind is, “How will the networks look like when the transition to SDN is done?”

This question is for the top SDN experts to answer, but I can tell you what I see through my looking glass. My view may be proven wrong but that would only mean something better has happened.

Without going into specifics, I think the next**GEN** (Great Earth Network) will be something like the model that we follow for air traffic nowadays.

Think of the plane as the packet and the passengers sitting inside the application data. The flight path is the data plane and the air traffic controller is the SDN controller.

As the flight tells the controller of its destination, the air traffic controller decides the time of departure, the runway, the flight path, its cruising altitude etc. Once the aircraft is airborne, it is tracked by one or more air traffic controllers at different airports along the way. The aircraft keeps checking in and out with different controllers in its flight path until it reaches its destination. Once it lands at the destination it delivers its payload of passengers, luggage, etc.

If the aircraft needs special attention or is in distress (engine problems, passenger problems) it can ask for special handling from the air traffic controller. On the other hand if there is an issue on the data path (storms, volcano eruptions) then the controller can divert the aircraft to an alternative path and make sure it reaches the destination safely. There will be times when the aircraft will need to land at an alternative location and continue the flight afterwards.

Aircraft = Data packet

Passengers = Application data

Runway number = Switch port number

Flight path = Data path

Air traffic controller = SDN controller

In SDN, the application talks to the controller and tells it the needs for that data packet like QoS, low latency (voice packet), or don’t care about latency and hops (backup data packet) etc. This is done using the northbound API’s. The controller then decides the data path for that packet and the sends it out the desired port. This packet follows the path assigned till the destination is reached (using south bound API’s). In case the data path is too long or the data packet is destined for another AS (Autonomous System), then the packet will be handed over the next controller at the edge along with the special handling instructions. This data packet can be diverted to alternative paths in case of congestion or failures and can be diverted to alternative ports in case of port failures.

This model for air traffic has worked for us and I think will work in the SDN world also.

You must be thinking that if SDN is so great then why is not there already? Enterprises and Service providers don’t want to implement cool technology. Networks are there to help and facilitate in their primary reason for existence which could be making cars, providing health care, handling your money or flying space missions. SDN has proven itself as a great concept. As of now there is no application marketplace per se for SDN which is selling APP’s that will facilitate these companies in their specific needs. All the big vendors are coming up with offerings that will help these customers leverage their existing infrastructure investment while moving ahead towards SDN at a pace that they choose. As time goes by we will start seeing and hearing more about this.

Part II: Hands on Labs on SDN using OpenFlow

Chapter 17 – The Prerequisites

It takes a collection of software to run a SDN lab on your PC/Laptop. So give yourself some time and patience to do this. Since I am doing this on a MS Windows 7/Windows 8 platforms all my references and screenshots will be based on that. Please download the XTerm emulator, telnet/SSH client and the virtual machines according to your platform of choice.

As we know by now that SDN uses a software based controller, so we will be emulating that in a virtual machine (VM) in our lab. We will be using the "**OpenDaylight**" controller from Open Daylight Project. We will also be downloading a virtual switch (mininet) from mininet.org that helps in simulating a network on your computer.

If you have access to OpenFlow compatible hardware then you can use that instead of the VM. You can also make your own OpenFlow compatible switch. **See appendix F** for details.

We will be using VMware player from VMware to run our OpenFlow controller VM and OpenFlow switch VM. This can be done equally well on Virtualbox. Although I have used both, VMware platform was already installed on my Laptop so I preferred this over others.

In short VMware player will be running at least two VM's, one for the controller and one for the switches to simulate your network playground.

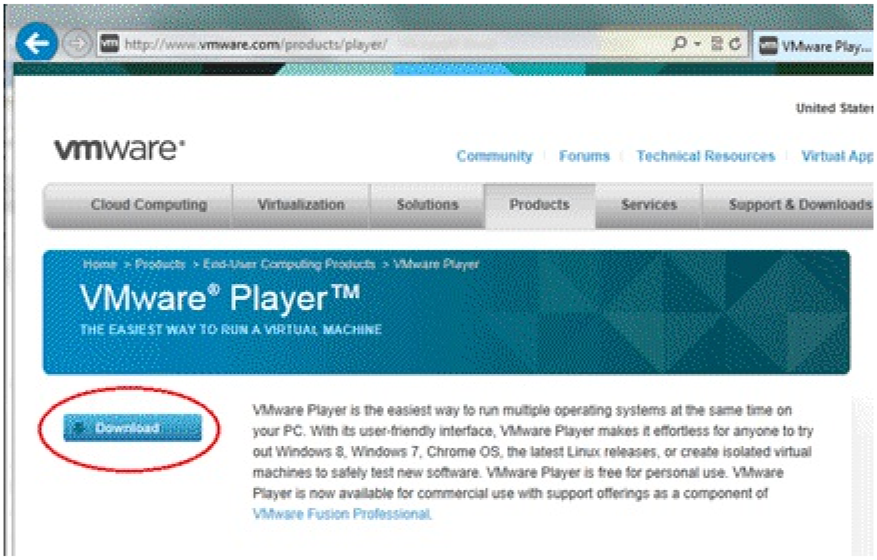
VMware player is free and is available for many other hardware platforms and operating systems besides windows.

I am running these labs on a core i5 2nd generation CPU with 6 Gigs of memory.

You will also need about 25 gigs of hard disk space.

Download VMware player

You can download VMware player from vmware.com/products/player/ (Figure 17.1 and 17.2 below). The player is free for personal use but VMware **used to** ask you to create an account on their website which was a very short process when I did that. Lately I have been told that you do not need to register with VMware to download it. Please check yourself to verify.



VMware Player is the easiest way to run multiple operating systems at the same time on your PC. With its user-friendly interface, VMware Player makes it effortless for anyone to try out Windows 8, Windows 7, Chrome OS, the latest Linux releases, or create isolated virtual machines to safely test new software. VMware Player is free for personal use. VMware Player is now available for commercial use with support offerings as a component of VMware Fusion Professional.

Figure 17.1

Even if you have to create an account it is worth the trouble. The size of this download is 76MB for version 5.0.2.

***Note**: As of time of writing this book, there was no registration requirement.*

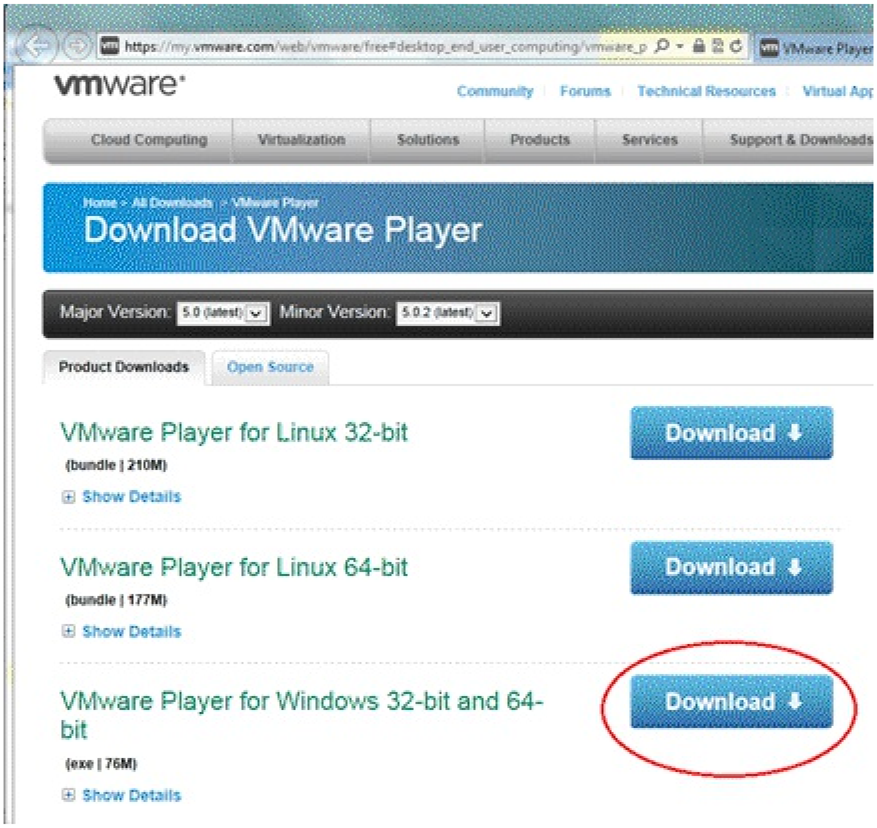


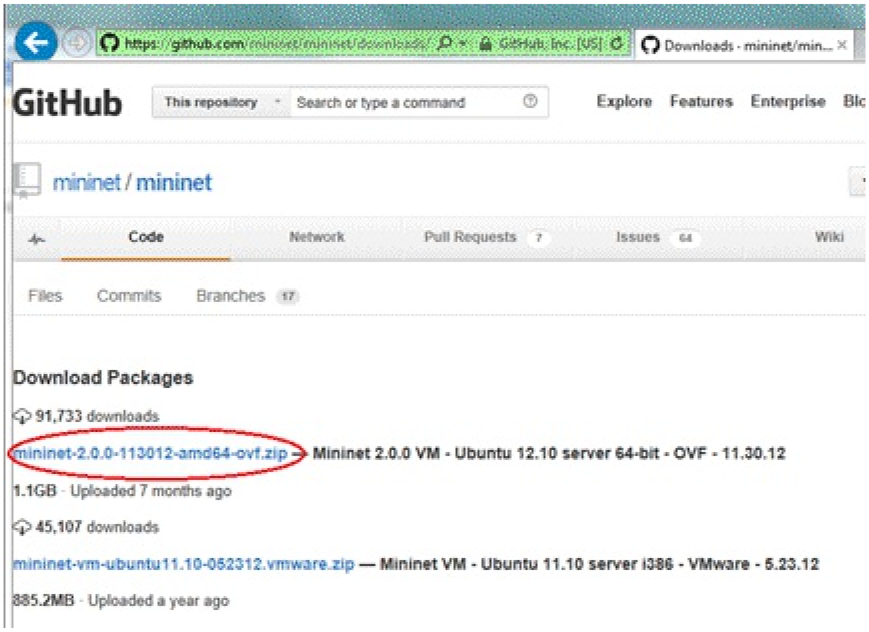
Figure 17.2

It is expected that you have enough familiarity with your operating system and basic networking that you will be able to download and install the player.

***Note:** Once the player is installed and you start running the virtual machines for the OpenFlow* *controller and virtual switch you may be asked to install VMware tools. As a precaution make a copy of the stable running VM before you install VMware tools.*

Download Mininet virtual switch - virtual machine

Download the virtual switch software from **mininet.org**. As of writing this book the latest version for VMware is "mininet-vm-ubuntu11.10-052312.vmware.zip" which is about a year old. There is a newer image "**mininet-2.0.0-113012-amd64-ovf.zip**" which is a 64 bit image and is a bit newer. Size is 1.1 GB.



Download Packages

91,733 downloads

mininet-2.0.0-113012-amd64-ovf.zip — Mininet 2.0.0 VM - Ubuntu 12.10 server 64-bit - OVF - 11.30.12

1.1GB · Uploaded 7 months ago

45,107 downloads

mininet-vm-ubuntu11.10-052312.vmware.zip — Mininet VM - Ubuntu 11.10 server i386 - VMware - 5.23.12

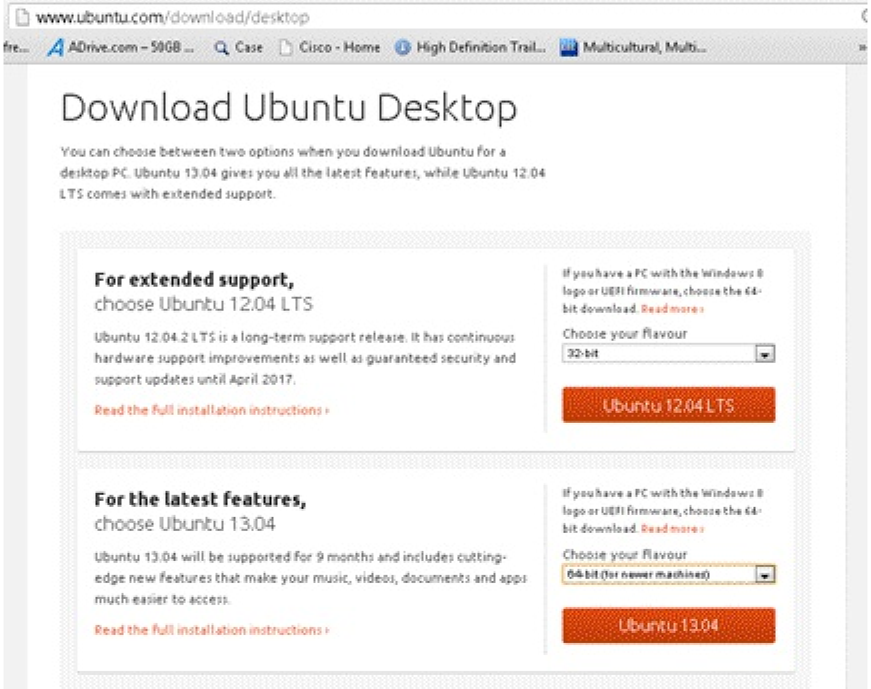
885.2MB · Uploaded a year ago

Figure 17.3

This can be easily imported into VMware. We will be using this newer image for our lab.

Download and install Ubuntu desktop

Next you need to download the latest desktop image from Ubuntu. I downloaded v13.04, 64-bit version. Choose the version that is right for you hardware. If you are installing a virtualization platform for the first time on your computer you may have to go to the bios and **enable hardware virtualization**. A quick web search for **"**enabling virtualization in bios**"** gets you good help. You can also get instructions to install Ubuntu on windows 7 using VMware by searching for "run Ubuntu in windows 7 with vmware player". Make sure you have a stable Ubuntu VM running before you proceed with installing the OpenDaylight controller.



Download Ubuntu Desktop

You can choose between two options when you download Ubuntu for a desktop PC. Ubuntu 13.04 gives you all the latest features, while Ubuntu 12.04 LTS comes with extended support.

For extended support,
choose Ubuntu 12.04 LTS

Ubuntu 12.04.2 LTS is a long-term support release. It has continuous hardware support improvements as well as guaranteed security and support updates until April 2017.

Read the full installation instructions ›

If you have a PC with the Windows 8 logo or UEFI firmware, choose the 64-bit download. Read more ›

Choose your flavour: 32-bit

For the latest features,
choose Ubuntu 13.04

Ubuntu 13.04 will be supported for 9 months and includes cutting-edge new features that make your music, videos, documents and apps much easier to access.

Read the full installation instructions ›

If you have a PC with the Windows 8 logo or UEFI firmware, choose the 64-bit download. Read more ›

Choose your flavour: 64-bit (for newer machines)

Figure 17.4

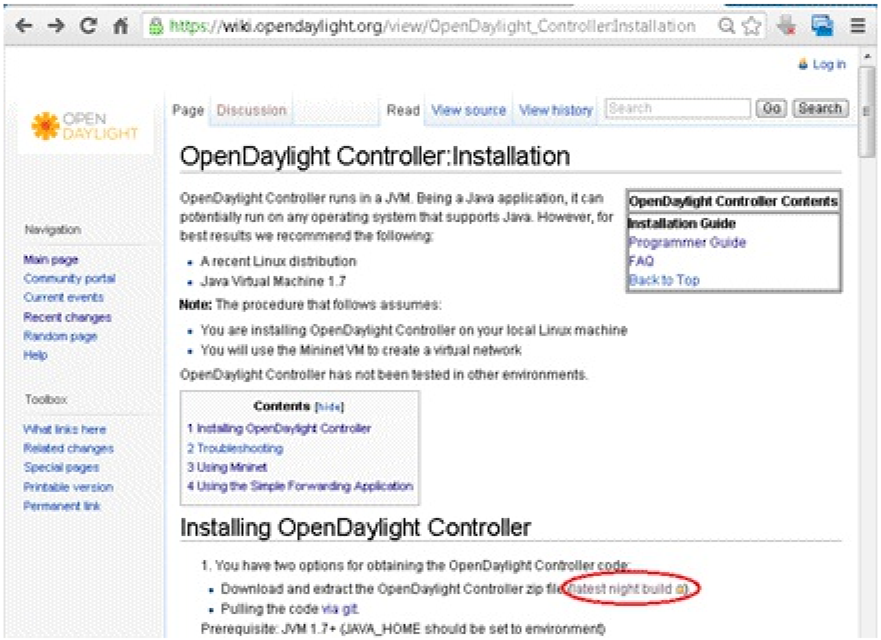
Downloading OpenDaylight controller

This is the controller that is being developed with the backing of big companies like Cisco, Microsoft, Juniper, Redhat, IBM, Brocade, Ericsson and many more.

There are two ways you can make this work for you.

1. Download and compile the latest code by yourself on your VM. Although this is bit more of a process but I did this myself and have listed that process in **Appendix C**.
2. Download a readymade snapshot of this software.

I can just list the URL for downloading the package but it is very long and typing that out from this book can be frustrating. So here is an easy way to reach that URL. Search for "opendaylight controller installation" and choose the URL (https://wiki.opendaylight.org/view/OpenDaylight_Controller:Installation) and you will reach the webpage shown in the **Figure 17.5** and **17.6** below.



OpenDaylight Controller:Installation

OpenDaylight Controller runs in a JVM. Being a Java application, it can potentially run on any operating system that supports Java. However, for best results we recommend the following:

- A recent Linux distribution
- Java Virtual Machine 1.7

Note: The procedure that follows assumes:

- You are installing OpenDaylight Controller on your local Linux machine
- You will use the Mininet VM to create a virtual network

OpenDaylight Controller has not been tested in other environments.

OpenDaylight Controller Contents

Installation Guide
Programmer Guide
FAQ
Back to Top

Contents [hide]

1 Installing OpenDaylight Controller
2 Troubleshooting
3 Using Mininet
4 Using the Simple Forwarding Application

Installing OpenDaylight Controller

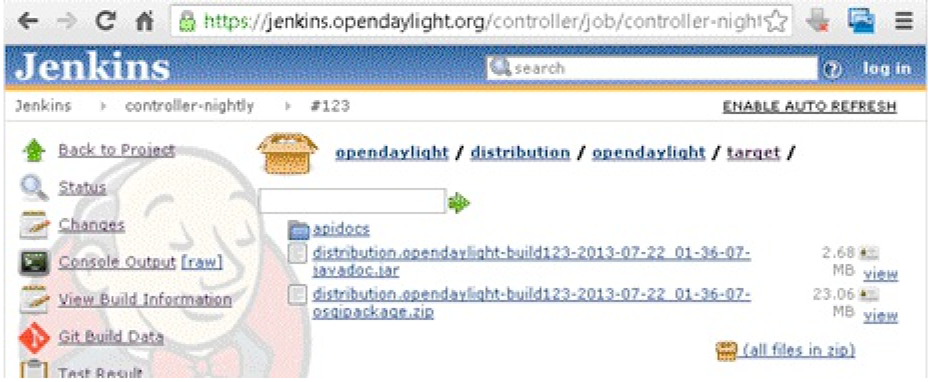
1. You have two options for obtaining the OpenDaylight Controller code:
   - Download and extract the OpenDaylight Controller zip file (latest night build)
   - Pulling the code via git

Prerequisite: JVM 1.7+ (JAVA_HOME should be set to environment)

Figure 17.5

Click on the latest night build link and you will be redirected to this URL

https://jenkins.opendaylight.org/controller/job/controller-nightly/lastSuccessfulBuild/artifact/opendaylight/distribution/opendaylight/target/



opendaylight / distribution / opendaylight / target /

apidocs		
distribution.opendaylight-build123-2013-07-22_01-36-07-javadoc.jar	2.68 MB	view
distribution.opendaylight-build123-2013-07-22_01-36-07-osgipackage.zip	23.06 MB	view

(all files in zip)

Figure 17.6

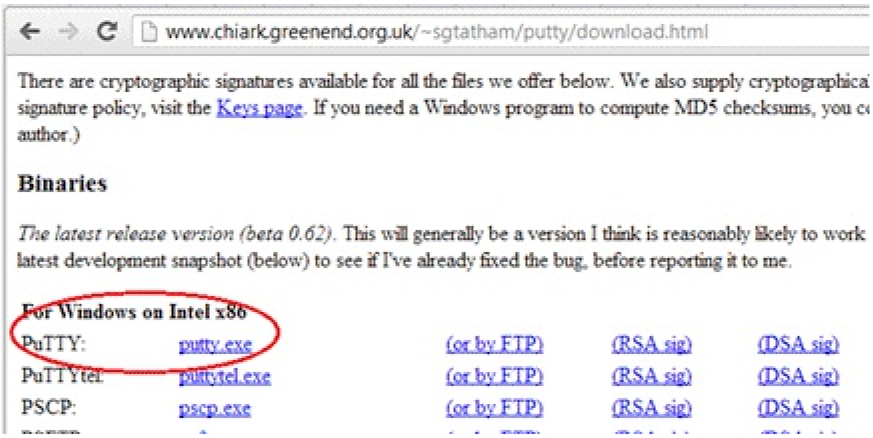
Download the zip file and you are done.

Download Putty

Just in case you do not have Putty or any other telnet/SSH client, please download this from

www.chiark.greenend.org.uk/~sgtatham/putty/download.html

I downloaded the “putty.exe” file only as shown in figure 17.7 below.



There are cryptographic signatures available for all the files we offer below. We also supply cryptographica signature policy, visit the Keys page. If you need a Windows program to compute MD5 checksums, you c author.)

Binaries

The latest release version (beta 0.62). This will generally be a version I think is reasonably likely to work latest development snapshot (below) to see if I've already fixed the bug, before reporting it to me.

For Windows on Intel x86

PuTTY:	putty.exe	(or by FTP)	(RSA sig)	(DSA sig)
PuTTYtel:	puttytel.exe	(or by FTP)	(RSA sig)	(DSA sig)
PSCP:	pscp.exe	(or by FTP)	(RSA sig)	(DSA sig)

Figure 17.7

We will be using Putty as it works great with XMING our Xterm emulator.

Download XMING

Lastly we will be using XMING, a free Xterm emulator for windows which is very compatible with putty. This can be downloaded from sourceforge.net/projects/xming/

Please do check each of these sites for their terms and conditions of use.

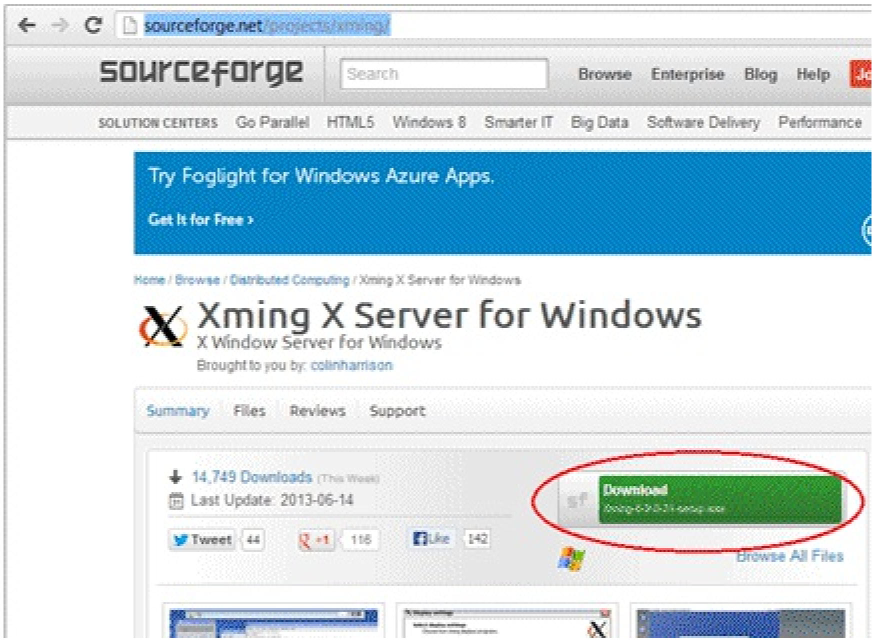

sourceforge.net/projects/xming/
sourceforge
Search
Browse
Enterprise
Blog
Help
SOLUTION CENTERS
Go Parallel
HTML5
Windows 8
Smarter IT
Big Data
Software Delivery
Performance
Try Foglight for Windows Azure Apps.
Get It for Free ›
Home / Browse / Distributed Computing / Xming X Server for Windows
Xming X Server for Windows
X Window Server for Windows
Brought to you by: colinharrison
Summary
Files
Reviews
Support
14,749 Downloads (This Week)
Last Update: 2013-06-14
Tweet
44
+1
116
Like
142
Download
Browse All Files

Figure 17.8

Phew!!! That was a lot of downloading, but it is worth the effort.

Here is a short checklist for your reference
Hardware CPU and Memory: I have successfully made this work on Intel i5 2ndGen 6 Gig and i7 4thGen 12 Gig laptops.
VMware player from “www.vmware.com/products/player/
mininet VM (mininet-2.0.0-113012-amd64-ovf.zip) switch from mininet.org
Ubuntu desktop from http://www.ubuntu.com/download/desktop
OpenDaylight Controller from https://jenkins.opendaylight.org/controller/job/controller-nightly/lastSuccessfulBuild/artifact/opendaylight/distribution/opendaylight/target/
telnet/SSH/Xterm emulator Putty from chiark.greenend.org.uk/~sgtatham/putty/download.html
Download XMING XSERVER for MS Windows from sourceforge.net/projects/xming/

CHAPTER 18 – SETTING UP THE PLAYGROUND

There are multiple ways that we can setup our playground for SDN hands on labs.

Option 1: One computer running two virtual machines for OpenDaylight and Mininet. You will need a beefier computer (8 Gigs of Memory and a core i5 or i7) for this. You can run the Mininet VM with 512M to 1024 M of Memory but the OpenDaylight needs between 1024M to 2048M. You can run these machines with a lower memory allocation but it all depends what you want to do. I am just telling you the settings that I used.

> ***Note:****My setup is all windows based and I am sure you can use a machine with lesser specs using Ubuntu***.**

Option 2: Use two computers. One is running the OpenDaylight controller in Windows (or on a virtual machine) and the other running the Mininet virtual machine. The computer running the Mininet virtual machine has minimal requirements while the computer running OpenDaylight will be needing 4 Gig of Memory and a core i3 or i5 processor.

Option 3: Have Mininet and OpenDaylight running on a single VM running on one computer. This computer should be like the one in option 1 (8 Gigs of Memory and a core i5 or i7). I tried this and it works great.The only issue I saw was that in the Wireshark traces you will see packets going from 127.0.0.1 to 127.0.0.1. This is because you are running packets inside the VM which is using the loopback interface to talk to the Mininet and OpenDaylight. That can become confusing at times. It will run great functionality wise.

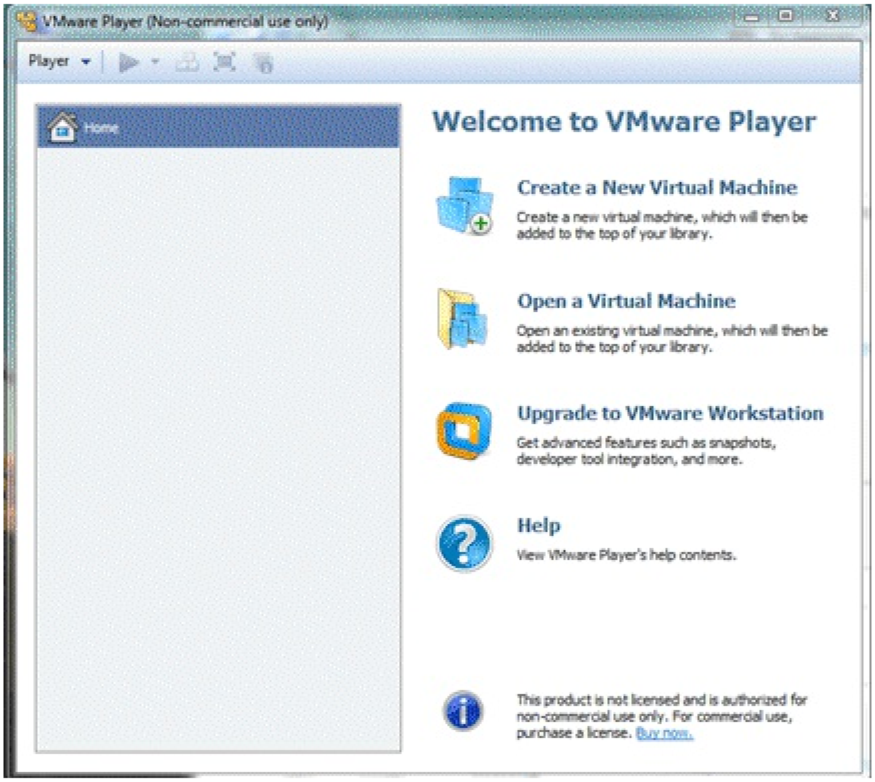
Option 4: I am mentioning this here because I tried this and was even able to make it work on one computer but could not replicate the steps to make it work again on another. I installed OpenDaylight on windows7 and ran the Mininet VM. This is great but making the Mininet VM talk to the host is tricky. Windows7 and Windows8 inherently put the VMware interfaces on the public / unidentified networks and you have to do a lot of tweaks to make VM talk to the host on all the tcp ports. You can ping but cannot telnet or ssh. I was able to make this work on one of my laptops so I know it is possible. However, I could not find a reliable and repeatable step by step way to do this. Maybe you, the reader, can make this work and can send me the methodology.

I used option 1 (one computer running two separate VM's) for the labs.

Option 5: You have the option of using a real physical hardware switch if you have one that is OpenFlow compatible. Since there are not many engineers who have access to this kind of hardware today, I will use the Virtual Mininet switch only. However these labs should be possible with physical hardware also.

Setting up VMware Player

Installing your VMware player should be straight forward. You may need to run the install with administrator privileges. Alternatively you can right click on the executable file and choose to run as administrator. After you are done installing you will see this VMware Player on your desktop and starting VMware player should show you the following screen.



Welcome to VMware Player

Create a New Virtual Machine
Create a new virtual machine, which will then be added to the top of your library.

Open a Virtual Machine
Open an existing virtual machine, which will then be added to the top of your library.

Upgrade to VMware Workstation
Get advanced features such as snapshots, developer tool integration, and more.

Help
View VMware Player's help contents.

This product is not licensed and is authorized for non-commercial use only. For commercial use, purchase a license. Buy now.

Figure 18.1

Setting up Ubuntu VM

Setting up Ubuntu desktop to run as a VM is not very difficult. I am not putting screen shots for that as you can you can get instructions to install Ubuntu on windows 7 using VMware by searching for “run Ubuntu in windows 7 with vmware player”.

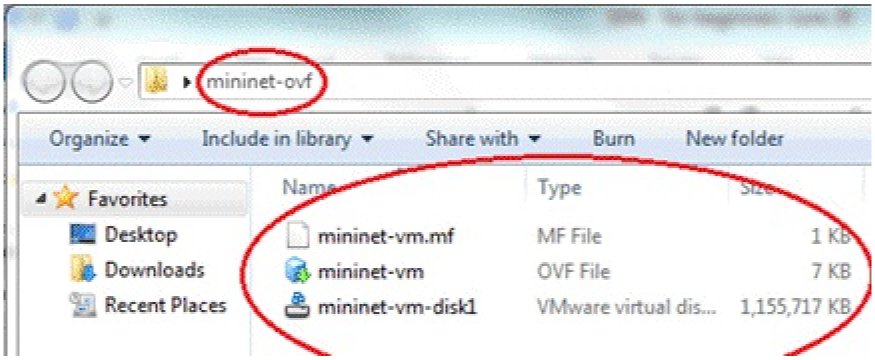
One thing I wanted to add here is the VM settings.

- To compile my own OpenDaylight code I had to use a VM with 8 gigs of memory.
- To run the compiled or downloaded code I used only 2 gigs of memory for the VM.

Setting up Mininet VM

Step 1

If your download for the Mininet VM was successful, you should have this **file "mininet-2.0.0-113012-amd64-ovf.zip"**. Unzip this file and you will have the following folder mininet-ovf with 3 files as seen below.

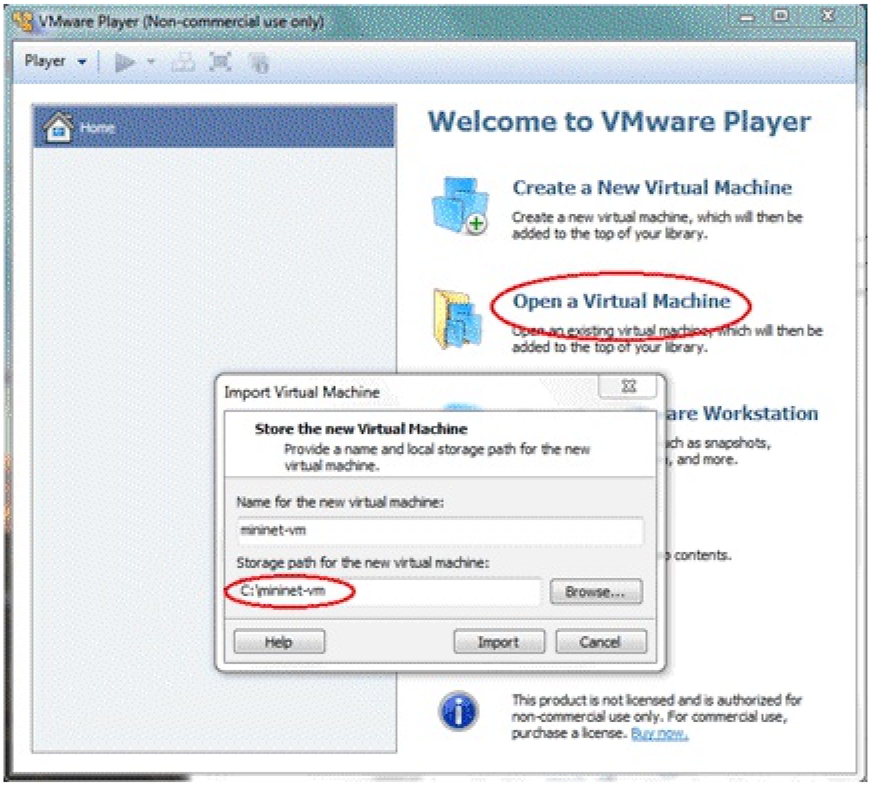


Name	Type	Size
mininet-vm.mf	MF File	1 KB
mininet-vm	OVF File	7 KB
mininet-vm-disk1	VMware virtual dis...	1,155,717 KB

Figure 18.2

Step 2

Now start your VMware Player click on open virtual machine. VMware player will ask for the path of the machine you want to open. Point it to the **folder mininet-ovf** which you made in step 1. Then click on mininet-vm. You will see a screen below.



Create a new virtual machine, which will then be added to the top of your library.

Open an existing virtual machine, which will then be added to the top of your library.

Store the new Virtual Machine

Provide a name and local storage path for the new virtual machine.

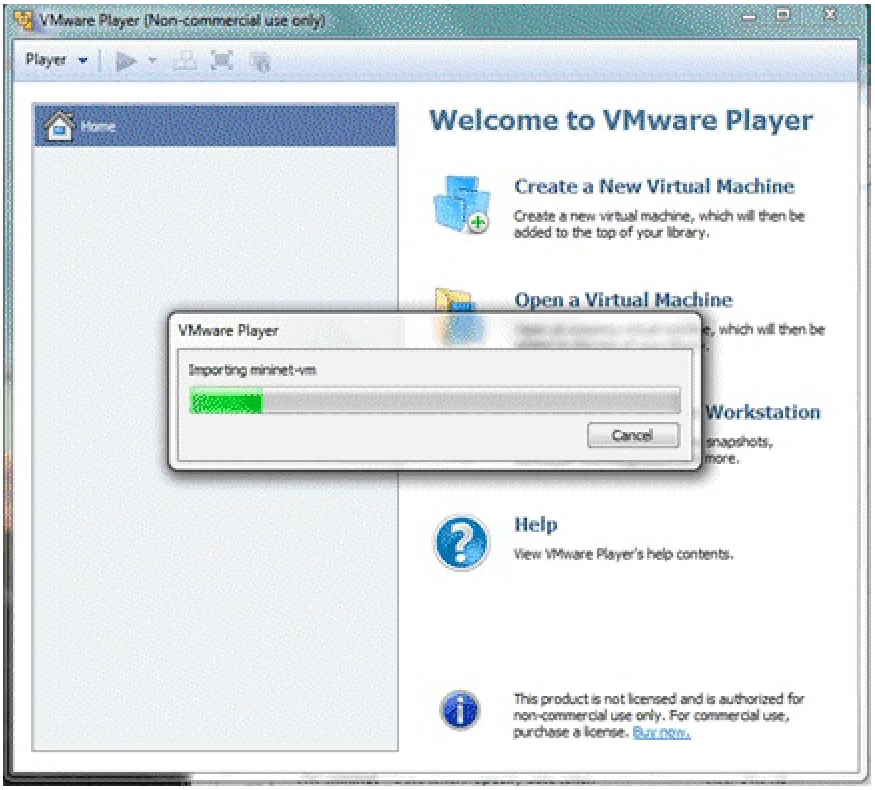
Name for the new virtual machine: mininet-vm

Storage path for the new virtual machine: C:\mininet-vm

This product is not licensed and is authorized for non-commercial use only. For commercial use, purchase a license. Buy now.

Figure 18.3

You can set the location where you want to import and store your mininet VM. I have used C:\ for simplicity. Please remember this location and click on import. This will start the import as seen below.



VMware Player (Non-commercial use only)

Welcome to VMware Player

Create a New Virtual Machine

Create a new virtual machine, which will then be added to the top of your library.

Open a Virtual Machine

, which will then be

Workstation

snapshots,
more.

Help

View VMware Player's help contents.

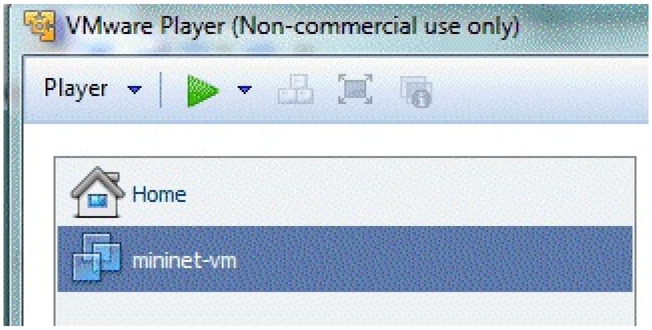
This product is not licensed and is authorized for non-commercial use only. For commercial use, purchase a license. Buy now.

VMware Player

Importing mininet-vm

Figure 18.4

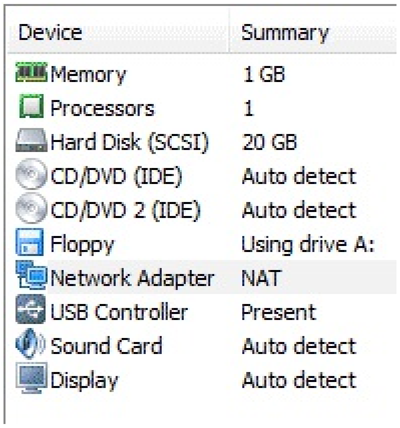
This import can take up to 5 minutes or more so be patient. Once this is complete you will have the mininet-vm in your VMware player.



VMware Player (Non-commercial use only)

Figure 18.5

Please note that this VM has been assigned 1 CPU, 1GB of memory and 20GB of disk space. Make sure that network adapter is set for NAT. See screenshot below.



Device	Summary
Memory	1 GB
Processors	1
Hard Disk (SCSI)	20 GB
CD/DVD (IDE)	Auto detect
CD/DVD 2 (IDE)	Auto detect
Floppy	Using drive A:
Network Adapter	NAT
USB Controller	Present
Sound Card	Auto detect
Display	Auto detect

Figure 18.6

These virtual machine settings are good for us, but I am showing these here so you can change these according to your needs and the power of your machine.

Setting up OpenDaylight

If you chose the option of downloading and building your own OpenDaylight package (I did this) then please follow the steps in Appendix C which has the procedure listed for Windows and Ubuntu.

For those who chose to download the prebuilt OpenDaylight package, you can set it up as follows:

OpenDaylight on Ubuntu

Open the zip file (distribution.opendaylight-buildzzz-2013-xx-yy_aa-bb-cc-osgipackage.zip) and you will see the "opendaylight" folder in it.

You will need to set Java home so that you can run Java from anywhere in the directory structure. For that you will use the following command.

export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-amd64

This will work but will not survive a reboot of the VM. To make this permanent you will need to edit your **.bashrc** file and place the line "export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-amd64 " preferably at the end. The **.bashrc** file is a hidden file in your home directory. If you are not very unix literate like me then please look this up on internet.

OpenDaylight on Windows

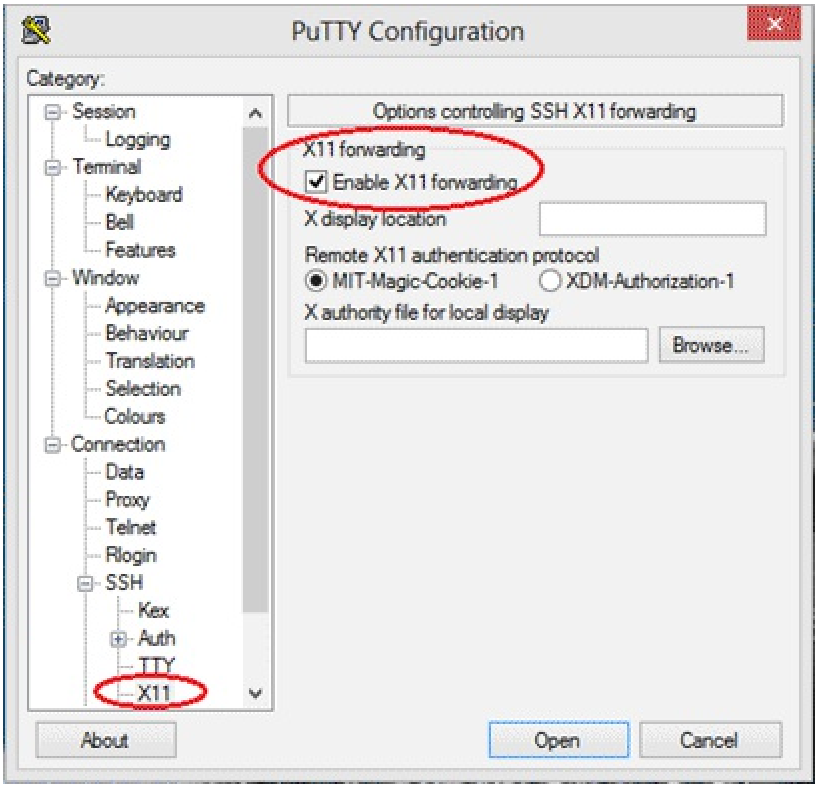
Open the zip file (distribution.opendaylight-buildzzz-2013-xx-yy_aa-bb-cc-osgipackage.zip) and you will see the "opendaylight" folder in it.

Extract that folder to your drive. I extracted the contents to c:\ because I installed mininet in there also. You can extract and copy the opendaylight folder anywhere you feel comfortable. The key is to remember that location so you can reach it and start the controller.

You may have to setup your environment for Java to make this work. Usually this is already there and you may not have to worry about it but I do have the instructions for it in the next chapter and **Appendix C**.

Setting up Putty

Since I downloaded the executable file "putty.exe" only there is no installation needed. However I did copy it on to my desktop so that I have easy access to it. We will be using putty with Xming so we need to make sure that X11 forwarding is enabled in putty. This is done by clicking on the checkbox as shown below. This option is available under the SSH section.



Options controlling SSH X11 forwarding

X11 forwarding

☑ Enable X11 forwarding

X display location

Remote X11 authentication protocol

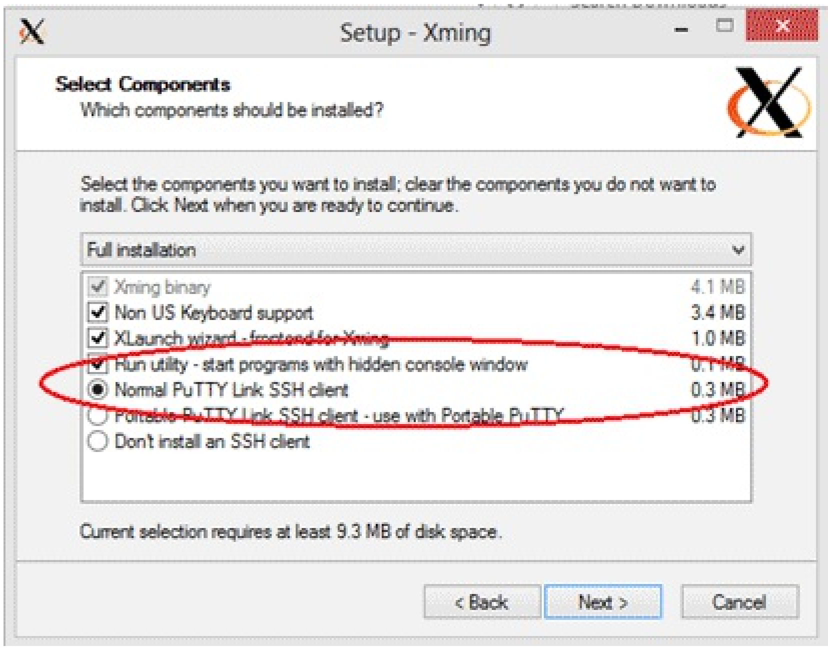
● MIT-Magic-Cookie-1 ○ XDM-Authorization-1

X authority file for local display

Figure 18.7

Setting up Xming

Xming installs like any other windows program. I chose the default option of Xming creating its hooks into putty as seen below.



Select Components

Which components should be installed?

Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

Full installation

☑ Xming binary	4.1 MB
☑ Non US Keyboard support	3.4 MB
☑ XLaunch wizard - frontend for Xming	1.0 MB
☑ Run utility - start programs with hidden console window	0.1 MB
● Normal PuTTY Link SSH client	0.3 MB
○ Portable PuTTY Link SSH client - use with Portable PuTTY	0.3 MB
○ Don't install an SSH client	

Current selection requires at least 9.3 MB of disk space.

Figure 18.8

To summarize we did the following:

1. Setup VMware player so that we can add the Mininet VM to it.
2. Create a new Ubuntu desktop VM.
3. Setup Mininet virtual machine and add it to VMware player.
4. Setup OpenDaylight controller for Ubuntu or Windows.
5. Setup putty.
6. Setup XMING.

Chapter 19 – Lets play

Starting the OpenDaylight controller

We will start the OpenDaylight controller first.

On Ubuntu

Open a terminal window and run the ifconfig command to find the IP address of your controller. You should note the IP address of eth0. Now navigate to the folder you extracted opendaylight package to,

cd /home/ opendaylight

then execute this command

sudo ./run.sh

Your OpenDaylight controller will be running in about a minute's time.

On Windows

Use windows explorer to navigate to the directory where you extracted the opendaylight folder. For me it was c:\opendaylight. Inside the opendaylight folder click on "run.bat". A command window will open which will run the code. Your OpenDaylight controller is now running.

You will also need to know the IP Address of your computer which can be done in multiple ways. I usually open a command prompt and run the ipconfig command.

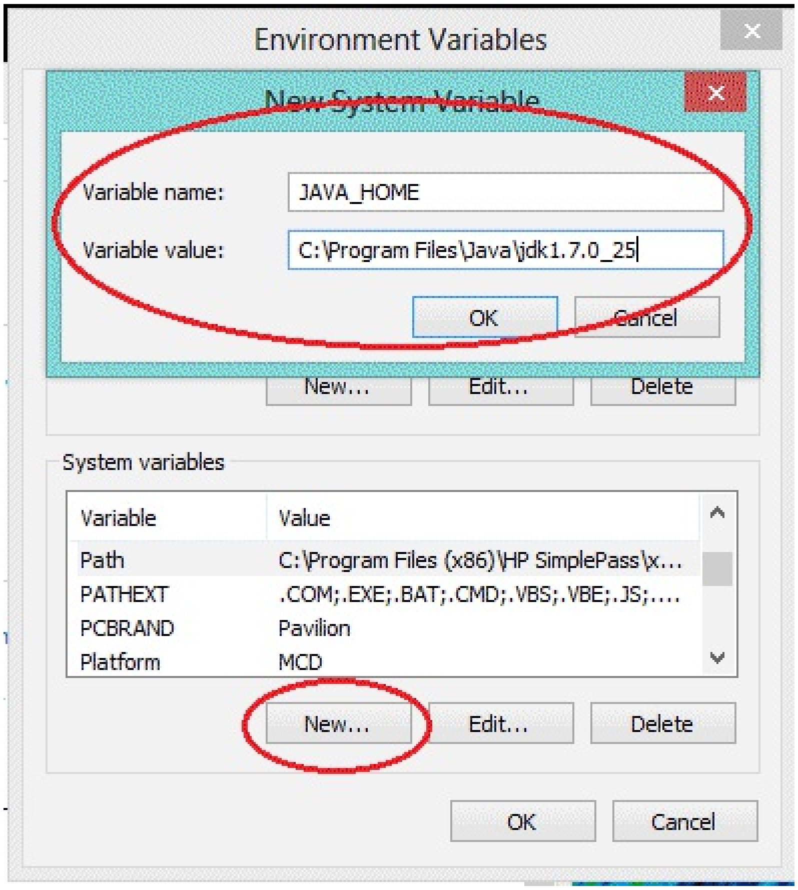
If you see an error related to java you will have to install java environment. See appendix C for details for downloading java. Once it is installed verify the path of the folder where it is installed. For me it was **C:\Program Files\Java\jdk1.7.0_25**. This path has to be added to system variables. You will need to follow steps 1 through 4 below.

Step 1: Right click on computer Icon Computer on your desktop and click on "properties". A new window will open.

Step 2: Now click on advanced system settings. Another window of "System Properties" will open.

Step 3: Now click on "Environment Variables" and that window will open.

Step 4: Click on "new" and add a new system variable of JAVA_HOME as seen in the screen shot below.



Variable name: JAVA_HOME

Variable value: C:\Program Files\Java\jdk1.7.0_25

Variable	Value
Path	C:\Program Files (x86)\HP SimplePass\x...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;....
PCBRAND	Pavilion
Platform	MCD

Figure 19.1

Verify using step 1 through 3 that the variable is now present in the system environment.

Accessing the OpenDaylight controller

Go to your favorite browser and type in **http://127.0.0.1:8080** and you will see the screen below.

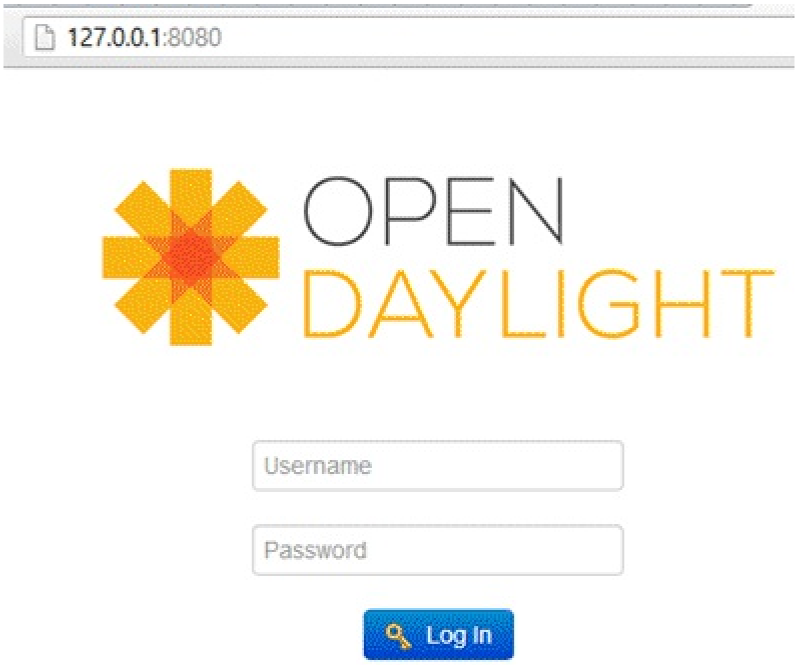


Figure 19.2

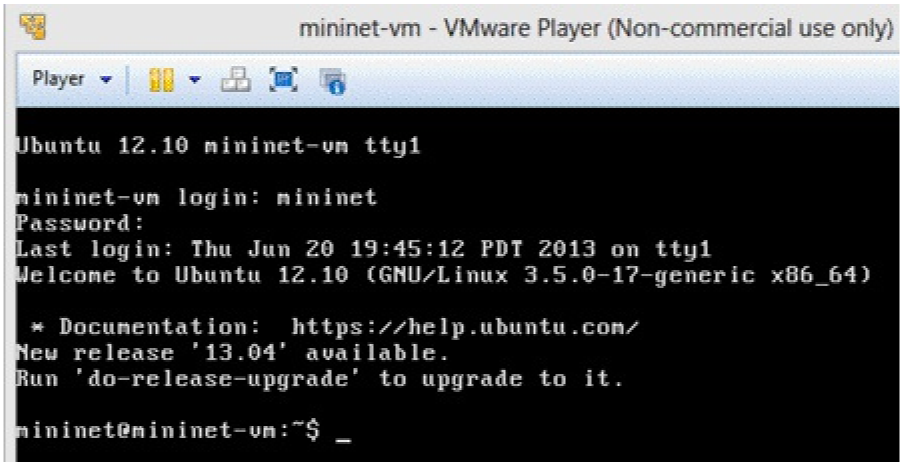
Username is “admin” and the **password** is “admin”

Hooray!! You are logged into the controller. This does not have any switches to control yet but we will be running Mininet next for that

Starting Mininet VM

Start your VMware player and click on the Mininet-vm to start the Mininet virtual machine. This starts up very quick and you will see something like the screen shot below. If VMware player prompts you to install VMware tools you can ignore that. Also the prompt in the virtual machine to upgrade Ubuntu should also be ignored.

The **username** is “mininet” and the **password** is also “mininet”.



```
Ubuntu 12.10 mininet-vm tty1

mininet-vm login: mininet
Password:
Last login: Thu Jun 20 19:45:12 PDT 2013 on tty1
Welcome to Ubuntu 12.10 (GNU/Linux 3.5.0-17-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
New release '13.04' available.
Run 'do-release-upgrade' to upgrade to it.

mininet@mininet-vm:~$ _
```

Figure 19.3

Find the ip address of your mininet by running the “ifconfig” command. You should note the ip address of eth0.

Starting Xming

Just doubleclick the Xming icon on your desktop, to start Xserver. There is nothing visible except for an icon in your system tray.

Using putty to SSH into Mininet VM

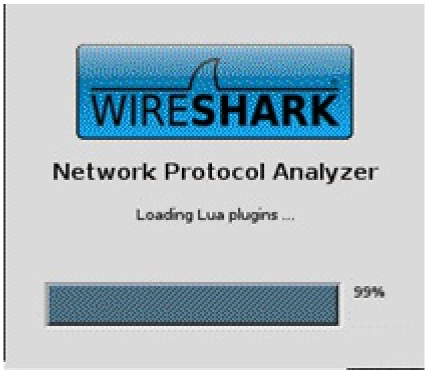
Use the “ifconfig” command in the Mininet window to find out the ip address of the Mininet virtual machine. Note that down as you will be using that to ssh into it using putty.

SSH into the Mininet virtual machine. After entering the username and password, start Wireshark as described in the next section.

Starting Wireshark

Wireshark is already installed in the Mininet virtual machine. You can run it by entering the command

“*sudo wireshark*”. If Xming server is running as intended you will see something like the screenshot below



WIRESHARK

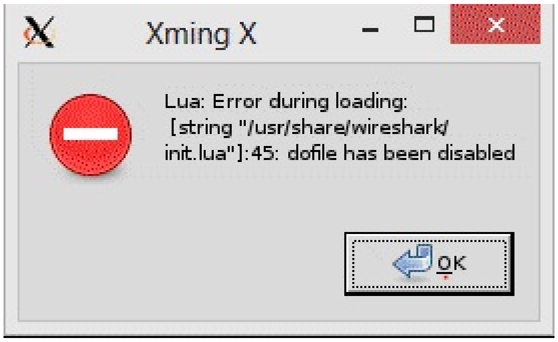
Network Protocol Analyzer

Loading Lua plugins ...

99%

Figure 19.4

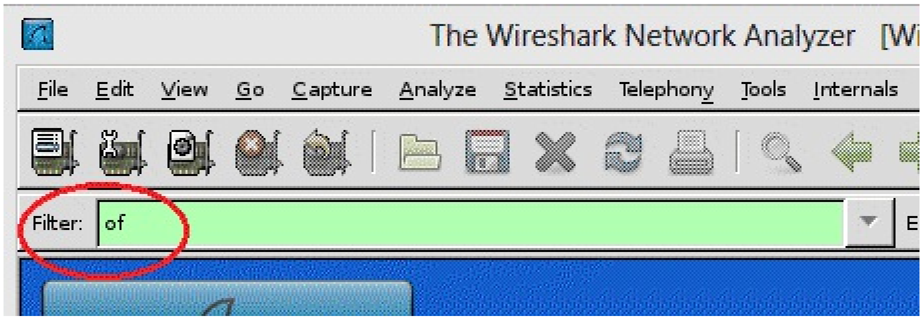
You may see this error message below. Ignore it.



Lua: Error during loading:
[string "/usr/share/wireshark/init.lua"]:45: dofile has been disabled

Figure 19.5

Lastly when the Wireshark GUI is up, enter “of” to filter openflow traffic as seen in the screenshot below.


The Wireshark Network Analyzer [Wi
File Edit View Go Capture Analyze Statistics Telephony Tools Internals
E

Filter: of

Figure 19.6

If you started Wireshark without root/superuser privilege then you will not see any interfaces to capture packets from.

Creating a small network

SSH into mininet and use the following command to create you small network.

sudo mn --mac --controller=remote,ip=x.x.x.x,port=6633

sudo – runs the command with root or superuser privileges

--mac – makes the mac address of your hosts similar to the IP address. This makes it easier to read a Wireshark trace.

--controller=remote – tells mininet that the SDN controller is not on the local machine

ip=x.x.x.x – specify the IP Address of the SDN controller

port=6633 – This is the standard port number to connect to the controller.

You will see something like this

```
mininet@mininet-vm:~$ sudo mn --mac --controller=remote, ip=192.168.57.144, port=6633
```

```
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
*** Starting 1 switches s1
*** Starting CLI:
mininet>
```

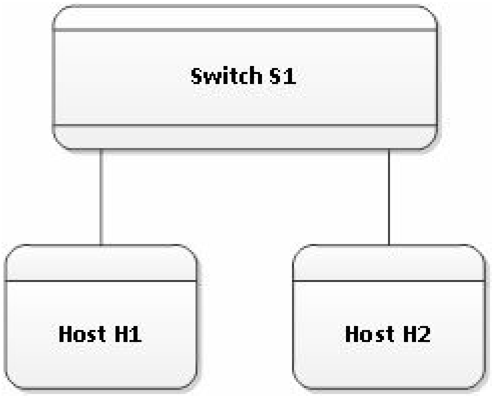


Figure 19.7

Your prompt will change to **mininet>**

On the mininet prompt use the commands like **dump** or **net** to see the details of the network created. It is a small network with one switch and two hosts connected to it. See table below.

mininet>net	**mininet>dump**
c0	<RemoteController c0: 192.168.57.144:6633 pid=2076>
s1 lo: s1-eth1:h1-eth0 s1-eth2:h2-eth0	<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=2087>
h1 h1-eth0:s1-eth1	<Host h1: h1-eth0:10.0.0.1 pid=2083>
h2 h2-eth0:s1-eth2	<Host h2: h2-eth0:10.0.0.2 pid=2084>

Figure 19.8

Your lab is up and running. For my setup I minimized both the VM's and opened 2 putty sessions on the host machine and one browser window.

1. The Browser window was used to logon to the controller. Use the ip address of the controller that you noted down earlier when you started the OpenDaylight VM (eth0 of Ubuntu VM). Use the **http://IPAddress:8080** command to logon to your controller. As mentioned earlier the

username and password are **admin** and **admin**.

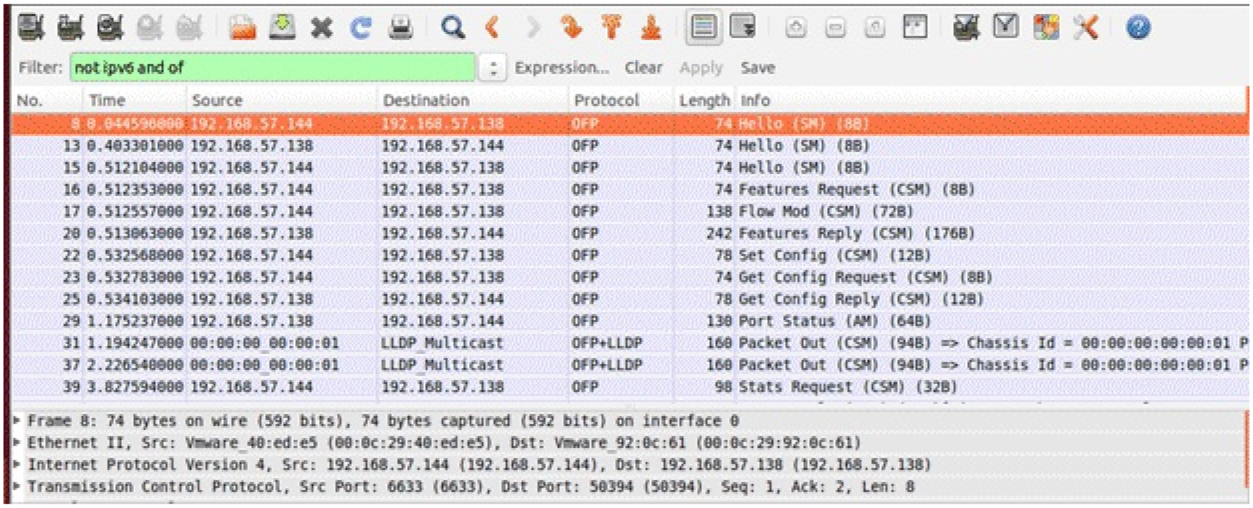
2. The first Putty session was used to ssh into the Mininet VM. You can start, stop and run various commands on the mininet switch from this session. (You should have the IP address of the mininet machine from the earlier step when you started the Mininet VM.
3. The second Putty session was use to ssh into Mininet VM and run Wireshark.

With these three windows I never had to touch the VM's at all. I also did this on purpose because in reality you will be doing this also. You will be logging on to your controller via web interface and will be doing an SSH session to the switches.

CHAPTER 20 – SNOOPING IN ON THE CONVERSATION

Now that we have started the controller and the switch and have established that the switch can see the controller, let us look at the conversation between the switch and the controller as a new switch comes on to the network.

For this purpose we will start Wireshark and put the filter of “**not ipv6 and of**”. I used this filter because there are some IPv6 conversations that will interrupt the flow of information on Wireshark display. With this filter you get to see a nice flow between the Controller and the switch as seen in the screenshot below. For you reference 192.168.57.144 is the controller and 192.168.57.138 is the Mininet switch.



No.	Time	Source	Destination	Protocol	Length	Info
8	0.044596000	192.168.57.144	192.168.57.138	OFP	74	Hello (SM) (8B)
13	0.403301000	192.168.57.138	192.168.57.144	OFP	74	Hello (SM) (8B)
15	0.512104000	192.168.57.144	192.168.57.138	OFP	74	Hello (SM) (8B)
16	0.512353000	192.168.57.144	192.168.57.138	OFP	74	Features Request (CSM) (8B)
17	0.512557000	192.168.57.144	192.168.57.138	OFP	138	Flow Mod (CSM) (72B)
20	0.513063000	192.168.57.138	192.168.57.144	OFP	242	Features Reply (CSM) (176B)
22	0.532568000	192.168.57.144	192.168.57.138	OFP	78	Set Config (CSM) (12B)
23	0.532783000	192.168.57.144	192.168.57.138	OFP	74	Get Config Request (CSM) (8B)
25	0.534103000	192.168.57.138	192.168.57.144	OFP	78	Get Config Reply (CSM) (12B)
29	1.175237000	192.168.57.138	192.168.57.144	OFP	130	Port Status (AM) (64B)
31	1.194247000	00:00:00_00:00:01	LLDP_Multicast	OFP+LLDP	160	Packet Out (CSM) (94B) => Chassis Id = 00:00:00:00:00:01 P
37	2.226540000	00:00:00_00:00:01	LLDP_Multicast	OFP+LLDP	160	Packet Out (CSM) (94B) => Chassis Id = 00:00:00:00:00:01 P
39	3.827594000	192.168.57.144	192.168.57.138	OFP	98	Stats Request (CSM) (32B)

```
▸ Frame 8: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
▸ Ethernet II, Src: Vmware_40:ed:e5 (00:0c:29:40:ed:e5), Dst: Vmware_92:0c:61 (00:0c:29:92:0c:61)
▸ Internet Protocol Version 4, Src: 192.168.57.144 (192.168.57.144), Dst: 192.168.57.138 (192.168.57.138)
▸ Transmission Control Protocol, Src Port: 6633 (6633), Dst Port: 50394 (50394), Seq: 1, Ack: 2, Len: 8
```

Figure 20.1

You can do this for yourself by

a. Stopping the Mininet switch.

b. Stopping any captures that are going on in Wireshark and restarting it with the correct filter applied.

c. Starting Mininet for a few seconds and stopping it again.

You will see in your capture similar to the Wireshark capture above that this is what we talked about in chapter 8. We will be referring to the first column in the above diagram as the packet number. Just in case you are wondering about missing numbers then you will be happy to know that these are the packets that we have filtered out.

1. Hellos from the controller and the switch to each other. (packets 8 and 13)
2. Another hello from the controller that establishes the version of OpenFlow they will talk. (Packet 15)

3. Controller does a feature request which is replied by the switch (Packets 16 and 20)
4. This is followed by the get config request and reply, status requests and much more.
5. The lldp packets you see are used to discover the network.

It would be a good idea to look at each packet in detail in Wireshark and see all the parts of it by looking at the window below the capture window of Wireshark. I am sure you will have questions and many of these can be answered by reading and referring to the OpenFlow 1.0 documentation.

Now that we have two hosts connected to the switch we should let them ping each other.

At the **mininet>** prompt Use the command **h1 ping h2** or **pingall**

Please note that h1 ping h2 substitutes the correct IP addresses and pingall makes all hosts ping each other. This means that if you have a network with 20 hosts this command becomes interesting and will a 1000 hosts it will become a nuisance. This command is there to generate network traffic and also test connectivity on small networks, although no one stops you from running this on a bigger network.

Will the hosts be able to ping each other? Why or why not? Pause and think about this and proceed to the next chapter.

Note: You can disable IPv6 on Ubuntu very easily by adding these lines to “**sysctl.conf**”

```
# IPv6 configuration

net.ipv6.conf.all.disable_ipv6 = 1

net.ipv6.conf.default.disable_ipv6 = 1

net.ipv6.conf.lo.disable_ipv6 = 1>
```

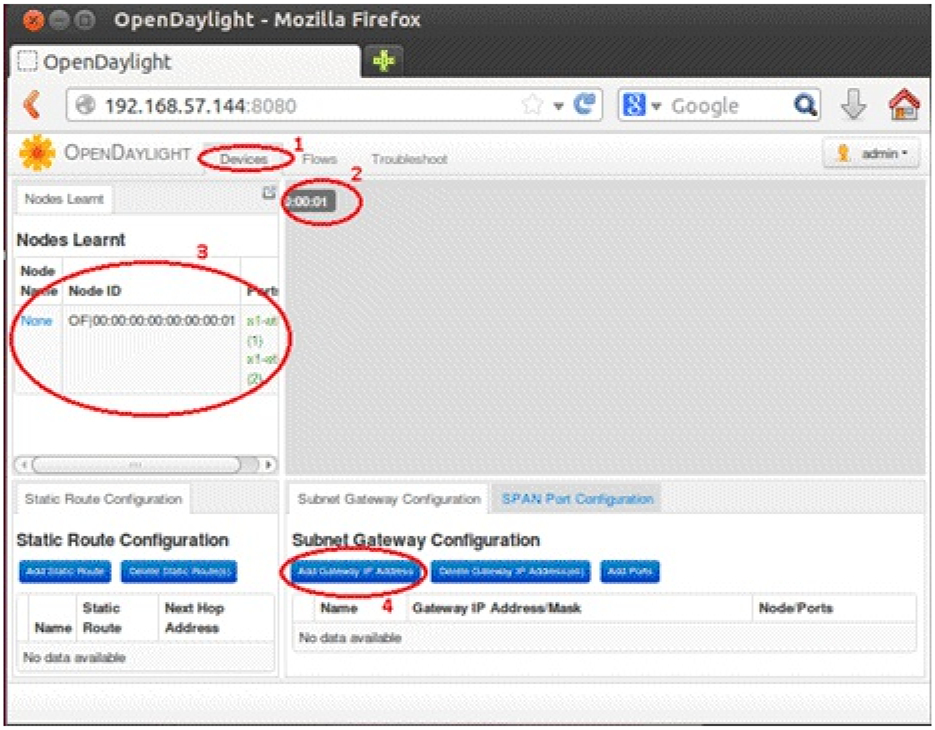
Then reload sysctl.conf using the command sudo sysctl -p

CHAPTER 21 – RING !! RING !!...... PING!!

Starting where we left off in the last chapter, you should have seen the pings failed. Why did this happen? I will walk you through the answer to this.

As soon as you logon to the controller you will see a screen similar to this.

1. Shows that you are on the devices tab.
2. Shows that you can partially see your discovered switch. You will need to click anywhere in that window and drag it down to see the whole switch.
3. Shows the details of the node that was learnt. Node name is none but it can be changed by clicking on "None". This also shows that the switch has two Ethernet interfaces *s1-eth(1) and s1-eth(2)*.
4. Show that there is no default gateway for this subnet and it can be added by clicking on the Add gateway IP Address button.



Nodes Learnt

Node Name	Node ID	Ports
None	OF|00:00:00:00:00:00:00:01	s1-et (1) s1-et (2)

Static Route Configuration

Name	Static Route	Next Hop Address
No data available		

Subnet Gateway Configuration

Name	Gateway IP Address/Mask	Node/Ports
No data available		

Figure 21.1

That gives you an idea about why the ping failed, but that is not the reason. Please see screenshot below for the answer. The real reason is that there are no flows there that can facilitate the flow of ping packets.

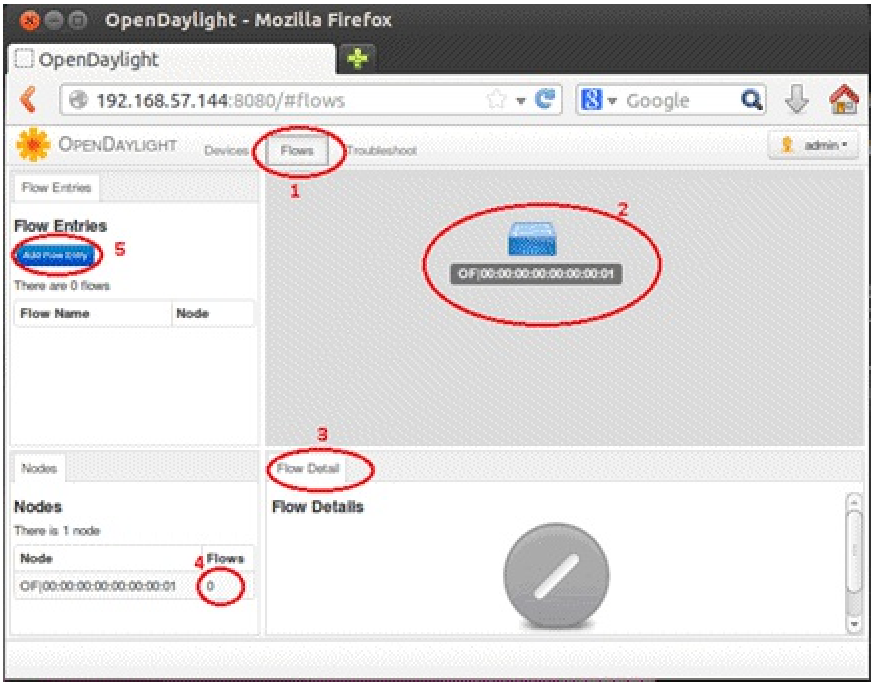
1. Shows that you have to be on the flows tab.

2. Shows how we have dragged and repositioned the switch.

3. Shows that the flow details are blank

4. Shows that the number of flows are 0

5. Shows the button to add flow entries manually



OpenDaylight - Mozilla Firefox
OpenDaylight
192.168.57.144:8080/#flows
Google
OpenDaylight
Devices
Flows
Troubleshoot
admin
1
2
3
4
5
OF|00:00:00:00:00:00:00:01
Add Flow Entry

Flow Entries

Flow Entries

There are 0 flows

Flow Name	Node

Nodes

Nodes

There is 1 node

Node	Flows
OF|00:00:00:00:00:00:00:01	0

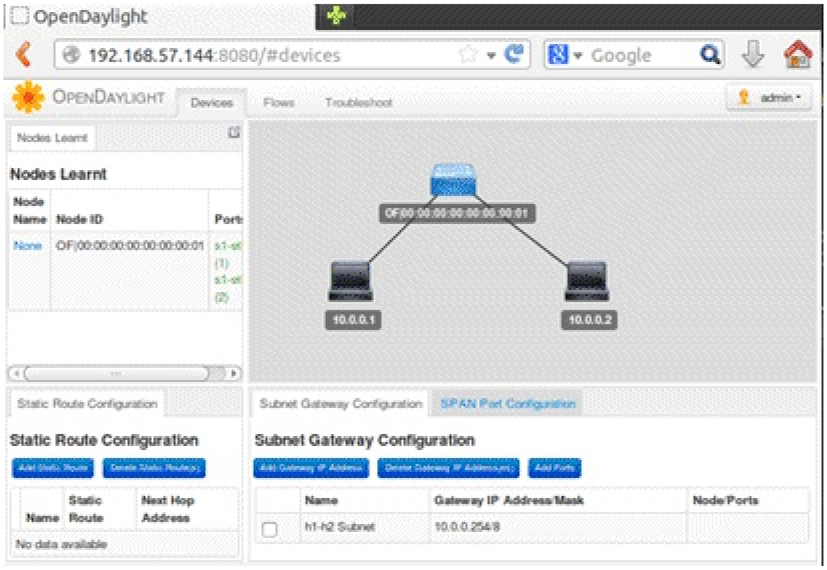
Flow Detail

Flow Details

Figure 21.2

There are multiple ways to solve this.

1. As seen in figure 21.1 earlier, click on the Add Default Gateway and type in 10.0.0.254/8 for the hosts subnet and that makes the controller learn the MAC addresses of the hosts and ping works. See screenshot below. As soon ping starts to work hosts h1 and h2 are seen by the controller. Please note the default gateway that was added.



Nodes Learnt

Nodes Learnt

Node Name	Node ID	Port
None	OF|00:00:00:00:00:00:00:01	s1-et (1) s1-et (2)

Static Route Configuration

Static Route Configuration

	Name	Static Route	Next Hop Address
No data available			

Subnet Gateway Configuration

SPAN Port Configuration

Subnet Gateway Configuration

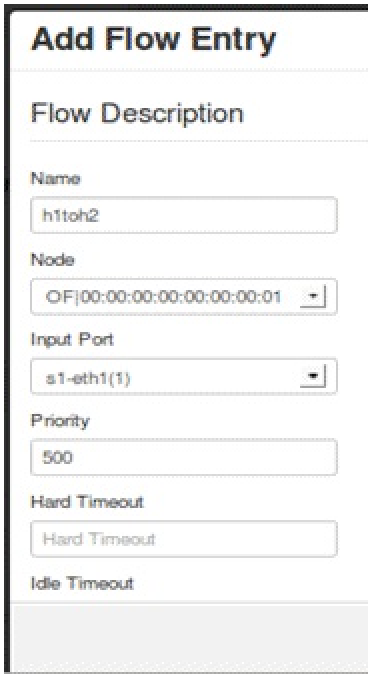
	Name	Gateway IP Address/Mask	Node/Ports
□	h1-h2 Subnet	10.0.0.254/8	

Figure 21.3

2. The second way is a bit long but it is something you should try. After all, it is your playground.

The way is to add flows from h1 to h2 and then add another flow from h2 to h1.

a. To do this you will need to remove the gateway that you added in step 1 and then click on the flows tab to add flows. Click on "**Add flows**" and now you will be amazed by how many options you have to play with here.



Add Flow Entry

Flow Description

Name

h1toh2

Node

OF|00:00:00:00:00:00:00:01

Input Port

s1-eth1(1)

Priority

500

Hard Timeout

Hard Timeout

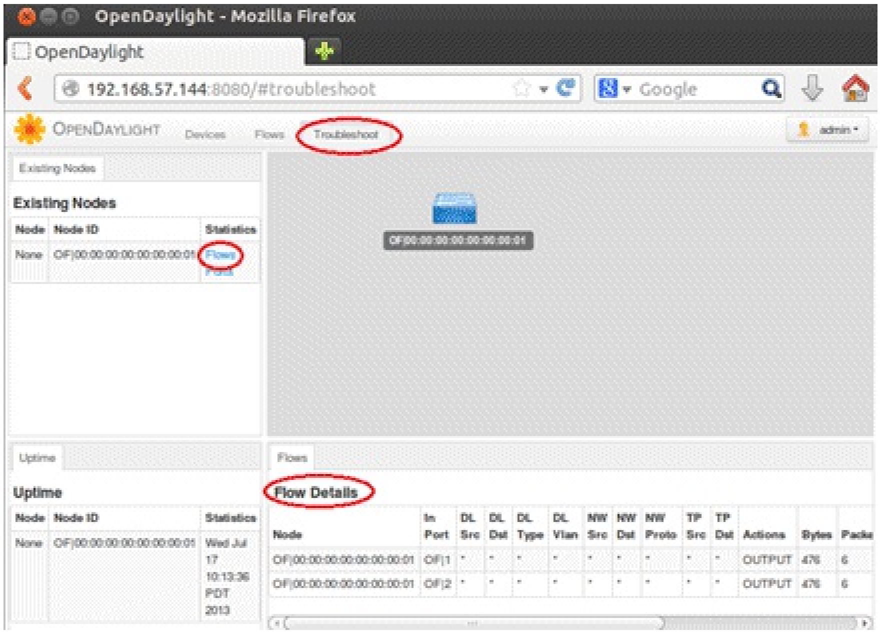
Idle Timeout

Figure 21.4

Scroll down and see the great flexibility you have. You can tweak so many parameters at all layers of the OSI model.

b. Start adding the flows as per the screenshot above. Keep scrolling down and under Layer 2 blank the ethertype field (unless you want to put the ethertype for icmp) and at the bottom under Actions select the action of add output port and choose s1-eth2(2).

c. Now add another flow for h2 to h1 using the step above as guidance. Ping in the mininet window and it should work however you will not see the hosts this time in the window. You will have to click on the troubleshoot tab and click on flows to see the installed flows as seen in the screenshot below.


OpenDaylight - Mozilla Firefox
OpenDaylight
192.168.57.144:8080/#troubleshoot
Google
OpenDaylight
Devices
Flows
Troubleshoot
admin
Existing Nodes
OF|00:00:00:00:00:00:00:01
Uptime
Flows

Existing Nodes

Node	Node ID	Statistics
None	OF|00:00:00:00:00:00:00:01	Flows Ports

Uptime

Node	Node ID	Statistics
None	OF|00:00:00:00:00:00:00:01	Wed Jul 17 10:13:36 PDT 2013

Flow Details

Node	In Port	DL Src	DL Dst	DL Type	DL Vlan	NW Src	NW Dst	NW Proto	TP Src	TP Dst	Actions	Bytes	Packe
OF|00:00:00:00:00:00:00:01	OF|1	*	*	*	*	*	*	*	*	*	OUTPUT	476	6
OF|00:00:00:00:00:00:00:01	OF|2	*	*	*	*	*	*	*	*	*	OUTPUT	476	6

Figure 21.5

3. The third way of making ping between h1 and h2 work is to use the command dpctl. This is an acronym for "Datapath Control". This is a utility that sends commands directly to the switch without the need for a controller. This means that you can install flows without using the GUI above. See Appendix C for DPCTL command reference.

For our case we can run the following command line to make ping work. Please note these commands are not run on the mininet> prompt but on the Ubuntu prompt on the mininet VM as seen below.

```
root@mininet-vm:/# dpctl add-flow tcp:127.0.0.1:6634
in_port=1,idle_timeout=10,actions=output:2
```

```
root@mininet-vm:/# dpctl add-flow tcp:127.0.0.1:6634
in_port=2,idle_timeout=10,actions=output:1
```

Now use the **pingall** command and you will see that ping work flawlessly.

```
mininet> pingall

*** Ping: testing ping reachability

h1 -> h2

h2 -> h1

*** Results: 0% dropped (0/2 lost)
```

You can also see the flows that you added using the **dpctl dump-flows** command as seen below.

```
root@mininet-vm:/#  dpctl dump-flows tcp:127.0.0.1:6634

stats_reply (xid=0xa7a5d648): flags=none type=1(flow)

cookie=0, duration_sec=16s, duration_nsec=153000000s, table_id=0, priority=32768, n_packets=6,
n_bytes=476, idle_timeout=10,hard_timeout=0,in_port=1,actions=output:2

cookie=0, duration_sec=14s, duration_nsec=839000000s, table_id=0, priority=32768, n_packets=6,
n_bytes=476, idle_timeout=10,hard_timeout=0,in_port=2,actions=output:1
```

As you can see the idle timeout for these flows is 10 seconds so if flows are not used for this duration these will be deleted.

This can be verified using the **dpctl dump-flows** command.

You can manually delete the flows using the command below also, and now ping will stop working.

```
root@mininet-vm:/# dpctl del-flows tcp:127.0.0.1:6634
```

If I were you, I would try **all three ways**, it's a great way to learn.

Use the convenient **pingall** command to verify connectivity before and after you put the flows.

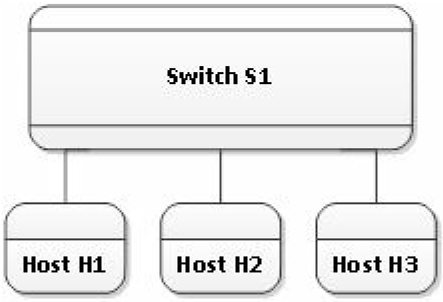
CHAPTER 22 – CREATING MORE TOPOLOGIES

One switch with three hosts

sudo mn --mac --controller=remote,ip=x.x.x.x,port=6633 --topo single,3

- **sudo** – runs the command with root or superuser privileges
- **--mac** – makes the mac address of your hosts similar to the IP address. This makes it easier to read a Wireshark trace.
- **--controller=remote** – tells mininet that the SDN controller is not on the local machine
- **ip=x.x.x.x** – specify the IP Address of the SDN controller
- **port=6633** – This is the standard port number to connect to the controller.
- **--topo single,3** – This specifies that there is a single switch with 3 hosts.

```
mininet@mininet-vm:~$ sudo mn --mac --controller=remote,ip=192.168.57.141,port=6633 --topo tree,2
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s2) (h2, s2) (h3, s3) (h4, s3) (s1, s2) (s1, s3)
*** Configuring hosts
```



```
h1 h2 h3 h4

*** Starting controller

*** Starting 3 switches

s1 s2 s3

*** Starting CLI:

mininet>
```

Figure 22.1

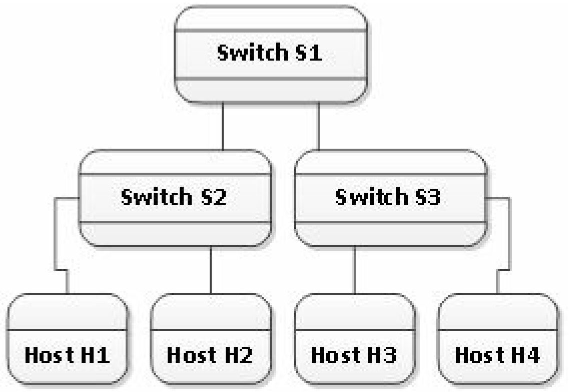
Two level topology with three switches and four hosts

sudo mn --mac --controller=remote,ip=x.x.x.x,port=6633 --topo tree,2

One change **--topo tree, 2** makes this a 2 level topology with 3 switches and 4 hosts

```
mininet@mininet-vm:~$ sudo mn --mac --controller=remote,ip=192.168.57.141,port=6633 --topo tree,2
```

```
*** Creating network

*** Adding controller

*** Adding hosts:

h1 h2 h3 h4

*** Adding switches:

s1 s2 s3

*** Adding links:

(h1, s2) (h2, s2) (h3, s3) (h4, s3) (s1,
s2) (s1, s3)

*** Configuring hosts

h1 h2 h3 h4

*** Starting controller
```



```
*** Starting 3 switches

s1 s2 s3

*** Starting CLI:

mininet>
```

Figure 22.2

Try changing this to a three level tree using the command below and see what happens

sudo mn --mac --controller=remote,ip=x.x.x.x,port=6633 --topo tree,3

Here is something more to make it interesting. You can add delay and packet loss using the command below or something similar. Isn’t this great?

sudo mn --link tc,delay='10ms',loss=5,max_queue_size=3

This creates a minimal topology of two hosts connecting to a switch with a latency of 10 ms, a packet loss of 5% on each link and a buffer size of 3.

You can also use the linear option in topology as below.

sudo mn --mac --controller=remote,ip=x.x.x.x,port=6633 --topo linear,5

CHAPTER 23 – CREATING CUSTOM TOPOLOGIES

Here is something that you will love. Creating custom topologies to suit your needs. To start with there is a sample python script “topo-2sw-2host.py” under /home/mininet/mininet/custom directory. This creates a simple two hosts (h1 and h2) two switches (s3 and s4) network. Although you have already created this network earlier, but if you look at this file, you will understand the logic which will help you create bigger topologies.

Running the Custom Topology

You can run this custom topology using the following command.

```
sudo mn --custom /home/mininet/mininet/custom/topo-2sw-2host.py --topo mytopo
```

Example for Custom Topology

Here is my example file for the topology in **Figure 22.2** in the last chapter.

Create this file using your favorite text editor in /home/mininet/mininet/custom directory on your Mininet VM.

””””Start of custom topology viv3level.py””””

```
from mininet.topo import Topo


class MyTopo( Topo ):
    "Two level topology example."
    def __init__( self ):
        "Create custom topo."
        # Initialize topology
        Topo.__init__( self )
        # Add hosts and switches

        host11 = self.addHost( 'h11' )

        host12 = self.addHost( 'h12' )

        host21 = self.addHost( 'h21' )

        host22 = self.addHost( 'h22' )

        switch1 = self.addSwitch( 's1' )

        switch2 = self.addSwitch( 's2' )

        switch3 = self.addSwitch( 's3' )


        # Add links

        self.addLink( host11, switch1 )

        self.addLink( host12, switch1 )

        self.addLink( host21, switch2 )

        self.addLink( host22, switch2 )

        self.addLink( switch1, switch3 )

        self.addLink( switch2, switch3 )


topos = { 'mytopo': ( lambda: MyTopo() ) }
```

You can run this file using the following command

```
sudo mn --custom mininet/custom/viv3level.py --topo mytopo
```

if you want some more fun try this

```
sudo mn --custom mininet/custom/viv3level.py --topo mytopo --test iper –v info
```

I hope this gives you the power to create more topologies that are of more interest to you.

A Custom Topology Tool

I also found this great webtool for those who are used the dynamips/dynagen/GNS3.

http://ramonfontes.com/vnd/bin-debug/network_new.html#

This is a graphic tool that will let you create topologies in a GUI.

Appendices

Appendix A - List of OpenFlow Software Projects

There are a huge number of SDN projects from many companies. Even if I am able to make a comprehensive list of all the projects today it will be outdated within a week.

You will be amazed to see that there are so many companies working on different pieces if the SDN puzzle. Here is a very small list of these projects.

- **SDN Controllers**
  - Floodlight
  - Opendaylight
  - Beacon
  - POX
  - NOX
  - RYU
- **Simulation and testing**
  - Mininet
  - NICE
  - Cbench
  - OFTest
  - OFLOPS
  - ODDissector
- There are many more like
  - FlowVisor

o Maestro

o Open Stack Quantum

o FortNOX

Roy Chua and team at SDN Central have a phenomenal website for SDN and maintain a comprehensive list of SDN projects that can be found at the link below

http://www.sdncentral.com/comprehensive-list-of-open-source-sdn-projects/

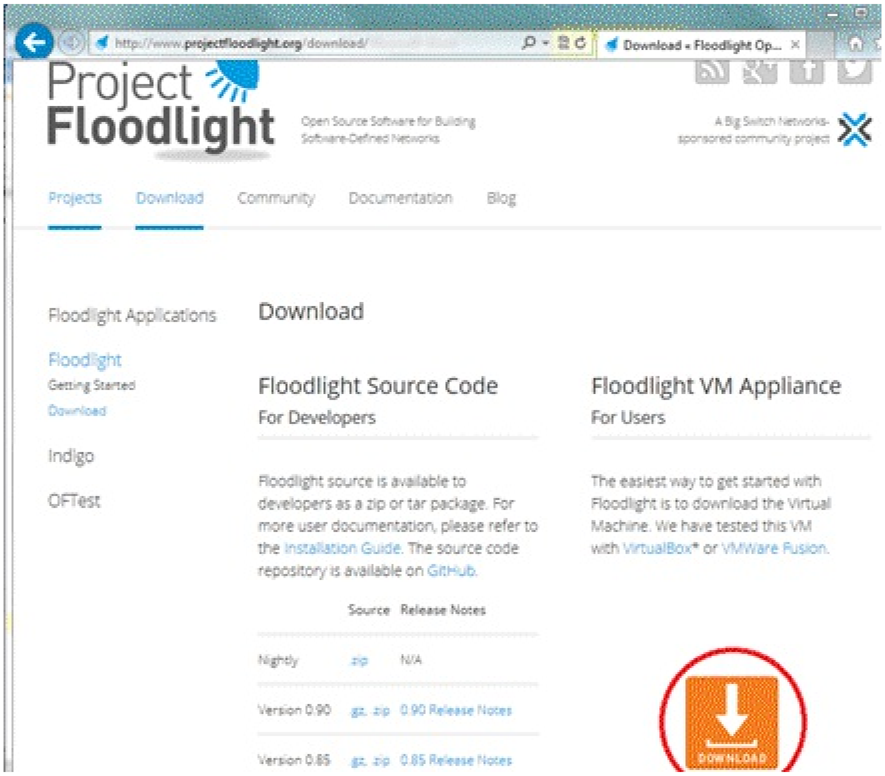
There is also a great list compiled by Mr. Martin Casado that can be found at the link below

http://yuba.stanford.edu/~casado/of-sw.html

APPENDIX B - SETTING UP FLOODLIGHT CONTROLLER

Download Floodlight controller virtual machine

Optionally you can download the "Floodlight" OpenFlow controller from projectfloodlight.org. Click on download to go to downloads as page as seen below.


http://www.projectfloodlight.org/download/
Download « Floodlight Op...
Project Floodlight
Open Source Software for Building Software-Defined Networks
A Big Switch Networks-sponsored community project
Projects
Download
Community
Documentation
Blog
Floodlight Applications
Floodlight
Getting Started
Download
Indigo
OFTest
DOWNLOAD

Download

Floodlight Source Code

For Developers

Floodlight source is available to developers as a zip or tar package. For more user documentation, please refer to the Installation Guide. The source code repository is available on GitHub.

	Source	Release Notes
Nightly	.zip	N/A
Version 0.90	.gz, .zip	0.90 Release Notes
Version 0.85	.gz, .zip	0.85 Release Notes

Floodlight VM Appliance

For Users

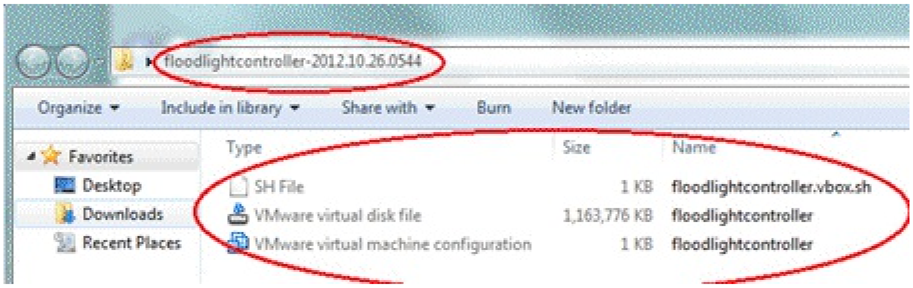
The easiest way to get started with Floodlight is to download the Virtual Machine. We have tested this VM with VirtualBox* or VMWare Fusion.

Download the Floodlight VM Appliance. 338 MB

Setting up Floodlight Controller VM.

If your download for the Floodlight controller was successful, you should have this file "**floodlight-vm-0.90.zip**". Unzip this file and you will have the following folder

floodlightcontroller-2012.10.26.0544 with 3 files as seen below.



Type	Size	Name
SH File	1 KB	floodlightcontroller.vbox.sh
VMware virtual disk file	1,163,776 KB	floodlightcontroller
VMware virtual machine configuration	1 KB	floodlightcontroller

Now you can run the controller in VMware and let the mininet switch talk to this like you did for OpenDaylight.

Appendix C – Setting up OpenDaylight Controller

Setting up OpenDaylight controller on Ubuntu

Downloading the pre-requisites for Ubuntu platform

Installing the software that will enable you to run OpenDaylight is very simple in Ubuntu. Logon to you Ubuntu VM and do the following.

- **Download git:**
  - apt-get install git
- **Download Apache Maven:**
  - apt-get install maven
- **Download Java JDK/JRE:**
  - apt-get install openjdk-7-jre
  - apt-get install openjdk-7-jdk

Optionally you can use this **one command**

apt-get install maven git openjdk-7-jre openjdk-7-jdk

This will install all of the above along with anything else that these packages depend on which is very convenient.

Setting up OpenDaylight on Ubuntu

After you are finished getting all the pre-requisites the first thing you do is to download the latest code for OpenDaylight by running the following command.

git clone http://git.opendaylight.org/gerrit/p/controller.git

Now run the following commands:

```
cd controller/opendaylight/distribution/opendaylight/
```

```
mvn clean install
```

This will take 10 to 15 minutes depending on your computer. I had to do this a couple of times as I was getting errors. This was because of memory issue and can be rectified using the following command

```
export MAVEN_OPTS="-Xmx512m -XX:MaxPermSize=256m
```

Run **mvn clean install** again.

You will need to set Java home so that you can run Java from anywhere in the directory structure. For that you will use the following command.

```
export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-amd64
```

This will work but will not survive a reboot of the VM. To make this permanent you will need to edit your .bashrc file and place the line “export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-amd64 “ preferably at the end. The .bashrc file is a hidden file in your home directory. If you are not very unix literate like me then please look this up on internet. Now navigate to the directory below.

```
cd target/distribution.opendaylight-0.1.0-SNAPSHOT-osgipackage/opendaylight
```

and run this command

```
sudo ./run.sh
```

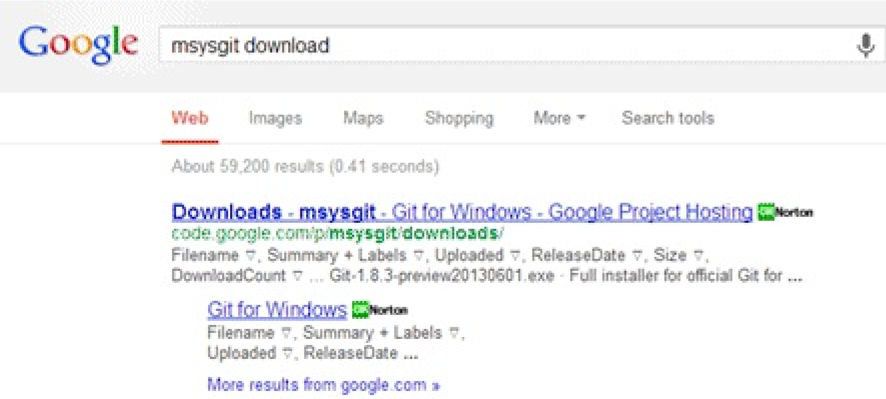
This will start the controller.

Now open a browser window in the VM and browse to http://ipaddress:8080 (substitute the ip address of your VM). You can find the VM IP address using the ifconfig command. Alternatively you can also do http://127.0.0.1:8080 (use the loopback address of the VM) and get to the starting screen of the controller.

Setting up OpenDaylight controller on Windows

Downloading the pre-requite for Windows platform

Download git for windows: I will be using mysisgit. As you can see on the screenshot below if you search for "mysysgit download" the first hit itself is very useful.



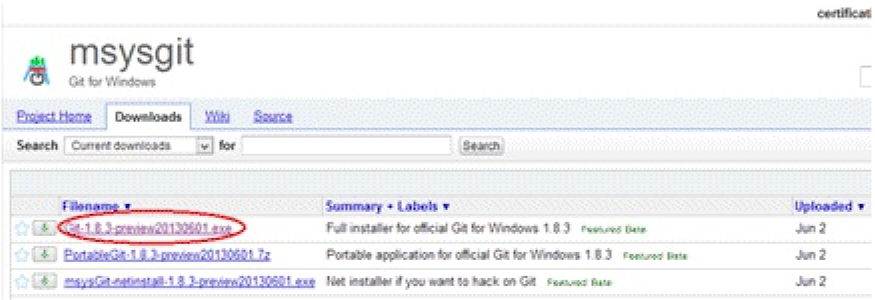
About 59,200 results (0.41 seconds)

Downloads - msysgit - Git for Windows - Google Project Hosting
code.google.com/p/msysgit/downloads/
Filename ▽, Summary + Labels ▽, Uploaded ▽, ReleaseDate ▽, Size ▽, DownloadCount ▽ ... Git-1.8.3-preview20130601.exe · Full installer for official Git for ...

Git for Windows
Filename ▽, Summary + Labels ▽, Uploaded ▽, ReleaseDate ...

More results from google.com »

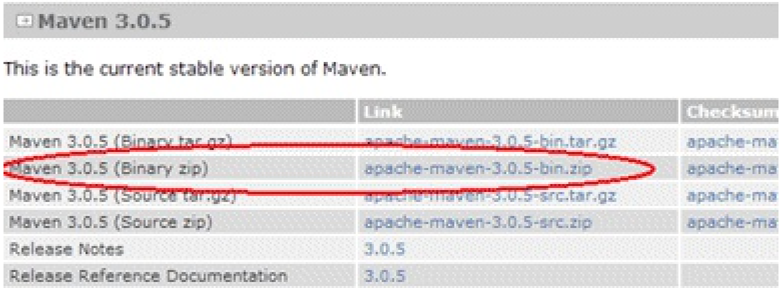
Clicking on the link takes you to the page below at
https://code.google.com/p/msysgit/downloads/list



	Filename ▾	Summary + Labels ▾	Uploaded ▾
☆	Git-1.8.3-preview20130601.exe	Full installer for official Git for Windows 1.8.3 Featured Beta	Jun 2
☆	PortableGit-1.8.3-preview20130601.7z	Portable application for official Git for Windows 1.8.3 Featured Beta	Jun 2
☆	msysGit-netinstall-1.8.3-preview20130601.exe	Net installer if you want to hack on Git Featured Beta	Jun 2

Download "Git-1.8.3-preview20130601.exe".

Download Apache Maven: Just search for "apache maven download" and the first search hit takes you to http://maven.apache.org/download.cgi. Since version 3.1.0 is in alpha stage I recommend that you download version 3.05 zip file as per the screenshot below.

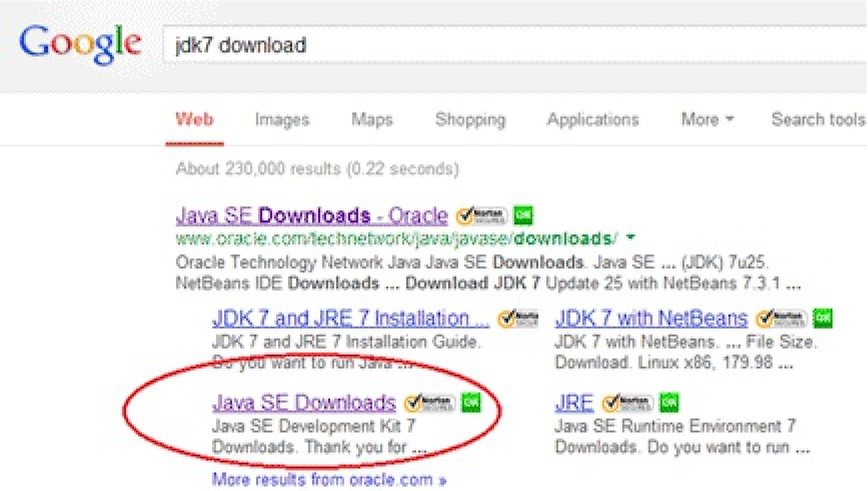


Maven 3.0.5

This is the current stable version of Maven.

	Link	Checksum
Maven 3.0.5 (Binary tar.gz)	apache-maven-3.0.5-bin.tar.gz	apache-ma
Maven 3.0.5 (Binary zip)	apache-maven-3.0.5-bin.zip	apache-ma
Maven 3.0.5 (Source tar.gz)	apache-maven-3.0.5-src.tar.gz	apache-ma
Maven 3.0.5 (Source zip)	apache-maven-3.0.5-src.zip	apache-ma
Release Notes	3.0.5	
Release Reference Documentation	3.0.5	

Download Java JDK/JRE: Just search for "jdk7 download" and click on the link as in the screenshot below.



jdk7 download

About 230,000 results (0.22 seconds)

Java SE Downloads - Oracle
www.oracle.com/technetwork/java/javase/downloads/
Oracle Technology Network Java Java SE Downloads. Java SE ... (JDK) 7u25. NetBeans IDE Downloads ... Download JDK 7 Update 25 with NetBeans 7.3.1 ...

JDK 7 and JRE 7 Installation ...
JDK 7 and JRE 7 Installation Guide. Do you want to run Java ...

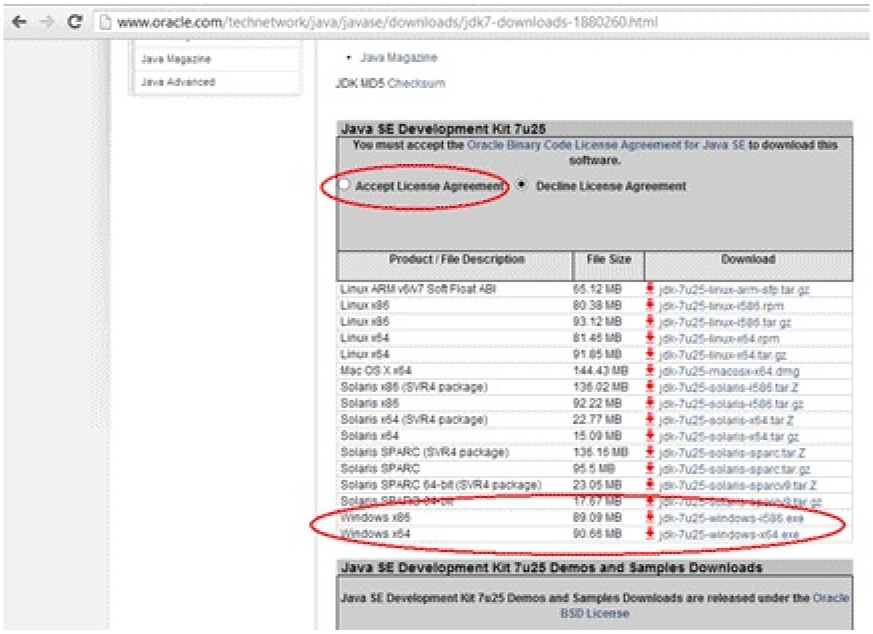
JDK 7 with NetBeans
JDK 7 with NetBeans. ... File Size. Download. Linux x86, 179.98 ...

Java SE Downloads
Java SE Development Kit 7 Downloads. Thank you for ...

JRE
Java SE Runtime Environment 7 Downloads. Do you want to run ...

More results from oracle.com »

You will have to accept the license agreement (see screen shot below) before you can download. Now download according to your operating system. Of the two windows versions highlighted in the screenshot below I downloaded the 64bit version.



www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html

- Java Magazine

JDK MD5 Checksum

Java SE Development Kit 7u25

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

○ Accept License Agreement ● Decline License Agreement

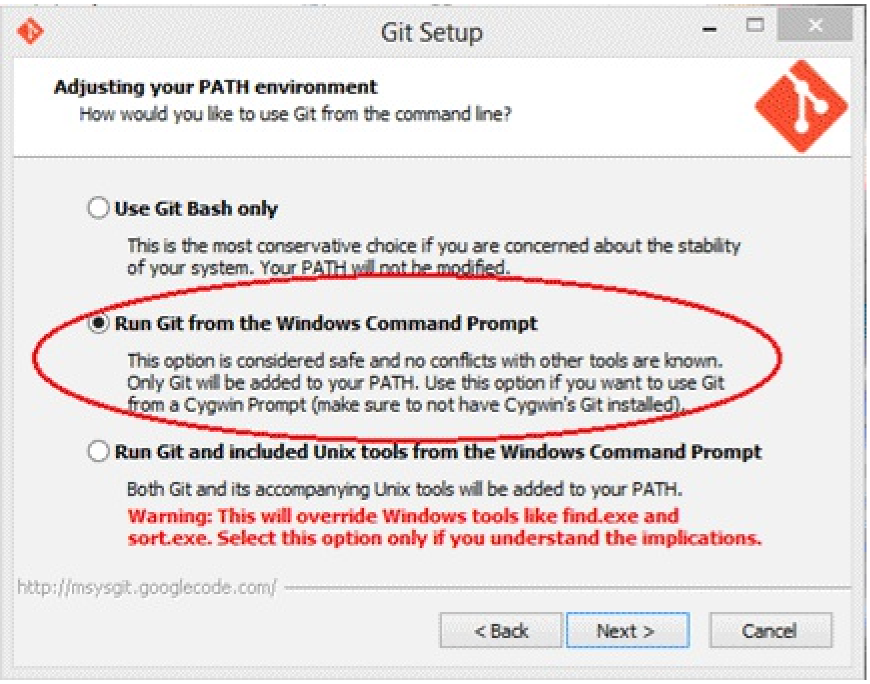
Product / File Description	File Size	Download
Linux ARM v6/v7 Soft Float ABI	65.12 MB	jdk-7u25-linux-arm-sfp.tar.gz
Linux x86	80.38 MB	jdk-7u25-linux-i586.rpm
Linux x86	93.12 MB	jdk-7u25-linux-i586.tar.gz
Linux x64	81.46 MB	jdk-7u25-linux-x64.rpm
Linux x64	91.85 MB	jdk-7u25-linux-x64.tar.gz
Mac OS X x64	144.43 MB	jdk-7u25-macosx-x64.dmg
Solaris x86 (SVR4 package)	136.02 MB	jdk-7u25-solaris-i586.tar.Z
Solaris x86	92.22 MB	jdk-7u25-solaris-i586.tar.gz
Solaris x64 (SVR4 package)	22.77 MB	jdk-7u25-solaris-x64.tar.Z
Solaris x64	15.09 MB	jdk-7u25-solaris-x64.tar.gz
Solaris SPARC (SVR4 package)	136.16 MB	jdk-7u25-solaris-sparc.tar.Z
Solaris SPARC	95.5 MB	jdk-7u25-solaris-sparc.tar.gz
Solaris SPARC 64-bit (SVR4 package)	23.05 MB	jdk-7u25-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	17.67 MB	jdk-7u25-solaris-sparcv9.tar.gz
Windows x86	89.09 MB	jdk-7u25-windows-i586.exe
Windows x64	90.66 MB	jdk-7u25-windows-x64.exe

Java SE Development Kit 7u25 Demos and Samples Downloads

Java SE Development Kit 7u25 Demos and Samples Downloads are released under the Oracle BSD License

Setting up OpenDaylight on Windows

Step 1: Install git for windows. This is very straight forward except that I chose the option to run git at the “Command Prompt” in windows as seen in the screenshot below.



Git Setup

Adjusting your PATH environment

How would you like to use Git from the command line?

○ **Use Git Bash only**

This is the most conservative choice if you are concerned about the stability of your system. Your PATH will not be modified.

● **Run Git from the Windows Command Prompt**

This option is considered safe and no conflicts with other tools are known. Only Git will be added to your PATH. Use this option if you want to use Git from a Cygwin Prompt (make sure to not have Cygwin's Git installed).

○ **Run Git and included Unix tools from the Windows Command Prompt**

Both Git and its accompanying Unix tools will be added to your PATH.

Warning: This will override Windows tools like find.exe and sort.exe. Select this option only if you understand the implications.

http://msysgit.googlecode.com/

Step 2: Unzip Apache Maven to a folder and add it to the system path. It is very helpful to have this run from anywhere on the system because changing folders with the command prompt needs a lot of typing. I personally unzipped it to C:\ and so I had to add the path C:\apache-maven-3.0.5\bin to my system path. Adding path in windows 7 is not very easy so I am putting a screen shot for reference.

You should start by

Step 1: Right click on computer Icon Computer on your desktop and click on “properties”. A new window will open.

Step 2: Now click on advanced system settings. Another window of “System Properties” will open.

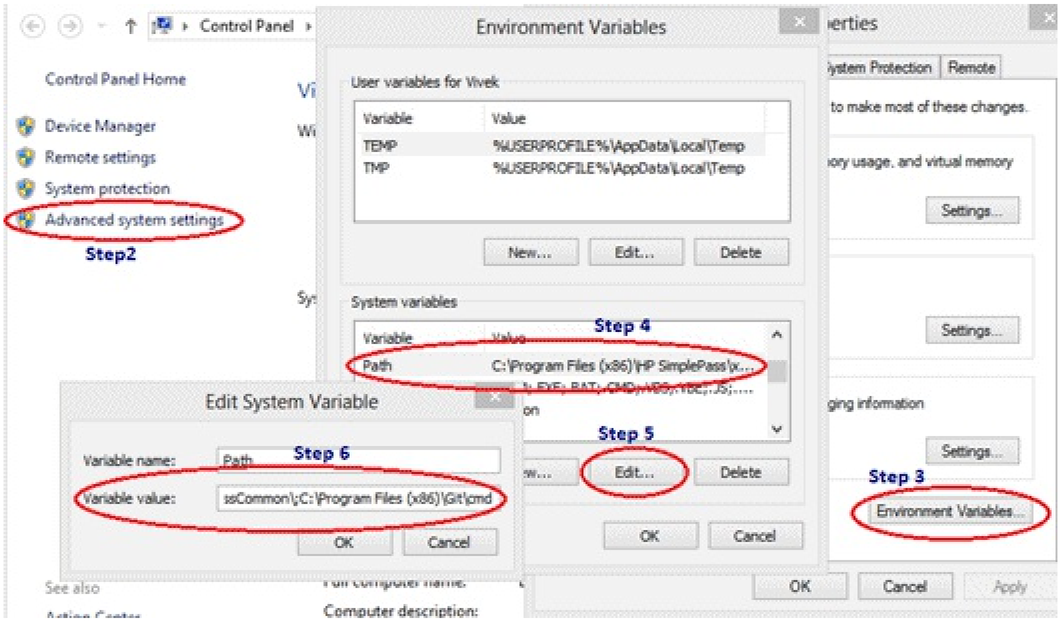
Step 3: Now click on “Environment Variables” and that window will open.

Step 4: Scroll down slowly in the lower system variable area till you see “Path” on the left hand side. Click on path to highlight that.

Step 5: Now click on edit. A new edit system variable window will open.

Step 6: Now you can edit the path. **BE VERY CAUTIOUS** Make sure that you do not overwrite any other variables. If you accidentally made a mistake just click on cancel and go back to step 5.

See screenshot below for reference.



User variables for Vivek

Variable	Value
TEMP	%USERPROFILE%\AppData\Local\Temp
TMP	%USERPROFILE%\AppData\Local\Temp

System variables

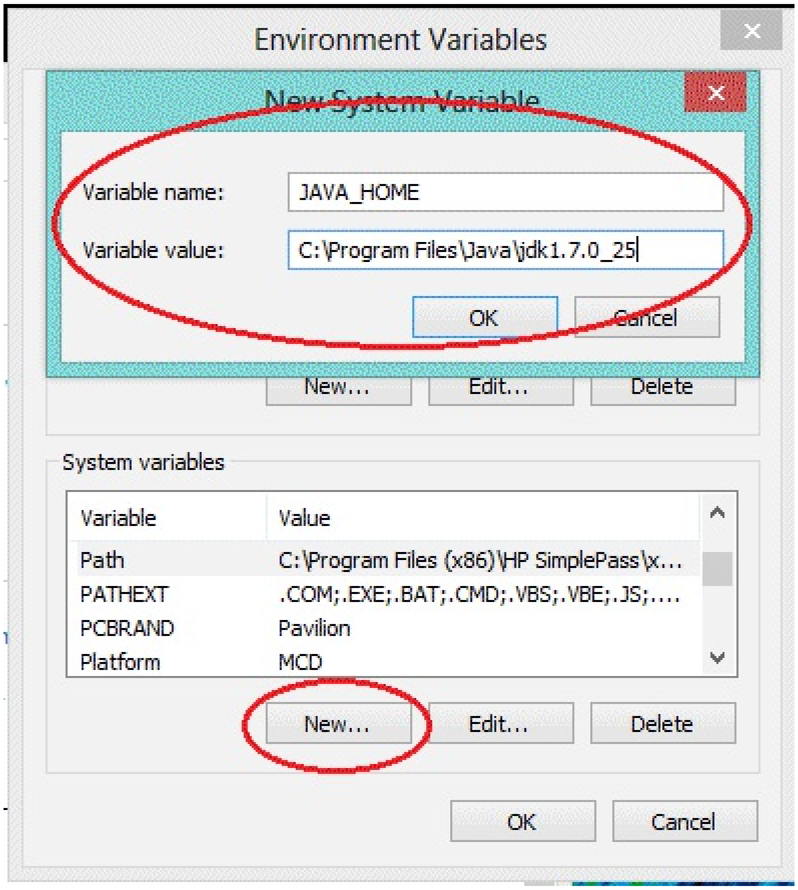
Variable	Value
Path	C:\Program Files (x86)\HP SimplePass\x...

Edit System Variable

Variable name: Path

Variable value: ssCommon\;C:\Program Files (x86)\Git\cmd

Step 3: Now install Java. Once it is installed verify the path of the folder where it is installed. For me it was **C:\Program Files\Java\jdk1.7.0_25**. This path has to be added to system variables. You will need to follow the same steps 1 through 3 as you did to edit system path but on step 4 you will click on "New" and then add a new system variable of JAVA_HOME as seen in the screen shot below.



Variable name: JAVA_HOME

Variable value: C:\Program Files\Java\jdk1.7.0_25

System variables

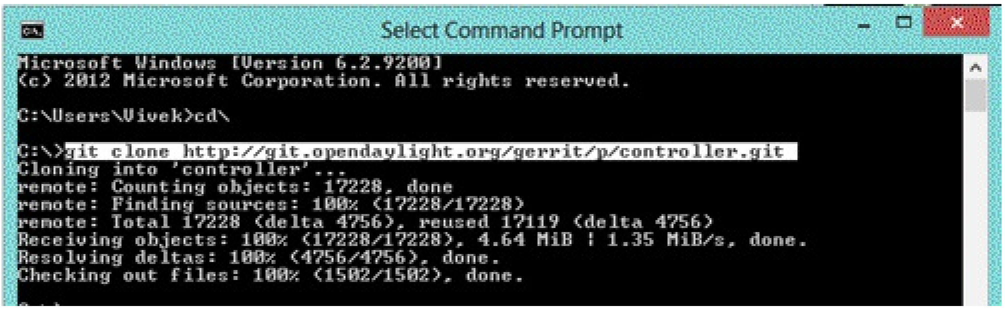
Variable	Value
Path	C:\Program Files (x86)\HP SimplePass\x...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;....
PCBRAND	Pavilion
Platform	MCD

I have used many screen shots in this section so that these serve as a guide other sections also.

All of this sets the stage for the OpenDaylight controller now.

Open a command prompt and type in this command

git clone http://git.opendaylight.org/gerrit/p/controller.git



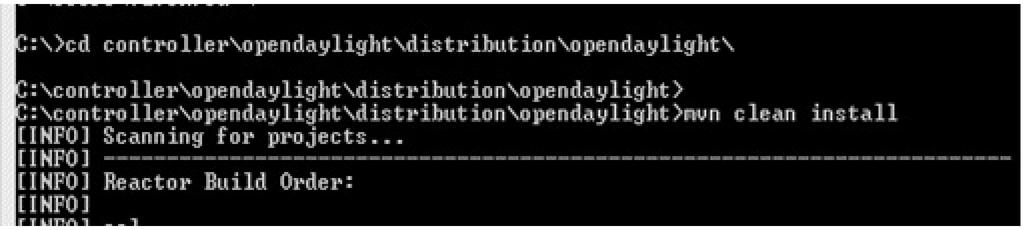
```
Microsoft Windows [Version 6.2.9200]
(c) 2012 Microsoft Corporation. All rights reserved.

C:\Users\Vivek>cd\

C:\>git clone http://git.opendaylight.org/gerrit/p/controller.git
Cloning into 'controller'...
remote: Counting objects: 17228, done
remote: Finding sources: 100% (17228/17228)
remote: Total 17228 (delta 4756), reused 17119 (delta 4756)
Receiving objects: 100% (17228/17228), 4.64 MiB | 1.35 MiB/s, done.
Resolving deltas: 100% (4756/4756), done.
Checking out files: 100% (1502/1502), done.
```

This command cloned the source code for the OpenDaylight controller on to your computer. Once this is complete execute the following commands to compile and make an executable. Please note that I installed it on C:\. Usually it would be installed in your user directory in windows. I also have complete Administrator access on my laptop.

cd controller\opendaylight\distribution\opendaylight\

mvn clean install

See screenshot below for reference.



```
C:\>cd controller\opendaylight\distribution\opendaylight\

C:\controller\opendaylight\distribution\opendaylight>
C:\controller\opendaylight\distribution\opendaylight>mvn clean install
[INFO] Scanning for projects...
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Build Order:
[INFO]
```

Now you see the advantage of installing that path to maven earlier. This will keep going for about 10 minutes. If you are prompted by the firewall about this program asking for access to the internet please allow that. Once this is completed we have a working version of OpenDaylight controller.

Now change your directory to

C:\controller\opendaylight\distribution\opendaylight\target\distribution.opendaylight-0.1.0-SNAPSHOT-osgipackage\opendaylight

You can use windows explorer to navigate to this directory.

Inside this folder you will see a few folders and “run.bat”. This is the batch file that will start up the controller which we will be doing in the next chapter.

Appendix D - Mininet Reference

mn starts mininet in a minimal topology of one switch and two hosts.

mn --help brings up the details of various mininet options

mn --version tells you the version number of the mininet software running

mn -c clean mininet if it crashes so that it starts clean on restart

> ***Note**: You may have to use sudo in front of these commands as mininet needs to run as root. So the above commands will look like below*

```
sudo mn

sudo mn --help

sudo mn –version

sudo mn –c

sudo mn –v debug
```

Once you start mininet the prompt changes to **mininet>**

Here are a few useful commands in mininet.

```
mininet> help

mininet> nodes

mininet> net

mininet> dump

mininet> h1 ifconfig -a

mininet> s1 ifconfig -a

mininet> h1 ps -a
mininet> h1 ping -c 1 h2
```

```
mininet> pingall
```

```
mininet> py dir(h2)
```

```
mininet> py h2.IP(),h2.MAC()
```

```
mininet> link s1 h2 down
```

Run a simple webserver on h1

```
mininet> h1 python -m SimpleHTTPServer 80 &
```

Get a webpage from h1.

```
mininet> h2 wget -O - h1
```

...

Kill the webserver

```
mininet> h1 kill %python
```

```
mininet> exit
```

APPENDIX E - DPCTL REFERENCE

Some useful DPCTL commands

dpctl show

dpctl dump-flows

dpctl del-flows

dpctl add-flow

dpctl –help gives you all the options below

```
ovs-dpctl: Open vSwitch datapath management utility

usage: ovs-dpctl [OPTIONS] COMMAND [ARG...]

  add-dp DP [IFACE...]     add new datapath DP (with IFACEs)

  del-dp DP                delete local datapath DP

  add-if DP IFACE...       add each IFACE as a port on DP

  set-if DP IFACE...       reconfigure each IFACE within DP

  del-if DP IFACE...       delete each IFACE from DP

  dump-dps                 display names of all datapaths

  show                     show basic info on all datapaths

  show DP...               show basic info on each DP

  dump-flows DP            display flows in DP

  del-flows DP             delete all flows from DP

Each IFACE on add-dp, add-if, and set-if may be followed by

comma-separated options.  See ovs-dpctl(8) for syntax, or the

Interface table in ovs-vswitchd.conf.db(5) for an options list.
```

```
Logging options:
  -v, --verbose=MODULE[:FACILITY[:LEVEL]]  set logging levels
  -v, --verbose           set maximum verbosity level
  --log-file[=FILE]       enable logging to specified FILE
  (default: /var/log/openvswitch/ovs-dpctl.log)


Other options:
  -t, --timeout=SECS          give up after SECS seconds
  -h, --help                  display this help message
  -V, --version               display version information
```

APPENDIX F – MAKING YOUR OWN OPENFLOW SWITCH

There is a project named Pantou which lets you convert a home wireless router into a OpenFlow enabled switch. This is based on the OpenWrt platform. This means that you can have your own OpenFlow enabled device for as little as USD 60. Chances are that you may have one of the older Linksys devices that are supported for this lying around in your house or with a friend.

This was something very exciting for me as now I could learn and experiment with real traffic. It was not just a simulation anymore.

Be careful when you are doing this because you can turn your working device into a paper weight. So read about it and do some research before jumping in.

Go to the URL below

http://www.openflow.org/wk/index.php/Pantou_:_OpenFlow_1.0_for_OpenWRT

Or just search for "project pantou" and you will see a suitable link.

If you have tried DD WRT or are familiar with it then this should not be a problem for you.

Note: *You have read and choose the correct firmware that matches your version of hardware and follow the given procedure flawlessly. If this is not done patiently and carefully you can "BRICK" your router. Which means it turns into a paperweight. There are ways to unbrick certain routers but that is even more difficult.*

APPENDIX G - RESOURCES

- Tail-f systems Survey Press release
- Openflow Specification 1.0.0
- Brents Blog @ networkstatic
- http://www.gartner.com/newsroom/id/2386215
- http://www.forbes.com/sites/ericsavitz/2012/10/22/gartner-10-critical-tech-trends-for-the-next-five-years/
- http://www.enterprisenetworkingplanet.com/datacenter/idc-sdn-a-2-billion-market-by-2016.html
- http://www.idc.com/getdoc.jsp?containerId=prUS23888012
- https://www.opennetworking.org/membership/member-listing
- http://embedded.communities.intel.com/community/en/applications/blog/2012/06/07/insights-into-sdn-for-service-providers-at-linley-carrier-conference
- http://www.lightreading.com/software-defined-networking/service-provider-sdn-gets-real/240157374
- http://mrfogg97.blogspot.com/2013/04/opendaylight-sdn-on-windows.html
- http://guides.beanstalkapp.com/version-control/git-on-windows.html
- http://yuba.stanford.edu/~casado/of-sw.html
- http://ramonfontes.com/vnd/bin-debug/network_new.html#