T.C.
İSTANBUL AYDIN ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

YAZILIM TANIMLI AĞLAR

YÜKSEK LİSANS TEZİ

Habiba AMED

Bilgisayar Mühendisliği Ana Bilim Dalı

Bilgisayar Mühendisliği Programı

Temmuz, 2020

T.C.
İSTANBUL AYDIN ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

YAZILIM TANIMLI AĞLAR

YÜKSEK LİSANS TEZİ

Habiba AMED
(Y1713.010073)

Bilgisayar Mühendisliği Ana Bilim Dalı

Bilgisayar Mühendisliği Programı

Tez Danışmanı: Doç. Dr. Taner ÇEVİK

Temmuz, 2020

ONUR SÖZÜ

Yüksek Lisans tezi olarak sunduğum “Yazılım Tanımlı Ağlar” adlı çalışmanın, tezin proje safhasından sonuçlanmasına kadarki bütün süreçlerde bilimsel ahlak ve geleneklere aykırı düşecek bir yardıma başvurulmaksızın yazıldığını ve yararlandığım eserlerin kaynakçada gösterilenlerden oluştuğunu, bunlara atıf yapılarak yararlanılmış olduğunu belirtir ve onurumla beyan ederim. (05/06/2020)

Habiba Amed

ÖNSÖZ

Bu tezin konusu yazılım tanımlı ağ hakkında bilgi sunmak ve bir örnek ağ tasarlamaktır. Yazılım tanımlı ağda tasarımcılara kolaylık sağlanmıştır çünkü bu ağda yazılım ve donanım bölümleri birbirinden ayırılmıştır ve böylelikle ağ yönetimini kolaylaştırılmıştır. Ve bu ağ türünde yazılım geliştirmek yöntemiyle de ağ tasarlanabilir. Bu tez konusu seçim aşamasından itibaren bana yardım eden tez danışmanım olan Doç.Dr.Taner Çevik'e çok teşekkür ederim. Aynı zamanda bütün aile bireylerime tüm eğitim süresince bana destek verdikleri için teşekkür ederim.

Haziran 2020 Habiba Amed

YAZILIM TANIMLI AĞLAR

ÖZET

Bu tezin konusu yazılım tanımlı ağlardır. Bu tezin amacı yazılım tanımlı ağlar hakkında bilgi sunmak ve aynı zamanda bir örnek ağ tasarlamaktır. Geleneksel ağlarda içerisinde kontrol mekanizması barındıran bir yönlendirici bulunur. Veri düzlemi bu kontrol mekanizmasından alınan emirleri yerine getirir. Aynı zamanda yönlendiricide ayarlamaların yapıldığı bir yönetim düzlemi bulunur. Komşu yönlendiriciden bir paket geldiği zaman kaynak ve hedef bilgisine bakılır ve yönlendirme yazılımı ile yönlendirme tablosuna göre gönderilecek bir sonraki verinin ne olacağına karar verilir. Yönlendirme yazılımı en kısa yola öncelik veren algoritmalar olabileceği gibi farklı metriklere dayalı başka yönlendirme algoritmaları da olabilir. Kontrol düzlemi ağın işletim sistemi gibi davranarak yapılacak işlemlerle ilgili kararlar veren yazılım kısmıdır. Bu yazılımın aldığı karar doğrultusunda da veri düzleminde paket, cihaz (donanım) aracılığıyla ileriye aktarılır. Anahtarın içinde yazılım ile donanım bir arada bulunmaktadır. Bunlar iki farklı yapı olmasına rağmen aynı yerde duran tek varlık gibi davranmaktadırlar. Ne yazık ki geleneksel ağlarda sorun burada ortaya çıkmaktadır. Çünkü Veri düzlemi ve kontrol düzlemi aynı ağ cihazında bulunur. Geleneksel ağlarda olan bu sorunun çözümü merakla arandığı için bu tez yazılmaktadır. Araştırma çoğunlukla internet üzerinden yapılmıştır. Çünkü kütüphanelerde bu konu hakkında veri neredeyse yok denilecek kadar azdır. Verileri internetten toplandıktan sonra analiz edilip daha sonra örnek bir yazılım tanımlı ağ tasarlanıp ve tez yazılmıştır. Özellikle belirtmek gerekir ki bu projede OpenFlow protokol kullanılmıştır ve bu protokolün özelliği kontrol ve veri düzlemini birbirinden ayırmaktan geçmektedir. Projeleri tasarlamak için öncelikle VirtualBox ve Ubuntu 18.04 LTS ve mininet yazılımı kurulmaktadır. Daha sonra test üç tür ağ üzerinde yapılmıştır. Birinci tür ağ tek kontrolü ağ ve ikinci tür ağ ise üç kontrolü ağ ve son ağ ise komut yöntem ile oluşturulmuştur.

Ağlardan elde edilen sonuçlara göre,h1 ve h40 ana makineleri arasında gönderilen ve alınan paket sayısı yedi ve minimum zaman dilimi 0.077 mikro saniyedir. Dolayısıyla h41 ve h81 arasında gönderilen ve alınan paketlerde yedi ve minimum zaman dilimi 0.084 mikro saniyedir. Aynı zamanda h1 ve h68 arasında gönderilen paketlerin sayısı da yedi ve minimum zaman dilimi 0.077mikro saniyedir . h80 ve h120 arasında gönderilen ve alınan paket sayısı ise yedi ve minimum zaman dilimi 0.087 mikro saniyedir.h121 ve h161 arasında gönderilen ve alınan paket sayısı yedi ve minimum zaman dilimi 0.077 mikro saniyedir. h162 ve h200 arasında gönderilen paket sayısı da yedi ve minimum zaman dilimi ise 0.092 mikro saniyedir. Son olarak, h201 ve h243 arasında gönderilen ve alınan paket sayısı gene yedi olup ve minimum zaman dilimi ise 0.0 81 mikro saniyedir.

Anahtar Kelimeler: Yazılım tanımlı ağlar, Kontrol düzlemi, Veri düzlemi, OpenFlow, Pox Kontrolör, Ubuntu 18.04 LTS

SOFTWARE DEFINED NETWORKS

ABSTRACT

The subject of this thesis is software defined networks. The aim of this thesis is to provide information about software defined networks and also to design a sample network. There is a control mechanism inside a router in traditional networks. The data plane executes the orders received from this control mechanism. At the same time, there is a management plane inside the router which adjustments are made. When a packet is received from the adjacent router, source and destination information is checked and the next data to be sent according to the routing software and routing table is determined. Routing software can be algorithms that give priority to the shortest path, or other routing algorithms based on different metrics. The control plane is the software part that acts as the operating system of the network and makes decisions about the operations to be performed. In accordance with the decision of this software, the packet is forwarded in the data plane by means of the device (hardware). There are software and hardware co-exist inside the switch. Although they are two different structures, they behave as the only entity standing in the same place. Unfortunately, in traditional networks the problem arises here. Because the data plane and the control plane are located in the same network device. Since the solution of this problem in traditional networks is curiously searched, this thesis is being written. The research was mostly done over the internet. Because there is almost no data on this topic in libraries. After the data were collected from the internet, it was analyzed and then a sample software defined network was designed and the thesis was written. It should be noted that the OpenFlow protocol is used in this project and the feature of this protocol is to separate the control and data plane. Specifically, it should be noted that the OpenFlow protocol is used in this project and the feature of this protocol is to separate the control and data plane. To design projects first of all VirtualBox and Ubuntu 18.04 LTS and mininet software are installed. The test was then performed on three types of networks. Three types of networks are designed and tested. First type is

the single-controller and the second one is three-controller and the last one is the network, which is created using the mininet command. According to the results from the networks, the number of packets sent and received between the h1 and h40 hosts is seven and the minimum time frame is 0.077 microseconds. Thus, the packets which are sent and received between h41 and h81 are seven and minimum time slots are 0.084 microseconds. At the same time, the number of packets sent between h1 and h68 is the minimum period of 0.077 microseconds. And the number of packets sent and received between h80 and h120 is seven and the minimum time frame is 0.087 microseconds. The number of packets sent and received between h121 and h161 is seven and the minimum time frame is 0.077 microseconds. The number of packets sent and received between h162 and h200 is seven and the minimum time frame is 0.092 microseconds. Finally, the number of packets sent and received between h201 and h243 is still seven, and the minimum time period is 0.0 81 microseconds.

Key Words: Software Defined Networks, Control Plane, Data Plane, OpenFlow, Pox Controller, Ubuntu 18.04 LTS

İÇİNDEKİLER

KISALTMALAR LİSTESİ

AWS : Amazon Web Services

LAN : Local Area Network

MPLS : Multi-Protocol Label Switching

VPNs : Virtual Private Networks

IP **:** Internet Protocol

GUI : Graphical User Interface

SNMP : Simple Network Management Protocol

SDN : Software Defined Networking

ONF : Open Networking Foundation

I2RS **:** Interface to the Routing System

IETF : Internet Engineering Task Force

RIB : Routing Information Base

HAL : Hardware Abstraction Layer

OSS : Operation Support Systems

SDN : Software Defined Networks

BGP : Border Gateway Protocol

API : Application Programming Interface

FIB : Forwarding Information Base

SPB : Shortest Path Bridging

TRILL : Transparent Interconnection of Lots of Links

MAC : Media Access Control

EVPN : Ethernet Virtual Private Network

LISP : Locator / ID Separation Protocol

GRE : Generic Routing Encapsulation

OAM : Operations, Administration, and Maintenance

GPU : Graphics Processing Unit

CPU : Central Processing Unit

NPU : Noise Pick-Up

FPGA : Field – Programmable Gate Array

ASICs **:** Anima Sana in Corpore Sano

OSPF : Open Shortest Path First

ACL : Access Control List

QoS : Quality of Service

ToS : Type of Service

CoS : Class of Service

TCP : Transmission Control Protocol

WAN : Wide Area Network

LSP : Link State Protocol

PCRF : Policy and Charging Rules Function

TDF : Traffic Detection Function

NFV : Network Functions Virtualization

EMS : Element Management System

ICSI : International Computer Services Incorporated

LLDP : Link Layer Discovery Protocol

NAT : Network Address Translation

LSR : Label Switching Router

ONRC : Open Network Research Center

TLS : Transport Layer Security

AMD : Advanced Micro Devices

ŞEKİLLER LİSTESİ

ÇİZELGELER LİSTESİ

I. GİRİŞ

A. Ağ Teknolojilerinin Gelişim Tarihçesi

Birkaç yıl öncesine kadar, depolama, bilgi işlem ve ağ kaynakları kasıtlı bir şekilde fiziksel ve operasyonel olarak birbirinden ayrı tutulmuşlardır. Bu kaynakları (Goda, et al., 2012), (History of Computing, 2017), (Horiuchi, et al., 2017) yönetmek için kullanılan sistemler bile çoğu kez fiziksel olarak ayrılmışlardır. Erişim ilkeleri, sistemler ve erişim prosedürlerini önemli ölçüde içeren operasyonel bir izleme sistemi gibi bu kaynaklardan herhangi biriyle etkileşime giren uygulamalar da güvenlik adına el altında bulundurulmuştur. Bilişim Teknolojileri departmanlarının tercih ettiği yöntemdir. Kuruluşların bu farklı unsurları bir araya getirmeye zorlanması veri merkezi ortamlarında, ucuz bilgi işlem gücü, depolama ve ağ oluşturma işleminin başlatılmasından(ve talep edilmesinden) hemen sonra gerçekleşmektedir. Bu kaynakları yöneten ve işleten uygulamaları, her zamankinden çok daha yakın hale getiren bir paradigma kaymasıdır.

Veri merkezleri (Geng, 2015), aslında geleneksel bilgi işlem elemanlarını (örneğin PC sunucuları), bunlarla ilişkili depoları ve onları kullanıcılarla birbirine bağlayan ağları fiziksel olarak ayırmak için tasarlanmıştır. Bu tür veri merkezlerinde bulunan bilgi işlem gücü, posta sunucusu, veri tabanı sunucuları yada yaygın olarak kullanılan diğer işlevsellik gibi masaüstü kullanıcılara hizmet etmek için uygulamaları yürüten belirli sunucu işlevselliğine odaklanmıştır. Önceden, bir kuruluş içindeki genellikle binlerce (veya daha fazla) masaüstünde yürütülen bu işlevler, yalnızca yerel kullanıma adanmış hizmetler sağlayan departman sunucuları tarafından kullanılıyordu. Departman sunucuları, yönetim kolaylığına olanak sağlamak ve kurumun kullanıcıları arasında paylaşım sağlamak gibi çeşitli nedenlerle zaman geçtikçe veri merkezine taşınmıştır.

10 yıl kadar önce ilginç bir dönüşüm gerçekleşmiştir. VMware adlı bir şirket, popüler Linux dağıtımlarından biri gibi bir ana bilgisayar işletim sisteminin bir veya

daha fazla istemci işletim sistemi (örneğin Windows) çalıştırmasına izin veren ilginç bir teknoloji icat etmiştir (Ward, 2002). VMware'in yaptığı, gerçek bir bilgisayar ortamını (örneğin, sanal NIC, BIOS, ses adaptörü ve video) sentezleyen sanal bir ortam yaratan küçük bir program oluşturmaktır. Daha sonra sanal makineler arasında gerçek kaynakları bir araya getirmişlerdir. Bu denetleyici program bir hipervizör (Parlakyigit, 2013) olarak adlandırılmıştır.

Başlangıçta, VMware, programlama gereksinimlerinin çoğunda Linux çalıştırmak isteyen mühendisler için Windows (o dönemlerde kurumsal model olarak kullanılmaktaydı), yalnızca belirli bir işletim sistemi ortamının yürütülmesini gerektiren durumlar için tasarlanmıştır. Bu süreç tamamlandığın zaman, Windows'u herhangi bir program gibi kapatmakta ve Linux ile devam etmekteydiler. Bu şekilde kullanıcının istemci işletim sistemini, sabit diskinde bulunan bir dosyadan (boyut olarak büyük olsa dahi) oluşan bir programmış gibi ele almasına izin vermenin ilginç bir etkisi olmuştur. Bu dosya, başka bir dosyaya yapılabileceği gibi manipüle edilebilir (yani, başka makinelere taşınmış veya kopyalanmış ve orada asıl kurulduğu makinede çalışıyormuş gibi çalıştırılmış olabilir). Daha da ilginci, işletim sistemi bilmeden durdurulabilir ve esasen askıya alınmış bir animasyon durumuna girmesine neden olabilir (Nadeau & Gray, 2013).

İşletim sistemi sanallaştırmasının (VMware, tarih yok) ortaya çıkmasıyla, Microsoft Windows Server (Minasi, et al., 2001) gibi tipik olarak tek ve özel bir işletim sistemi kullanan sunucular ve bu işletim sistemi için özel olarak tasarlanmış uygulamalar artık her yerde bulunan bir bilgi işlem ve depolama platformu olarak görülebilir. Bellek, bilgi işlem ve depolama alanındaki ileri gelişmeler ve artışlarla birlikte, veri merkezi bilgi işlem sunucuları, sanal bir ortamda çeşitli işletim sistemlerini eş zamanlı olarak yürütme yeteneğine sahiptir. VMware, tek ana bilgisayar versiyonunu, yüzlerce veya binlerce sanal makineyi tek bir konsoldan yürütme ve kontrol etme yeteneğine sahip, veri merkezi dostu bir ortama genişlemiştir. Daha önce tüm “çıplak metal” bir makineyi işgal eden Windows Server gibi işletim sistemleri şimdi her biri müşteri kullanıcısının istediği uygulamaları çalıştıran sanal makineler olarak yürütülüyordu. Kendi kendine yeten ortamında yürütülmesi tek farktı, duraklatılabilir, yer değiştirilebilir, klonlanabilir veya yedek olarak kopyalanabilmekteydi. Böylece elastik hesaplama çağı başlamıştır.

Operasyon departmanları, esnek hesaplama ortamında, sanal bir makineyi duraklatıp bir dosyayı kopyalayarak sunucuları herhangi bir fiziksel veri merkezi konumuna taşıyabilmektediler. Aynı dosyayı klonlayıp hipervizöre yeni bir örnek olarak çalıştırma komutunu vererek yeni sanal makinelere dönüştürebilmekteydiler. Bu esneklik, ağ operatörlerinin veri merkezi kaynak konumunu optimize etmeye başlamasını ve böylece güç ve soğutma gibi ölçütlere dayanarak kullanımı sağlanmıştır. Tüm aktif makineleri bir araya getirerek, bir operatör tüm bankaları veya fiziksel makine sıralarını uyutarak veya boşaltarak veri merkezinin başka bir yerinde soğutmayı azaltabilir, böylece bir veri merkezindeki soğutma yükünü optimize edebilir. Benzer şekilde, bir operatör coğrafi talebe göre bilgisayar, depolama veya ağ kaynaklarını taşıyabilir veya dinamik olarak genişletebilir (VMware, tarih yok).

Teknolojideki tüm ilerlemelerde olduğu gibi, bilgisayar, depolama ve ağ kaynaklarının operasyonel dağıtımında bu yeni keşfedilen esneklik, yalnızca hem depolama hem de hesaplama gücünün kullanımının en üst düzeye çıkarılması açısından operasyonel verimliliğin değil güç ve soğutma şartlarında yeni bir soruna yol açmıştır. Daha önce de belirtildiği gibi, şebeke operatörleri, bilgisayar gücü talebinin genel olarak zaman içinde arttığının farkına varmaya başlamıştır. Bu talebe yetişmek için, BT departmanları (genellikle yıllık bazda bütçe olan), gelecek yıl için ihtiyaç duyulacağını öngördükleri teçhizatı karar verecektir. Ancak, bu ekipman bir kez gelip raflara yerleştirildikten sonra henüz kullanılmamış olsa bile gücü, soğutmayı ve alan kaynaklarını tüketecektir! Bu ilk Amazon'da (Landrecht, et al., n.d.) keşfedilen ikilem oldu. O zamanlar, Amazon'un işi bir "hokey sopası" grafiğiyle büyüyordu; her altı ila dokuz ayda bir iki katına çıkıyordu. Sonuç olarak, büyümenin perakende sipariş, stok ve depo yönetim sistemlerinin yanı sıra dahili BT sistemlerine hizmet veren bilgisayar hizmetleri için talebin önünde kalması gerekiyordu. Sonuç olarak, Amazon'un BT departmanı önceden büyük miktarda depolama, ağ ve bilgi işlem kaynağı siparişi vermek zorunda kalmış, ancak bu kaynaklarla talep yetişene kadar boşta durması gerekmiştir. Amazon Web Servisleri (AWS), kullanılmayan kaynak havuzunu %100'e yakın bir oranda kullanabilmeleri için ticarileştirmenin bir yolu olarak icat edilmiştir. Dahili kaynaklar daha fazla kaynağa ihtiyaç duyulduğunda, AWS perakende kullanıcıları zorlamakta ve bu ihtiyaç olmadığı zamanlarda perakende işlem yapan kullanıcılar kullanılmayan kaynaklardan faydanabilmektedir.

Bu yönteme elastik hesaplama servisi denildiği gibi hiper sanallaştırma da denilmektedir.

Fiyat verimliliği için büyük miktarda depolama alan ve hesaplama yapan Amazon ve Rackspace (Kelly, tarih yok) gibi şirketler, tüm bilgisayar ve depolarını verimli bir şekilde kullanmadıklarını ancak o zaman fark ettiler ve yedek yatırım güçlerini ve depolarını, sermaye yatırımlarını telafi etmek amacıyla dış kullanıcılara yeniden satabildiler. Çok kullanıcılı bir veri merkezi böylece gelişmiştir. Ancak, kaynakları farklı fiziksel veri merkezlerinin sanal makinelerine keyfi bir şekilde yayılması gereken binlerce potansiyel kiracının nasıl ayrılacağıyla ilgili yeni bir sorun yaratmıştır.

Bu ikilemi anlamanın bir başka yoluda, hiper sanallaştırılmış ortamlara geçiş sırasında yürütme ortamlarının genellikle tek bir işletme veya kuruluş tarafından yönetildiğine dikkat etmektir. Yani, tipik olarak tüm bilgisayar ve depolama alanlarını (kiralanmış ortak konum alanlarına rağmen), çok sayıda sanal veya fiziksel makineyi ve ağa bağlı depolamayı birbirine bağlayan tek, düz bir yerel alan ağı (LAN) gibi kullanılmakta ve işletilmektedir. (Düzenleme şartlarının ayrılmayı zorunlu kıldığı finansal kurumlar bu işlemden istisnadırlar.) Bununla birlikte, bu vakalardaki bölümlerin sayısı göreceli olarak küçük (100'den az) olduğundan katman 2 veya katman 3 MPLS (Defteri, 2013) VPN'ler gibi mevcut araçlar kullanılarak kolayca çözülmüştür. Her iki durumda da, tüm bilgisayar ve depolama kaynaklarını bu noktaya kadar bağlayan ağ bileşenleri oldukça basittiler; genel olarak tüm fiziksel ve sanal makineleri birbirine bağlayan düz bir Ethernet LAN idi. Bu ortamların çoğu, ağdaki tüm cihazlara (sanal veya fiziksel) tek bir ağdan (IP alt ağları ile de olabilmektedir) IP adresleri atamakta, tek bir kuruluş olarak makinelere sahip olmakta ve bunlara erişim sağlaması gerekmekteydi. Bu aynı zamanda, genel olarak o işletmede bulunan farklı veri merkezleri arasında sanal makineleri hareket ettirmenin bir sorun olmadığı anlamına geliyordu, çünkü yine de hepsi aynı yönlendirilmiş alan içinde olup fiziksel konumlarına bakmaksızın birbirlerine ulaşabildiler.

Çok kullanıcılı bir veri merkezinde (Anon., 2020), bilgi işlem, depolama ve ağ kaynakları birbirinden bağımsız veya birbirinden izole edilmiş dilimler halinde sunulabilir. Aslında, ayrı tutulmaları kritik bir durumdur. Bu, geçmişin tek geçici veri merkezi ortamında bulunmayan bazı ilginç zorluklar ortaya koydu. Ortamlarının, bu

işletim sistemlerinin üstünde herhangi bir sayıda işletim sisteminin ve uygulamasının yürütülmesine izin verdiğini, ancak sahibinin veya müşteri gibi diğer harici kullanıcıların erişebilmesi için her birinin benzersiz bir ağ adresine ihtiyaç duyduğunu unutulmamalıdır. Geçmişte, adresler, muhtemelen özel adreslerin bulunduğu tek bir iç bloktan atanabilmekte ve dahili olarak kolayca yönlendirilebilmekteydiler. Ancak günümüzde, harici olarak yönlendirilebilir ve erişilebilir benzersiz adresler atanması gerekmektedir. Ayrıca, söz konusu her sanal makinenin de benzersiz bir katman 2 adresi olduğunu göz önünde bulundurulursa bir yönlendirici bir paket teslim ettiğinde, nihayetinde Ethernet (sadece IP değil) kullanarak bir paket vermesi gerekir. Sanal makine hareketliliği (VM mobilitesi) (Foskett & Rat, 2012) dikkate alınana kadar bu genellikle bir sorun değildir. Bu durumlarda, sanal makineler güç, soğutma veya bilgi işlem sıkıştırma nedenlerinden dolayı yeniden konumlandırılmıştır. Burada asıl sorun şu ki fiziksel yer değiştirme fiziksel adres yer değiştirme anlamına gelir. Ayrıca, bu makine için önceden hedeflenen paketlerin orijinal konumlarında şimdi yeni konumlarına değiştirilebilmelerini sağlamak için katman 3 yönlendirmesinde değişiklik yapılması anlamına da gelir.

Aynı zamanda, veri merkezleri de gelişirken ağ donanımları, besleme ve hızların ötesindeki yenilikler açısından hala durmuş gibi görünmekteydi. Yani, anahtar yapısının kapasitelerindeki ve arayüz hızlarındaki sürekli artışın ötesinde, IP, MPLS ve mobil teknolojilerin ortaya çıkışından bu yana veri iletişimi fazla gelişmemiştir. IP ve MPLS, bir ağ operatörünün, veri merkezi operatörlerinin bilgisayar sanallaştırmasının ortaya çıkmasıyla fiziksel olanları üzerinden çalıştırmak için sanal makineler oluşturabilecekleri şekilde ağ tabanlı ve sanal ağ kaplamaları oluşturmalarına izin vermekteydi. Ağ sanallaştırması genellikle sanal özel ağlar (VPN) (Zhao, et al., 2008) olarak anılır ve noktadan noktaya dahil olmak üzeri (örneğin, dizüstü bilgisayarınızda çalıştırabileceğiniz ve kurumsal ağınıza bağlanabileceğiniz kişisel bir VPN) çeşitli türlerde ortaya çıkar; katman 3 (bir ağ operatörünün, trafiğini başka bir kuruluştan izole edecek şekilde güvenli bir şekilde barındırmasına izin verilmesi gibi durumlarda bir IP veya yönlendirilmiş ağın sanallaştırılması); ve katman 2 VPN'ler (kullanılan adreslerin Ethernet olması dışında, katman 3 VPN'e benzer şekilde yalıtılan anahtarlı ağ sanallaştırması).

Ticari yönlendiriciler ve anahtarlar tipik olarak bir şebeke operatörünün bu cihazları yapılandırmasına ve yönetmesine izin veren yönetim arayüzleriyle

(TechLibrary, 2018) birlikte gelir. Bazı yönetim arayüzleri örnekleri arasında komut satırı arayüzleri, XML/Netconf, grafiksel kullanıcı arayüzleri (GUI'ler) ve Basit Ağ Yönetim Protokolü (SNMP) (Erlang, 2019) bulunur. Bu seçenekler, bir operatörün bir cihazın özelliklerine uygun bir şekilde erişmesini sağlayan bir arayüz sağlar, ancak yine de operatörden en düşük seviyede ayrıntıları gizler. Örneğin, şebeke operatörleri statik rotaları veya diğer statik yönlendirme girişlerini programlayabilir, ancak bunlar sonuçta cihazın işletim sisteminden geçen isteklerdir. Bu genellikle, bir aygıtta var olan sözdizimi veya işlevsellik anlambilimini kullanarak programlama yapmaya çalışana kadar bir problem değildir. Birisi yeni bir yönlendirme protokolü denemek isterse, bu yazılımı desteklemek için firmware yazılı olmadığı bir cihazda olamaz. Bu gibi durumlarda, müşterinin bir cihaz satıcısına özellik geliştirme isteği yapması ve daha sonra genellikle bir süre beklemesi yaygın bir şekilde uygulanmaktaydı. (Birkaç yıl beklemek normal karşılanmaktaydı).

Aynı zamanda dağıtılmış (en azından mantıksal olarak) kontrol plan (Bhowmik, et al., 2015)kavramı tekrar ortaya çıkmıştır. Bir ağ cihazı, genellikle bir cihazdaki çeşitli ağ portlarını bağlayan bir anahtar yapısı ve bir cihazın beyni olan bir kontrol düzleminden oluşur. Örneğin, bir ağ içinde döngü içermeyen yollar oluşturmak için kullanılan yönlendirme protokolleri en yaygın olarak dağıtılmış bir şekilde uygulanır. Diğer bir deyişle, ağdaki her cihaz protokolü uygulayan bir kontrol düzlemine sahiptir. Bunlar ağ yolu yapımını koordine etmek için birbirleriyle iletişim kurarlar. Bununla birlikte, merkezi bir kontrol düzlemi paradigmasında, bir kontrol düzlemi (veya en azından mantıklı) mevcut olacaktır. Bu über beyin her cihaza komutları dayatır, böylece fiziksel anahtarlama ve yönlendirme donanımını manipüle etme komutu göndermektedir. Aygıtların veri düzlemlerini uygulayan donanımın oldukça özel ve bu nedenle pahalı olmasına rağmen, kontrol düzleminin, Intel tarafından üretilen merkezi işlem birimleri gibi daha az pahalı, genel amaçlı bir bilgi işlemeye yönelmeye devam ettiğini not etmek önemlidir.

Bu söz konusu kavramların hepsi, günümüzde yazılım olarak tanımlanmış ağ iletişimi (SDN) (Contini, 2016) olarak adlandırılan şey için motivasyon çekirdeğini yarattıkları için önemlidir. SDN'in ilk destekçileri, özellikle cihaz geliştirme ve yenilik alanlarında ağ cihazı satıcılarının ihtiyaçlarını karşılamadığını gördü. Cihazlarının en azından kontrol düzlemi bileşenleri için üst seviye yönlendirme ve anahtarlama teçhizatının çok fazla pahalı olduğu görüldü. O zaman, bu işleme gücünün, mantıksal

olarak merkezi bir kontrol düzlemi çalıştırmak ve potansiyel olarak ucuz, emtia fiyatlandırmalı anahtarlama donanımı kullanmak için kullanılabileceğinin farkına varıldı. Stanford Üniversitesi'nden birkaç mühendis, böyle bir yapılandırmada uygulanabilecek OpenFlow adlı bir protokol ortaya çıkardılar. OpenFlow, bu ağ için tek kontrol düzlemini barındıran (mantıksal) merkezi bir denetleyiciden kendilerine gönderilen komutlara yanıt vermek için yalnızca veri düzlemi içeren birkaç cihaz için tasarlanmıştı. Tüm ağ yollarının korunmasından ve kontrol ettiği ağ cihazlarının her birinin programlanmasından denetmen sorumlu olacaktır. Komutlar ve bu komutlara verilen yanıtlar OpenFlow protokolünde açıklanmıştır. Açık Ağ Vakfı'nın (ONF) (Anon., 2019) SDN çabalarını ticari olarak desteklediğini ve günümüzde merkezi standardizasyon otoritesini ve pazarlama organizasyonunu sürdürdüğünü belirtmekte fayda var. Tarif edilen bu temel mimariye dayanarak, meta fiyatlandırılmış donanıma ilişkin bir veri merkezi içinde uygulanarak yeni bir ağ protokolü oluşturmanın artık ne kadar hızlı ve pratik olduğu tahmin edilebilir. Daha da iyisi, sanal bir makinede elastik bir bilgi işlem ortamında uygulanabilir.

SDN ile ilgili biraz farklı bir görüş, sektörde bazılarının yazılım tanımlı ağların aksine, yazılım odaklı ağlar olarak adlandırdığı şeydir. Yazılım odaklı yaklaşımda, biri OpenFlow ve bu mimariyi mümkün olan ayrı bir işlevsellik alt kümesi olarak görmektedir. Ağı, beyinsiz ağ cihazlarıyla mantıksal olarak merkezi kontrol düzleminden oluşan bir ağ olarak görmek yerine, dünyayı eski ve yeninin bir melezi olarak görür. ONF ve yazılım tanımlı ağlar tarafından (Kesden, 2014) önerilen yeni bir dünyaya yol açmak için mevcut ağların toptan olarak dağıtılacağını düşünmek gerçekçi değildir. Ayrıca, bugün var olan ve Internet gibi şeylerden sorumlu olan ağ teknolojisindeki tüm ilerlemelerin atılması da gerçekçi değildir. Bunun yerine, ağların bir kısmının mantıksal olarak merkezi bir kontrol cihazı tarafından çalıştırıldığı, diğer kısımların daha geleneksel dağınık kontrol düzlemi tarafından çalıştırılacağı hibrit bir yaklaşım daha muhtemeldir. Bu aynı zamanda, bu iki dünyanın birbiriyle etkileşime girmesi gerektiği anlamına gelecektir.

SDN ve OpenFlow (GERRITY & HU, 2014) destekçilerinin elde etmeye çalıştığı şeylerin en az bir kısmının daha büyük ve daha esnek bir ağ cihazı programlanabilirliği olduğunu gözlemlemek ilginçtir. Bunun mutlaka ağ kontrolü ve veri düzlemi konumu ile ilgisi olması gerekmemektedir; ancak, nasıl programlandıkları ile ilgilidir. SDN ve OpenFlow oluşturma motivasyonlarından

birinin, yalnızca programlandığı yer değil, bir ağ cihazını nasıl programlayabileceğinin esnekliği olduğu unutulmamalıdır. Tarif edilen SDN mimarisinde neler olup bittiği gözlemlenirse, bu soruların ikisi de çözülür. Programlanabilirliğin en uygun seçenek olup olmadığı soru sorgulanabilir.

Bunu ele almak için, Juniper, Cisco, Level3 ve diğer satıcı ve servis sağlayıcılarını temsil eden kişiler yakın zamanda Yönlendirme Sistemine Arayüz (I2RS) (White, et al., 2019) adı verilen ağ programlanabilirliği konusunda çaba sarf ettiler. Bu kaynaklardan bazı kişiler, Alia Atlas, David Ward ve Tom'un birincil katkı sağlayan birincil gereksinimleri ve çerçeve taslakları dahil olmak üzere birçok IETF taslaklarına katkıda bulunmuştur. Yakın gelecekte, bu konunun etrafında en az bir düzine taslak çevrimiçi görünmelidir. Açıkçası, bu çabaya büyük ilgi var. I2RS ile ilgili temel fikir, bir ağ cihazının yönlendirme bilgi tabanını (RIB) (Bahadur, et al., 2018) programlamanın bir yol olarak gerçekleştirilmesi için provizyon işlemlerinin hızlı bir şekilde kesilmesine izin veren hızlı bir yol protokolü kullanarak programlama aracı olarak hareket edecek bir protokol ve bileşenlerinin -RIB ve onu kontrol eden RIB yöneticisi ile zaman etkileşimi oluşturmaktır. Önceden, RIB'ye tek erişim cihazın konfigürasyon sistemi üzerinden yapılmaktaydı (Juniper'in durumunda, Netconf veya SNMP).

I2RS'yi (White, et al., 2019) anlamanın anahtarı, kesinlikle başka bir sağlama protokolü değil; ağ elemanları, ağ programlaması, durum ve istatistiksel toplama ve işlem sonrası analitik arasındaki geri besleme döngüsünü hızlandırma sorununa tam bir çözüm içeren çok sayıda başka temel kavramlar olmasıdır. Bugün, bu döngü acı verici bir şekilde yavaştır. I2RS'ye dahil olanlar, programlanabilir ağların geleceğinin anahtarının bu döngüyü optimize etmenin geçtiğine inanmaktadır.

Bu amaçla, I2RS, ağ yollarının, ilkelerinin ve bağlantı noktası konfigürasyonunun programlanabilirliği bakımından çeşitli seviyelerde soyutlama sağlar, ancak her durumda, söz konusu programlamanın yetişkinlere yönelik denetimini, bunları yerine getirmeden önce komutları kontrol etme aracı olarak sağlama avantajına sahiptir. Örneğin, bugün ağın verimliliği için çok ayrıntılı ya da ayrıntılı olan donanım soyutlama katmanında (HAL) (TAŞDEMİR, 2015) programlama için bazı protokoller mevcuttur ve aslında operasyonel sistemler üzerinde aşırı yük vardır. Başka bir örnek, değişikliklerin hızlı bir şekilde

programlanması ve ardından sonuçlara tanıklık etmek amacıyla, yalnızca ağın davranışını optimize etmek için hızlı bir şekilde yeniden programlayabilmek için RIB'ye hızlı ve optimum erişim sağlamak için operasyonel destek sistemleri (OSS) (Hanrahan, 2007) uygulamalarının sağlanmasıdır. Tüm bu örneklerin etrafındaki en önemli hususlardan biri, başvurular ile RIB arasındaki söylemin RIB yöneticisi aracılığıyla gerçekleşmesidir. Bu çok önemlidir, çünkü birçok operatör operasyonel ve iş akışı yatırımlarını, Junos veya IOS-XR gibi cihaz işletim sistemlerinde var olan yönlendirme protokolü zekasında muhafaza ederken, ağlarında ilave optimizasyon seviyelerine izin vermek için bu yeni ve kullanışlı programlanabilirlik paradigmasından yararlanırlar.

I2RS (White, et al., 2019) ayrıca, rotalama ve yol kararlarını ve programlanabilirliğini mantıksal olarak merkezileştirme arzusuna kendini vermektedir. Protokol, bir cihazda veya bir cihazın dışında çalışmak için gereksinimlere sahiptir. Bu şekilde, dağıtılmış denetleyici işlevi, istendiği durumlarda benimsenmiştir; Bununla birlikte, daha klasik dağınık kontrolün istendiği durumlarda, bunları da destekleyebilmektedir.

Son olarak, I2RS'nin (White, et al., 2019) bir başka önemli alt bileşeni normalleştirilmiş ve soyutlanmış topolojidir. Ortak ve genişletilebilir bir nesne modeli tanımlamak bu topolojiyi temsil edecektir. Bu hizmet ayrıca, çoklu topolojik gösterimlerin soyutlanmasına da izin verir. Bu modelin temel bir özelliği, yönlendirici olmayanların (ve yönlendirme protokolü hoparlörlerinin) ilerideki RIB durumunu daha kolay manipüle edip değiştirebilmeleridir. Günümüzde, yönlendirici olmayanlar bu bilgilere en iyi şekilde ulaşmakta büyük zorluk çekmektedir. İleride, bir ağ yönetiminin / OSS'nin bileşenleri, analitik ya da henüz öngöremediğimiz diğer uygulamalar, yönlendirme durumu ve ağ topolojisi ile hızlı ve verimli bir şekilde etkileşime girebilecektir.

Bu yüzden, bu düşünceleri sonuçlandırmak için, ne olduğunu ve ne olacağını düşündüğümüz için SDN tanımlamamız uygun olur:

B. Motivasyon

Geleneksel ağlarda içerisinde kontrol mekanizması barındıran bir yönlendirici bulunur. Veri düzlemi bu kontrol mekanizmasından alınan emirleri yerine getirir. Aynı

zamanda yönlendiricide ayarlamaların yapıldığı bir yönetim düzlemi bulunur. Komşu yönlendiriciden bir paket geldiği zaman kaynak ve hedef bilgisine bakılır ve yönlendirme yazılımı ile yönlendirme tablosuna göre gönderilecek bir sonraki verinin ne olacağına karar verilir. Yönlendirme yazılımı en kısa yola öncelik veren algoritmalar olabileceği gibi farklı metriklere dayalı başka yönlendirme algoritmaları da olabilir. Kontrol düzlemi ağın işletim sistemi gibi davranarak yapılacak işlemlerle ilgili kararlar veren yazılım kısmıdır. Bu yazılımın aldığı karar doğrultusunda da veri düzleminde paket, cihaz (donanım) aracılığıyla ileriye aktarılır. Anahtar'ın içinde yazılım ile donanım bir arada bulunmaktadır. Bunlar iki farklı yapı olmasına rağmen aynı yerde duran tek varlık gibi davranmaktadırlar. Ne yazık ki geleneksel ağlarda sorun burada ortaya çıkmaktadır. Çünkü Veri düzlemi ve kontrol düzlemi aynı ağ cihazında bulunur. Geleneksel ağlarda olan bu sorunun çözümü merakla arandığı için bu tez yazılmaktadır.

C. Araştırma Sorusu

Bu yüksek lisans tezinin araştırma soruları aşağıda verilmiştir:

- Yazılım Tanımlı Ağ nedir?
- Bir Yazılım Tanımlı Ağ nasıl tasarlanır?

D. Tezin Yapısı

Tezin ikinci bölümü yazılım tanımlı ağlar hakkında detaylı bilgi vermekte ve üçüncü bölüm OpenFlow anahtarı hakkında bilgi vermeyi amaçlamaktadır. Dördüncü bölüm, bir yazılım tanımlı ağın nasıl tasarlandığını göstermektedir. Beşinci bölüm elde edilen bulgular hakkında bilgi vermektedir. Son bölümde (Altıncı Bölüm) elde edilen bulgular analiz edilmektedir.

E. Yazılım Tanımlı Ağlar (SDN)

Gerçek veya sanallaştırılmış olmasına rağmen, uygulamalar ve ağ hizmetleri ile aygıtları arasındaki etkileşimi daha yakından bağlayarak (yani, sağlama, mesajlaşma ve alarm verme) ağ işlemlerini optimize eden ve basitleştiren mimari bir yaklaşımdır. Genellikle, mantıksal olarak merkezi bir ağ kontrolü noktası kullanılarak elde edilir- bu genellikle bir SDN denetleyicisi olarak gerçekleştirilir - bu daha sonra ağ

elemanları ile etkileşime girmek isteyen uygulamalar ile bu uygulamalara bilgi iletmek isteyen ağ elemanları arasında iletişimi düzenler, yönlendirir ve kolaylaştırır. Kontrolör daha sonra modern, uygulama dostu ve çift yönlü programatik ara yüzler aracılığıyla ağ fonksiyonlarını ve operasyonlarını açığa çıkarır ve özetler (Nadeau & Gray, 2013).

Gördüğünüz gibi, yazılım tanımlı, yazılım odaklı ve programlanabilir ağlar, zengin ve karmaşık bir dizi tarihsel soy, zorluklar ve bu sorunlara çeşitli çözümler ile gelir. Bu, mümkün olan şeylere dayalı teknolojiyi ilerleten, yazılım tanımlı, yazılım odaklı ve programlanabilir ağlardan önce gelen teknolojilerin başarısıdır. Meselenin gerçeği, internet dahil olmak üzere, dünyadaki ağların çoğunun IP, BGP (TUTAR, 2016), MPLS ve Ethernet temelinde çalışmasıdır. Günümüzde sanallaştırma teknolojisi, VMware'in yıllar önce başlattığı teknolojilere dayanmaktadır ve bunun ve diğer ürünlerin dayandığı temel olmaya devam etmektedir. Ağa bağlı depolama, benzer şekilde zengin bir tarihe sahiptir.

I2RS, ağ, hesaplama ve depolama sanallaştırmasının yanı sıra bu hiper sanallaştırılmış ortamlarda yürütülen uygulamaların programlanabilirliği, erişilebilirliği, konumu ve yer değiştirmesinin sorunlarını çözme konusunda da benzer bir geleceğe sahiptir.

2013 yılında SDN kontrolörleri basını yönetmeye devam etseler de, o sırada birçok başka gelişme oldu. Çok ilginç ve parlak olanlardan biri OpenDaylight (Anon., 2018) projesidir. OpenDaylight'ın misyonu, ortak ve sağlam bir yazılım tanımlı ağ platformunu hızlandırmak ve ilerletmek için kod ve mimariyi içeren topluluk ve endüstri destekli bir açık kaynak çerçevesine olanak sağlamaktır. Bu amaçla, Open Daylight, Linux Vakfı'nın şemsiyesi altında barındırılmakta ve SDN kontrolörleri etrafında gerçek bir oyun değiştirme ve potansiyel olarak saha seviyelendirme çabasına olanak sağlamaktadır. Bu çaba, bu alanda en önemli olduğunu düşündüğümüz yeniliği olan uygulamaları da teşvik edecektir. Denetim birimlerinde son birkaç yılda birçok gelişme görmemize rağmen, Denetim birimleri SDN destekli uygulamalar için temel altyapıyı gerçekten temsil etmektedir. Bu bağlamda, endüstri son birkaç yıl içinde denetleyicileri tasarlamak ve geliştirmek için mücadele ederken çoğunlukla uygulamaları görmezden gelmiştir. SDN'in nihayetinde gerçekten operasyonel optimizasyon ve verimlilik ile ilgili olduğunu düşünülmekte ve bunu

başarmanın en iyi yolu bu altyapının hızlı bir şekilde kontrol edilmesi ve endüstrinin SDN mimarisinin uygulama ve cihaz katmanlarında yeniliklere odaklanmasına izin verilmesidir.

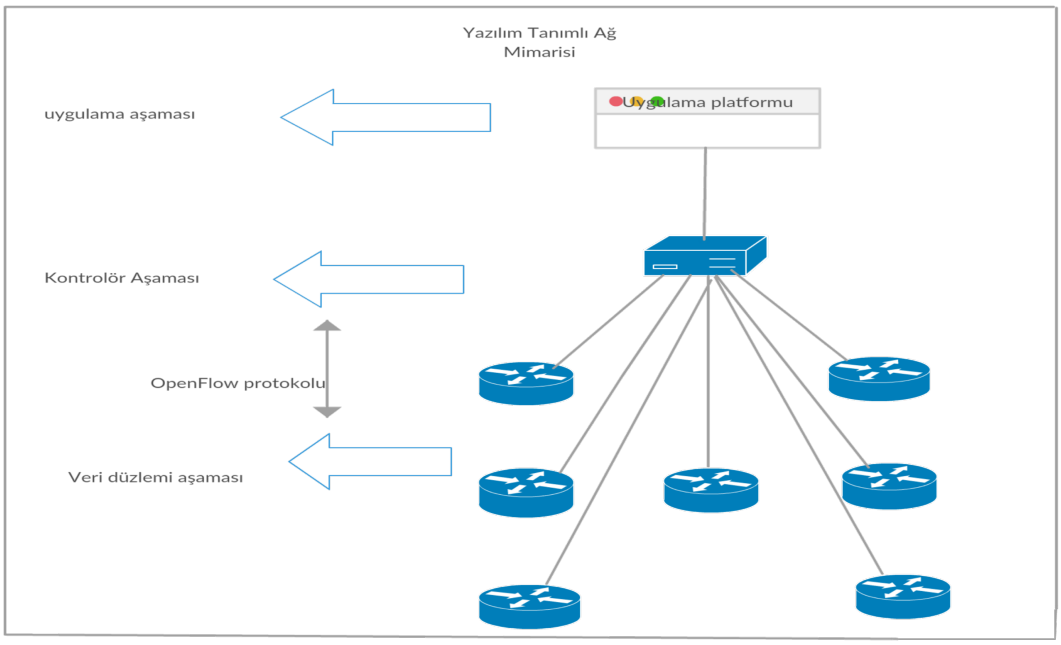
II. YAZILIM TANIMLI AĞLAR

A. SDN Mimarisi

Bir sonraki bölümde, SDN mimarisinin ana bileşenlerini açıklanmaktadır. Temel olarak SDN mimarisi, kontrol düzlemini veri düzleminden ayıran, ağ gelişimini, birlikte çalışabilirliği ve ölçeklenebilirliği kolaylaştıran yeni bir ağ oluşturma paradigmasını temsil eder. Bu ayrıştırma SDN "çekirdek" olan, anahtar bileşenleri ayrımı yoluyla mümkündür. Ayrıca, "klasik" ağ ile yeni ağ kurma paradigması arasındaki temel farklar, aşağıda ayrıntıları verilen üç farklı mantıksal katmanla tanımlanmaktadır (Al-Ani, 2015).

1. Mantıksal Katmanlar

SDN mimarisi, şekil 2.1'de gösterildiği gibi üç farklı mantıksal katmanla temsil edilebilir.



Şekil 1: SDN Mimarisi

Her katman özellikle farklı işlevlere sahiptir:

- Veri planı katmanı, fiziksel cihazlardan (ör. Anahtarlar ve yönlendiriciler) oluşan ağ altyapısını gösterir.
- Denetleyici katmanı "ağ zekasını" temsil eder ve bu katmanda bulunan SDN denetleyicisinde mantıksal olarak merkezileştirilir. Bu çözüm, denetleyicinin altyapı katmanına yerleştirilen ağın genel görünümünü korumasını sağlar.
- Uygulama katmanı, ağ operatörlerinin ve yöneticilerin çalışabileceği katmanı temsil eder. Ağ durumunu denetleyici katmanında merkezleyerek, uygulama katmanında, ağ kaynaklarını dinamik, otomatik SDN programları kullanarak yapılandırmak, yönetmek, güvenli hale getirmek ve optimize etmek mümkündür. Ayrıca, ağ operatörleri, uzun zaman alabilecek satıcı sürümlerini beklemeden, özelleştirilmiş programları doğrudan yazabilir ve yayabilir.

Bu nedenle, yukarıda açıklanan katmanlar soyutlaması, programcıların Uygulama Programlama Arabirimleri (API) aracılığıyla binlerce farklı fiziksel aygıt yerine bir ağ soyutlama katmanı üzerinde çalışmasını sağlar. Ancak, bu soyutlama alt katman altyapısı kendisiyle etkileşimi mümkün kılarsa mümkündür (ONGARO, 2013-2014).

B. Veri Düzlemi

Veri düzlemi, gelen datagramları (Admin, 2019) (tel, fiber veya kablosuz medyada), datagramı toplayan ve temel doğruluk/uygunluk testi gerçekleştiren bir dizi bağlantı düzeyinde işlem yoluyla işler. İyi düzenlenmiş (yani doğru) bir datagram, kontrol düzlemi tarafından daha önce programlanmış FIB tablosunda (veya tablolarda, bazı uygulamalarda) aramalar gerçekleştirilerek veri düzleminde işlenir. Buna bazen paket işleme için hızlı yol denir, çünkü önceden programlanmış FIB kullanarak paketin hedefini tanımlamak dışında başka bir sorgulamaya gerek yoktur. Bu işlemin istisnası, paketlerin bilinmeyen bir hedef tespit edildiğinde olduğu gibi bu kurallarla eşleştirilemediği ve bu paketlerin kontrol düzleminin RIB kullanarak onları daha fazla işleyebildiği rota işlemcisine gönderildiği durumdur. FIB tablolarının bir dizi iletme hedefinde - yazılım, donanım hızlandırmalı yazılım (Intel veya ARM tarafından örneklendiği gibi GPU / CPU), emtia silikonu içerebileceğini (Ethernet anahtar pazarında Broadcom, Intel veya Marvell tarafından örneklendiği gibi NPU), FPGA ve özel silikon (Juniper Trio gibi ASIC'ler) veya ağ elemanı tasarımına bağlı olarak herhangi bir birleşimini anlamak önemlidir.

Bu göstergedeki yazılım yolu, işlemci belleği görünüşte sınırsız tablo depolama için bir işlemci yoğun araması (Bunun çekirdekte mi yoksa kullanıcı alanında mı olduğu, ana bilgisayar işletim sisteminin özellikleri ve altyapısına bağlı satıcıya özgü bir tasarım kararıdır) gerçekleştiren modern özel ağ elemanının (örneğin, yönlendirici veya anahtar) CPU-yönlendirmeli iletimi ile örneklenmiştir (Salisbury, 2012).

Modern bilgi işlem ortamının hiper yönetici temelli anahtarı veya köprüsü karşılığı, donanım yönlendirme modellerinin optimizasyonlarının çoğuna (ve bazı sınırlamalarına) sahiptir. Tarihsel olarak, donanım tablolarındaki aramaların çok daha yüksek paket yönlendirme performansı ile sonuçlandığı kanıtlanmıştır ve bu nedenle özellikle daha yüksek bant genişlikli ağ elemanları için hâkim ağ elemanı tasarımları vardır. Ancak, jenerik işlemcilerin G / Ç (Stallings, 2016,2013,2010) işlemlerinde son gelişmeler, bulut bilişimde büyüme ve yenilikçiliğin teşvik ettiği, amaca yönelik, özellikle orta-düşük performans aralıklarında, para için tasarımlar yapılmaktadır.

Donanım iletme tasarımlarındaki farklılıklar (tahta ve raf) alan, bütçe, güç kullanımı ve verim (Rivenes, 2016) hedefi gereksinimleri gibi çeşitli faktörlere yayılmıştır. Bunlar, belirli bir hedef paket büyüklüğü (veya harmanlama) için hat hızında (yani, bir arayüz için maksimum sinyalli veya teorik verime yakın) iletmeyi sürdürmek için bellek türünün (hız, genişlik, boyut ve konum) yanı sıra işlem bütçesinde (paket üzerinde gerçekleştirilen işlemlerin sayısı, sırası veya türü) farklılıklara yol açabilir. Sonuç olarak, tasarım özellikleri arasında yönlendirme özelliği desteği ve yönlendirme ölçeğinde (örneğin, yönlendirme giriş sayısı, tablo sayısı) farklılıklara neden olur.

Veri düzlemi iletme aramasından kaynaklanan tipik eylemler ileri (ve çok noktaya yayın, çoğaltma gibi özel durumlarda), bırakma, yeniden işaretleme, sayma ve sıra'dan ibarettir. Bu işlemlerin bazıları birleştirilebilir veya birlikte zincirlenebilir. Bazı durumlarda, ileriye dönük karar, OSPF (OSPF Explained | Step by Step, 2018) veya BGP (Demicoli, 2016) gibi yerel olarak çalışan bir işlem için trafiğin hedef olduğunu belirten yerel bir port döndürür. Bu datagramlar, donanım iletme yolundan ayrıldıkları ve bir iç iletişim kanalı kullanarak rota işlemcisine iletildikleri punt yolu olarak adlandırılanı alırlar. Bu yol genellikle, normal trafiğin yüksek verimli paket iletimi için tasarlanmadığından, nispeten düşük verimli bir yoldur; bununla birlikte,

bazı tasarımlar, bu amaç için iç anahtarlama sistemi/yapısına kutu içinde yakın oranlı iletime neden olabilen ilave bir yol ekler.

Yönlendirme kararına (Casado, et al., 2008) ek olarak, veri düzlemi genellikle yönlendirme özellikleri olarak adlandırılan bazı küçük hizmetler / özellikler uygulayabilir (Erişim Kontrol Listeleri ve QoS / ilke ile örneklenmiştir). Bazı sistemlerde, bu özellikler kendi ayrık tablolarını kullanır, diğerleri ise ileri yönlendirme tablolarına genişletme işlemi yapar (giriş genişliğini arttırır).

Ek olarak, farklı tasarımlar farklı özellikleri ve iletme işlem sırasını uygulayabilir (tablo 1). Bazı siparişler, bazı özellik işlemlerini diğerleri dışında özel yapabilir.

Bu özelliklerle, (küçük bir dereceye kadar) yerel olarak yönlendirme aramasının sonucunu değiştirebilir veya engelleyebilirsiniz. Örneğin:

- Bir erişim kontrol listesi (Bakanlığı, 2008) girişi, belirli bir eşleşen akış için bir bırakma işlemi
  (ACL'de, ileriye dönük kararda daha geniş bir parametre grubunun olabilir) belirtebilir. Yokluğunda, meşru bir yönlendirme girişi olmuş olabilir ve bu nedenle paket düşürülmeyecektir.
- Bir QoS (Manzanares, et al., 2018) poliçesinin en nihayetinde çıkış ağındaki bir kuyruğa giden akışı haritalandırabilir veya TOS / COS'u ağdaki poliçe hizmetini normalleştirmek için belirtebilir. Hedef / akış için mevcut yönlendirme girişinden bağımsız olarak paketin düşürülmesini (şekillendirilmesini) ACL gibi işaretleyebilir.

Çizelge 1::Geleneksel bir yönlendirici/anahtar üzerindeki bir giriş özelliği uygulamasının genel bir örneği

Muhtemelen, bu hizmetlerin bir veri düzlemi ve kontrol düzlemi bileşeni var ve bunların veri yönetimi başlığının oturum yönetimi, proxy ve büyük ölçekli dönüşümlerini tartışmaya başladığımızda tanımları temiz bir şekilde birbirinden ayrılır gibi görünmektedir. İletme işleminin bir parçası olarak, veri düzlemi elemanlarının bir miktar datagram üstbilgisi yeniden yazma işlemi yapması gerekir.

C. Kontrol Düzlemi

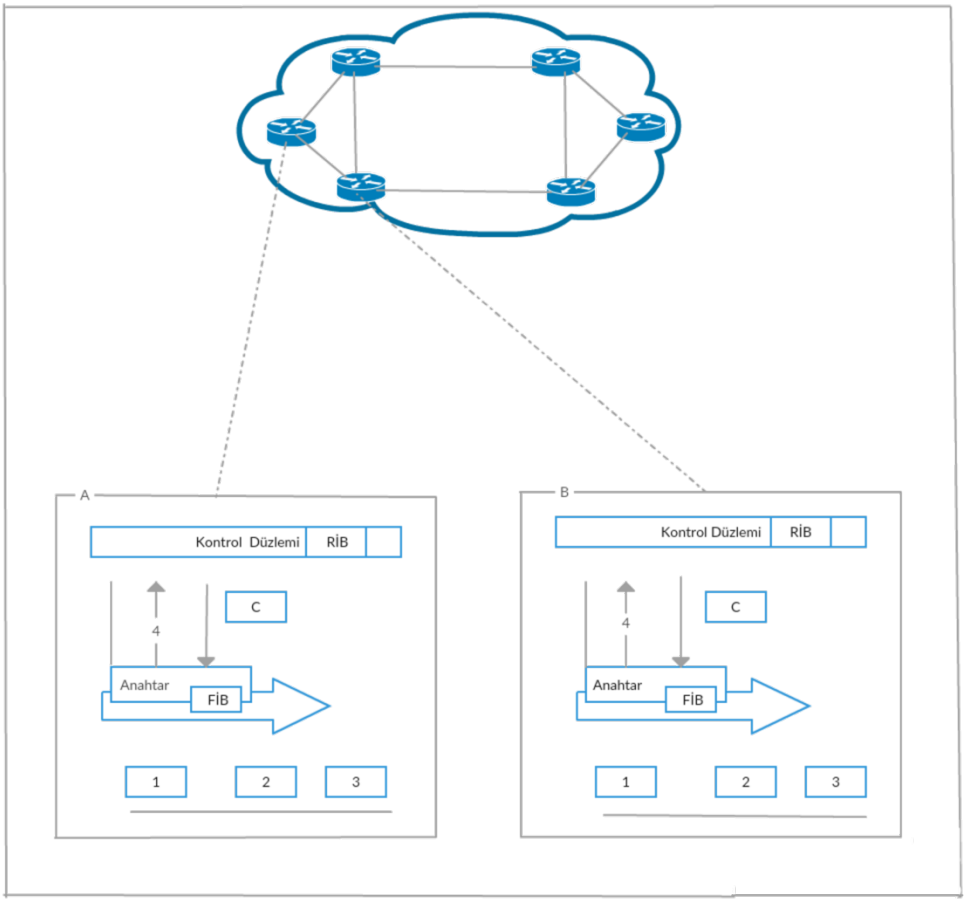
Kontrol düzlemi, çok yüksek bir seviyede, bir cihazdaki giriş ve çıkış portları arasındaki trafiği iletmek için kullanılan, veri düzlemi tarafından kullanılan yönlendirme tablosu girişlerini oluşturmak için kullanılan yerel veri setini oluşturur. Ağ topolojisini depolamak için kullanılan veri setine yönlendirme bilgi tabanı (RIB) denir. RIB, genellikle ağdaki diğer kontrol düzlemi oluşumları arasındaki bilgi alışverişi sırasında tutarlı (yani, döngüsüz) tutulur. Yönlendirme tablosu girişleri genellikle iletim bilgi tabanı (FIB) (Darren, 2011) olarak adlandırılır ve genellikle tipik bir cihazın kontrol ve veri düzlemleri arasında yansıtılır. FIB, tutarlı ve kararlı olarak kabul edildiğinde programlanır. Bu görevi gerçekleştirmek için kontrol kuruluşu/programı, belirli kısıtlamaları sağlayan ağ topolojisinin bir görünümünü

geliştirmelidir. Ağın bu görünümü, manuel olarak programlanabilir, gözlem yoluyla öğrenilebilir veya bir veya daha fazla yönlendirme protokolü, manuel programlama veya bir kombinasyonunun kullanılması yoluyla olabilecek diğer kontrol düzlemlerinin örnekleri ile söylem yoluyla toplanan bilgi parçalarından her ikisi de oluşturulabilir.

Birbirine bağlı anahtarlar ağını temsil eden kontrol ve veri düzlemlerinin mekaniği Şekil 2'de gösterilmektedir. Şeklin en üstünde, kontrol ayrıntılarının ve bu anahtarların ikisinin veri düzlemlerinin (A ve B olarak not edilmiştir) genişlemesi ile bir anahtar ağı gösterilmektedir. Şekilde, paketler en soldaki kontrol düzleminde A şalteri ile alınmakta ve en sonunda şeklin sağ tarafında B şalterine iletilmektedir. Her genişlemenin içerisinde, kontrol ve veri düzlemlerinin, işlem düzlemi kendi işlemcisinde/kartında ve veri düzleminde ayrı bir yapı üzerinde çalıştığı şekilde ayrıldığı dikkate alınmalıdır. Her ikisi de tek bir şasi içerisinde bulunur. Şekilde gösterildiği gibi, paketler, veri düzleminin bulunduğu hat kartının giriş portlarına alınır. Örneğin, bilinmeyen bir MAC adresinden gelen bir paket alınırsa, öğrenildiği, işlendiği ve daha sonra ileri iletildiği cihazın kontrol düzlemine yönlendirilir. Yönlendirme protokolü mesajları (örneğin, OSPF bağlantı durumu reklamları) gibi trafiği kontrol etmek için de aynı işlem geçerlidir. Bir paket kontrol düzlemine teslim edildikten sonra, içerdiği bilgiler işlenir ve muhtemelen RIB'nin değişmesine ve ek mesajların eşlerine iletilmesiyle sonuçlanır ve bu güncellemelerden haberdar edilir (yani, yeni bir yol öğrenilir). RIB kararlı hale geldiğinde, FIB hem kontrol düzleminde hem de veri düzleminde güncellenir (4). Daha sonra, yönlendirme güncellenerek bu değişiklikleri yansıtacaktır. Ancak, bu durumda alınan paket öğrenilmemiş bir MAC adresinden biri olduğundan, kontrol düzlemi paketi (C) veri düzlemine (2) geri gönderir, veri düzlemi de paketi uygun şekilde (2) iletir. Eğer ek FIB programlama gerekliyse, bu durum şu an için geçerli olacak olan (C) adımında gerçekleşir, ki bu şimdilik MAC adresleri kaynağı öğrenildiği durumdur. Paket işleme için aynı algoritma bir sonraki sağdaki anahtarda gerçekleşir.

İnternet'in tarihi (Defteri, 2013), ulaşılabilirlik bilgisini yönetmek için kontrol şemalarının gelişimi, ulaşılabilirlik bilgilerinin dağıtımı için protokoller ve çeşitli zorluklar karşısında optimize edilmiş yolların algoritmik olarak üretilmesi ile eşleşmektedir. İkinci durumda, kullanılan bilgi tabanının (yani rota tablosu boyutunun büyümesi) artan bir büyümesini ve bunun nasıl yönetileceğini içerir. Bunu yapmamak,

fiziksel ağda çok fazla kararsızlık olasılığına neden olabilir. Bu da, ağda yüksek oranda değişime ya da çalışmamasına neden olabilir. Yönlendirme bilgilerinin büyüklüğü arttıkça üstesinden gelinmesi gereken diğer bir zorluk, reklamın ulaşılabilirlik konusundaki sorumluluğunun yalnızca veri düzleminin yerel örnekleri arasında değil, aynı zamanda idari sınırlar arasında da varış noktası/ hedef veri bölümlerine yayılmasıdır.



Şekil 2:Tipik bir ağın kontrol ve veri düzlemleri

Az önce tartışılan İnternet için kontrol düzlemi gerçekte katman 2 veya katman 3 kontrol düzlemlerinin (Kahya, 2019) bir birleşimidir. Bu nedenle, aynı ilerleme ve evrimin hem katman 2 hem de katman 3 ağları ve bu kontrol düzlemlerini oluşturan protokoller için gerçekleşmesi şaşırtıcı olmamalıdır. Aslında, bu protokollerin hem işlevsellik açısından gelişmesi hem de bunların nasıl ölçeklenebilir ve yüksek oranda kullanılabilir yollarla uygulanacağını öğrenilmesinin donanım satıcıları açısından gelişmesi ile İnternet'in ilerlemesi gerçekleşmiştir.

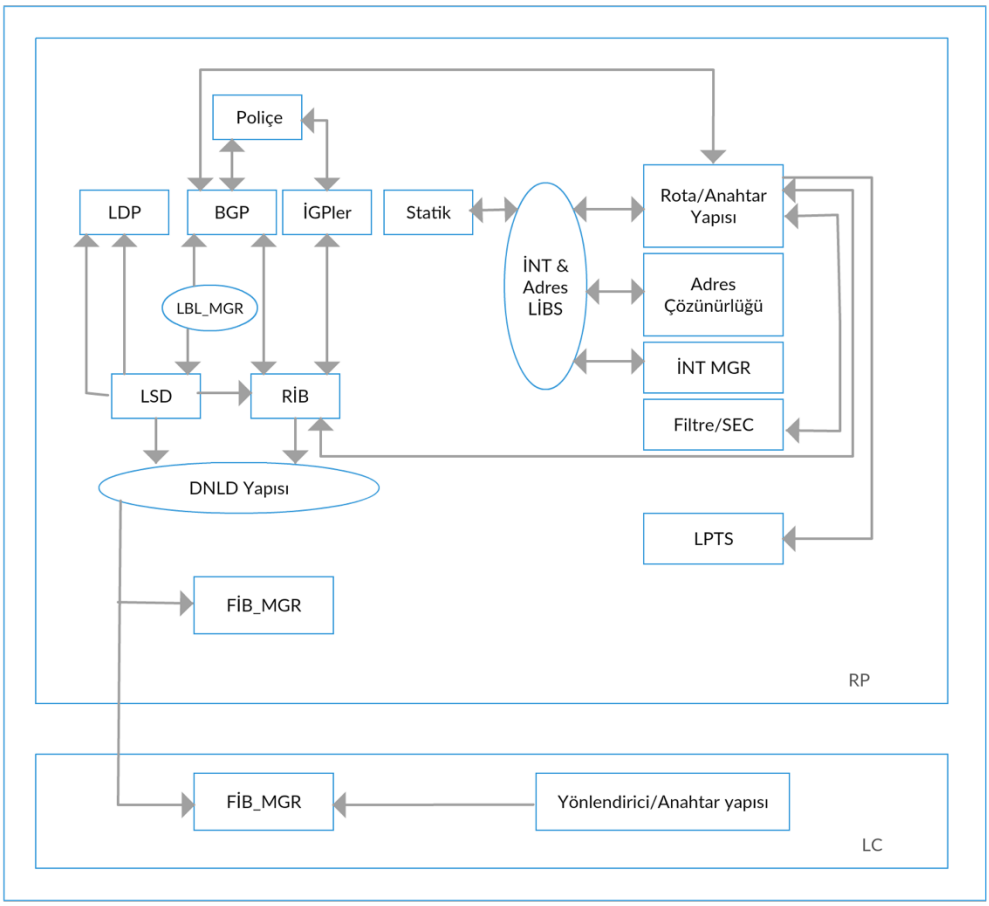
Katman 2 kontrol düzlemi, IEEE MAC (know-how, 2019) adresleri gibi donanım veya fiziksel katman adreslerine odaklanır. IP protokolünün adresleri gibi ağ katmanı adreslerini kolaylaştırmak için bir katman 3 kontrol düzlemi oluşturulmuştur. Bu ve diğer hususları ele alma amacıyla birkaç yineleme veya standart tabanlı katman 2 kontrol protokolü jenerasyonu olmuştur. En önemlisi, bunlar IEEE'den SPB / 802.1aq'i (Fedyk, 2012) ve IETF'den TRILL'i (Holik & Karamazov, n.d.) içermektedir.

Genelleme olarak, 2. katman ağları sonuçta çok sayıda son ana bilgisayar nedeniyle iyi ölçeklenemediği için 2. katman ve 3. katman ölçeklendirmeyle ilgili kaygılar ve bunlardan ortaya çıkan kontrol düzlemi tasarımları sonunda birleşir veya melezleşir. Bu sorunların özü, ağlar arasında hareket eden son ana bilgisayarlarla uğraşmaktır, bu da büyük bir ileri yönlendirme tabloları kaybıyla sonuçlanır ve trafik akışını aksatmayacak kadar hızlı bir şekilde güncellemek zorunda kalır. Bir katman 2 ağında, yönlendirme MAC adreslerinin erişilebilirliğine odaklanır. Bu nedenle, katman 2 ağları öncelikle yönlendirme amacıyla MAC adreslerinin depolanması ile ilgilenir. Ana makinelerin MAC adresleri büyük bir kurumsal ağda çok büyük olabileceğinden yönetimi zordur. Tüm MAC adreslerini birden fazla işletme veya İnternet üzerinden yönetmek daha da zordur.

Katman 3 ağında, yönlendirme ağ adreslerinin erişilebilirliğine odaklanır. Katman 3 ağ erişilebilirliği bilgisi, öncelikle bir hedef IP alan kodu erişilebilirliği ile ilgilidir. Bu, hem tek noktaya yayın hem de çok noktaya yayın (Fairhurst, 2009) için birkaç adres ailesi arasındaki ağ alan kodlarını içerir. Tüm modern durumlarda, katman 2 ölçeği sorunlarının üstesinden gelmek için katman 2 alanlarını bölümlemek veya birleştirmek için katman 3 ağı kullanılır. Özellikle, bazı IP alt ağ kümelerini temsil eden katman 2 köprüleri tipik olarak bir katman 3 yönlendiricisi ile birbirine bağlanır. Katman 3 yönlendiricileri daha büyük ağlar veya gerçekten farklı alt ağ adres aralıkları oluşturmak için birbirine bağlanır. Genellikle büyük ağları birbirine bağlamakta uzmanlaşmış ağ geçidi yönlendiricileri aracılığıyla daha büyük ağlar diğer ağlara bağlanır. Bununla birlikte, tüm bu durumlarda, yönlendirici, katman 3'teki ağlar arasındaki trafiği yönlendirir ve paketin daha sonra belirli bir ana bilgisayara iletilmesi gereken son hedef katman 3 ağlarına ulaştığını bildiği zaman yalnızca katman 2'deki paketleri iletir.

Bu hatların bazı önemli bulandırmaları, Çok Protokollü Etiket Değiştirme (MPLS) protokolü, Ethernet Sanal Özel Ağı (EVPN) (Pepelnjak, 2018) protokolü ve Konumlandırıcı / Kimlik Ayrıştırma Protokolü (LISP) (Meyer, et al., 2007) ile gerçekleşir. MPLS protokolü (gerçekten bir protokoller takımı) ATM'nin IP dünyasından kabul edilen çok esnek ve karmaşık yol işaretleme teknikleriyle icat ettiği son derece hızlı paket iletmeyi paylaşan bir teknolojiyi oluşturmak için katman 2 IP iletiminin en iyi bölümlerini (veya anahtarlamayı) katman 3 IP yönlendirmesinin en iyi bölümleriyle birleştirmeyi temel alarak oluşturulmuştur. EVPN protokolü, uzak katman 2 köprülerinin bir MPLS (veya GRE (Anon., 2019) (Javvin, 2004-2005)) altyapısı üzerinde etkili bir şekilde tünelleşmesiyle açıklanan katman 2 ağ ölçeği sorunlarını çözme çabasıdır; ancak o zaman katman 2 adresleme ve ulaşılabilirlik bilgileri bu tüneller üzerinden aktarılır ve alttaki katman 3 ağlarının ölçeğini bozmaz (veya etkilemez). Uzak köprüler arasındaki ulaşılabilirlik bilgisi, yeni bir BGP adres ailesinin içindeki veriler olarak tekrar temel ağa bulaşmayacak şekilde değiştirilir. Tüneller üzerinde değiştirilen 2. katman adreslerinin miktarını sınırlayan ve yine köprüler arasındaki etkileşim seviyesini optimize eden başka optimizasyonlar da vardır. Yayın ve çok noktaya yayın gereksinimini en aza indiren bir tasarımdır. LISP, genel dağıtılmış kontrol düzlemi modelinin bazı ana hatlarını çok ana noktalara uygulandığı gibi, bazı yeni adresleme alanları ekleyerek ve site adresini sağlayıcıdan yeni bir harita ve kapsüllüme kontrol ve iletme protokolünde ayırmaya çalışmaktadır. Biraz daha düşük bir seviyede, daha büyük kontrol düzleminin bilgisini arttırmak için kullanılan belirli ağ tiplerine özgü ek kontrol işlemleri vardır. Bu işlemler tarafından sağlanan hizmetler arasında bağlantı kullanılabilirliği veya kalite bilgilerinin doğrulanması/bildirilmesi, komşu keşfi ve adres çözümlemesi bulunur.

Bu hizmetlerin bazıları çok sıkı performans döngülerine sahip olduğundan (kısa olay tespit süreleri için), kontrol düzlemi için seçilen stratejiden bağımsız olarak neredeyse her zaman veri düzleminde (örneğin OAM (Ergun, 2015)) yereldir. Bu, Şekil 3'te, kontrol düzleminin kalbini oluşturan çeşitli yönlendirme protokollerinin yanı sıra RIB-FIB kontrolünün tasviri ile gösterilmektedir. Kontrol ve veri düzlemlerinin bulunduğu yeri belirtmediğimizi, yalnızca veri düzleminin hat kartında (LC kutusunda Şekil 3'te gösterildiği gibi) bulunduğunu ve kontrol düzleminin rota işlemcisinin üzerinde bulunduğunu belirtmediğimizi unutmayın (RP kutusu ile belirtilir).



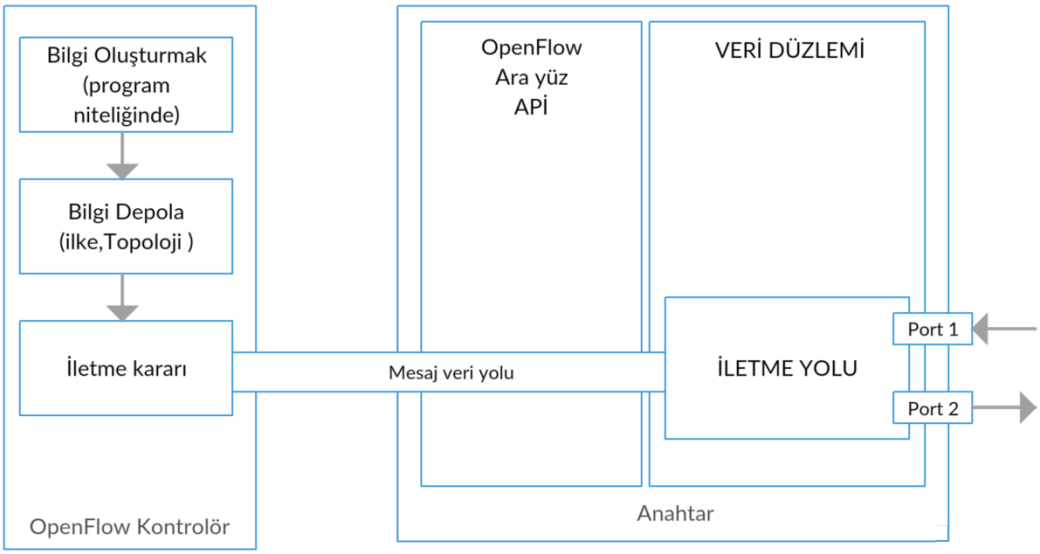
Şekil 3:Tipik bir ağ cihazının kontrol ve veri düzlemi

D. SDN Anahtarı

Bir yandan, SDN mimarisi sayesinde, ağ bir "basit" paket iletme elemanı haline gelir. Öte yandan, üst düzey yönlendirme kararları ve durum bilgileri Şekil 4'te gösterildiği gibi dağınık cihazların bir evrimi üzerine dayatmacı ilkeler uygulamak ve protokolleri yürütmek yerine, harici ve ayrı bir sunucu kontrolöründe merkezileştirilmiştir (ONGARO, 2013-2014).

SDN, kontrol düzlemini veri düzleminden ayırarak, esnek bir ağ otomasyonu ve yönetim çerçevesi sunabilir. Bu çerçeve görevleri otomatikleştirmek için araçlar geliştirmeyi (bugün manuel olarak yapılmaktadır) mümkün kılar. Bu otomasyon araçları, operatör hatası ile ortaya çıkan operasyonel ek yük azaltan ağ kararsızlığını azaltabilir. Maalesef, satıcıların yazılım ortamları genellikle tescilli ve kapalıdır ve ağın yönetimini ve düzenlemesini kolaylaştırmaz. Bununla birlikte, SDN mimarisi yeniliği kolaylaştırabilir ve programlanabilir ağlar fikrine yol açan ağ veri yolunun

basit programa ait kontrolünü sağlayabilir. Bu, yeni fikirler için girişin önündeki engeli azaltmaya izin veren önemli bir husustur (ONGARO, 2013-2014).



Şekil 4:SDN anahtar bileşenleri

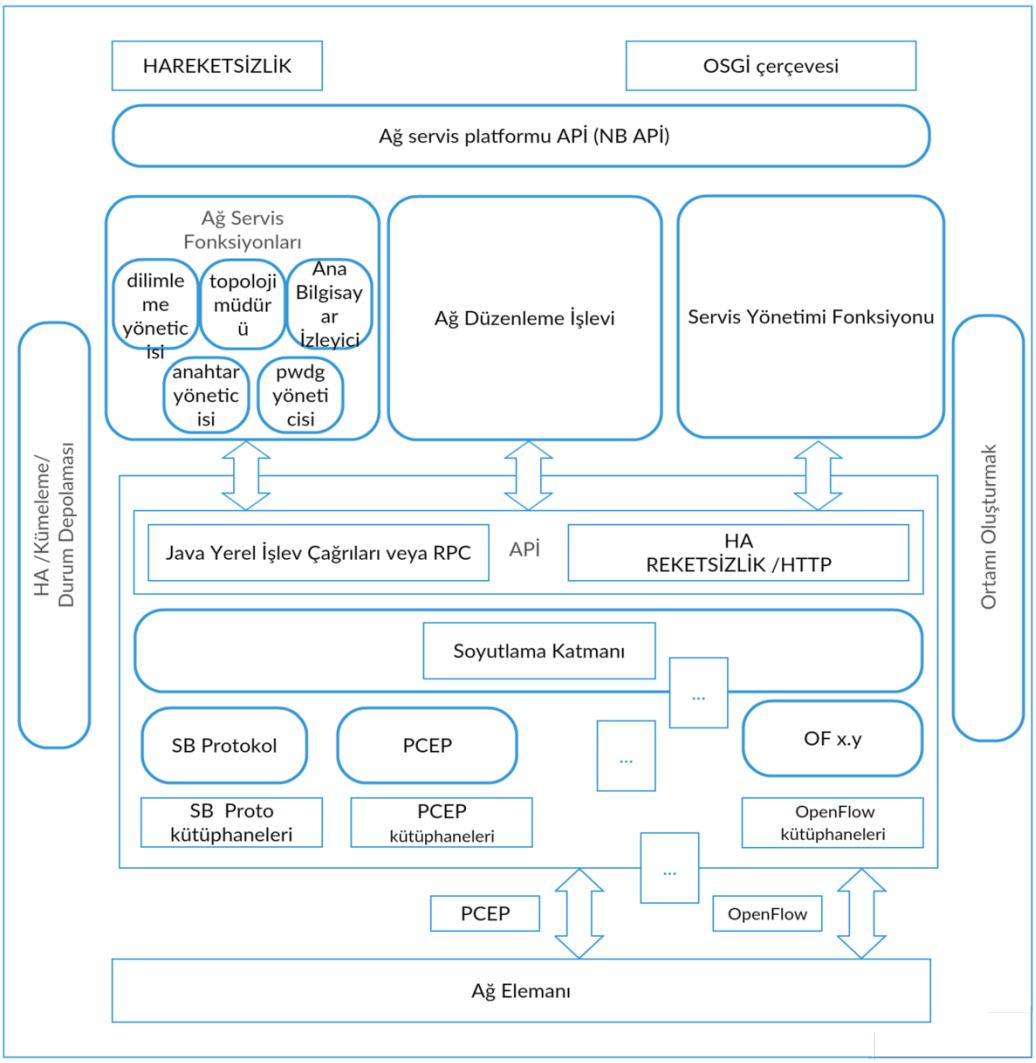
E. SDN Denetim Birimleri

Merkezileşme derecesine bakılmaksızın, geçici ağ durumunun merkezi bir kontrol modelinde yönetimi, kontrol ve veri düzlemlerinin ayrılması ve programlanabilirlik SDN'in en belirgin üç kavramıdır (Kesden, 2014).

Teoride, bir SDN kontrol cihazı, geçici durum yönetimi ve merkezileşme kavramlarını aşmanın yanı sıra, dağıtılmış bir kontrol düzlemini gerçekleştirebilecek hizmetler sunar (Göransson & Black, 2014). Gerçekte, herhangi bir denetleyicinin herhangi bir örneği, bu işlevlerin bir dilimini veya alt kümesini ve ayrıca bu kavramları üstlenmesini sağlayacaktır. Bu bölümde, hem ticari satıcılardan hem de açık kaynak topluluğundan gelen en popüler SDN denetleyici tekliflerini ayrıntılı olarak ele alacağız.

1. Genel konsept

İdealize edilmiş bir kontrol cihazı Şekil 5'te gösterilmiştir.



Şekil 5:İdealleştirilmiş Denetleyici/Çerçeve

SDN denetleyicisinin genel açıklaması, bir yazılım sistemi veya birlikte sağlayan sistemler topluluğudur:

- Ağ durumunun yönetimi ve bazı durumlarda bu durumun yönetimi ve dağıtımı bir veri tabanı içerebilir. Bu veri tabanları, kontrol edilen ağ elemanlarından ve ilgili yazılımdan türetilen bilgilerin yanı sıra ağ durumu, bazı geçici yapılandırma bilgileri, öğrenilmiş topoloji ve kontrol oturumu bilgisi dahil olmak üzere SDN uygulamaları tarafından kontrol edilen bilgiler için bir depo görevi görür. Bazı durumlarda, denetleyici birden çok amaca yönelik veri yönetimi işlemine(örneğin ilişkisel ve ilişkisel olmayan veri tabanlarına) sahip olabilir. Diğer durumlarda, diğer bellek içi veri tabanı stratejileri de kullanılabilir.

- Yönetilen kaynaklar, ilkeler ve denetleyici tarafından sağlanan diğer hizmetler arasındaki ilişkileri yakalayan üst düzey bir veri modelidir. Çoğu durumda, bu veri modelleri Yang modelleme dili kullanılarak oluşturulmuştur.
- Denetleyici hizmetlerini bir uygulamaya maruz bırakan, modern, genellikle RESTful (temsili durum aktarımı) uygulama programlama arabirimi (APİ) sağlanır (Göransson & Black, 2014). Bu, denetleyici-uygulama etkileşiminin çoğunu kolaylaştırır. Bu arayüz ideal olarak, kontrolörün servislerini ve özelliklerini tanımlayan veri modelinden sağlanmıştır. Bu arayüz ideal olarak, kontrolörün servislerini ve özelliklerini tanımlayan veri modelinden sağlanmıştır. Bu arayüz ideal olarak, kontrolörün servislerini ve özelliklerini tanımlayan veri modelinden sağlanmıştır. Bazı sistemler daha da ileri giderek denetleyici özelliklerinin dinamik genişlemesini destekleyenler de dahil olmak üzere çekirdek modüllerin genişletilmesine ve daha sonra yeni modüller için APİ'lerin yayınlanmasına izin veren güçlü geliştirme ortamları sunar:
    - Denetleyici ve ağ öğelerindeki ilişkili aracılar arasında güvenli bir TCP denetim oturumu
    - Ağ elemanlarında uygulamaya yönelik ağ durumunun sağlanması için standartlara dayalı bir protokol
    - Bir cihaz, topoloji ve servis bulma mekanizması; bir yol hesaplama sistemi; ve potansiyel olarak diğer ağ merkezli veya kaynak merkezli bilgi hizmetleri

Mevcut kontrol cihazı tabiatı, VMware (vCloud/vSphere), Nicira (NVP) (Pepelnjak, 2013), NEC (Trema), Büyük Anahtar Ağları (Floodlight/BNC) (Rao, 2015) ve Juniper/Contrail (Linnerz, 2019) ticari ürünlerini içerir. Aynı zamanda bir dizi açık kaynaklı kontrol cihazı içermektedir.

OpenFlow ve tescilli protokollerin kullanımının yanı sıra, veri merkezi için katman 3'e göre 3 kiracı ayırma modeli veya WAN'daki katmanlar için MPLS LSP'ler olarak MPLS VPN'leri oluşturmak için IP / MPLS ağ işlevselliğini kullanan SDN kontrolörleri vardır.

NETCONF (Anon., 2019) tabanlı kontrol cihazlarının neredeyse ağ yönetim çözümlerinden ayırt edilemeyeceği ya da mobil ortamlarda PCRF (Anon., 2019-2020) ve / veya TDF (Anon., 2013) gibi Radius / Çap tabanlı kontrol cihazlarının da SDN

kontrol cihazları olduğu iddialarını görmezden gelemeyiz. Bu, özellikle güneye bağlı protokolleri daha bağımsız hale geldiği ve geçici ağ / konfigürasyon durumu oluşturabildiği için geçerlidir.

Veri merkezi orkestrasyonunun orijinal SDN uygulaması, entegre bir çözümün parçası olarak SDN kontrol cihazlarının çoğalmasına neden olmuştur. Bilgi işlem, depolama ve sanal makine görüntüleri gibi veri merkezi kaynaklarının ve ayrıca ağ durumunun yönetilmesine odaklanmış kullanım durumudur. Son zamanlarda, bazı SDN kontrolörleri, ağ soyutlamasının yönetimi konusunda uzmanlaşmış olanı ortaya çıkarmaya başlamış ve açık kaynak API'lerinin (OpenStack (Anon., n.d.), Cloudstack (Anon., 2017)) desteği ile veri merkezlerinde gerekli olan kaynak yönetimi ile birleştirilmiştir. Bu ikinci kontrolör dalgasının sürücüsü, SDN uygulamalarının veri merkezinden dışarı ve ağın diğer alanlarına, işleme ve depolama gibi sanal kaynakların yönetiminin bir çözelti içinde bu kadar sıkı bir şekilde bağlanması gerekmediği potansiyel genişlemesidir.

Ağ veri merkezi sektöründe büyüme, hiper yönetici anahtarı / yönlendirici / köprü yapısına odaklanan çok sayıda yeni ağ elemanını da tanıtmıştır. Bazen Ağ İşlevleri Sanafllaştırma (NFV) (Tong & Wade, 2017) olarak da adlandırılan ağ hizmeti sanallaştırması, gelecek nesil ağ mimarisine bu öğelerin daha da fazlasını ekleyerek denetleyicinin bu işlemleri yapması ve yönetmesi gerekliliğini vurgulamaktadır.

Sanal anahtarlar veya yönlendiriciler, ağ ortamındaki en düşük ortak paydayı temsil eder ve genellikle, donanım odaklı çalışan aynı türden daha az sayıda iletme girişi yapabilir. Teknik olarak bir servis VM'sindeki büyük tabloları destekleyebilmelerine rağmen, gerçek limitleri servis VM'siz davranışlardadır. Özellikle, sadece amaca yönelik tasarlanmış yönlendiricilerde veya anahtarlarda bulunan özel donanımda uygulanan hipervizör (Parlakyigit, 2013) içindeki entegre tablo ölçeği ve yönetim yeteneğidir. Daha basit bir hipervizör tabanlı iletme yapısının, geleneksel bir amaca yönelik elemanda mevcut olan RIB / FIB kombinasyonuna yer yoktur. Bu, dağıtılmış ağ bilgilerini bu birkaç girişe kaynatmak için yardıma ihtiyaç duyan dağıtılmış kontrol paradigmasında geçerlidir- bu girdiler ya ana bilgisayar oluşturma işleminin bir parçası olarak oluşturulan ve ana bilgisayar üzerinde bir hizmet VM'si olarak çalışan bir kullanıcı alanı aracısından veya SDN denetleyicisinden gelmektedir. İkinci durumda ise, bu dağıtılmış ortamda bir vekil

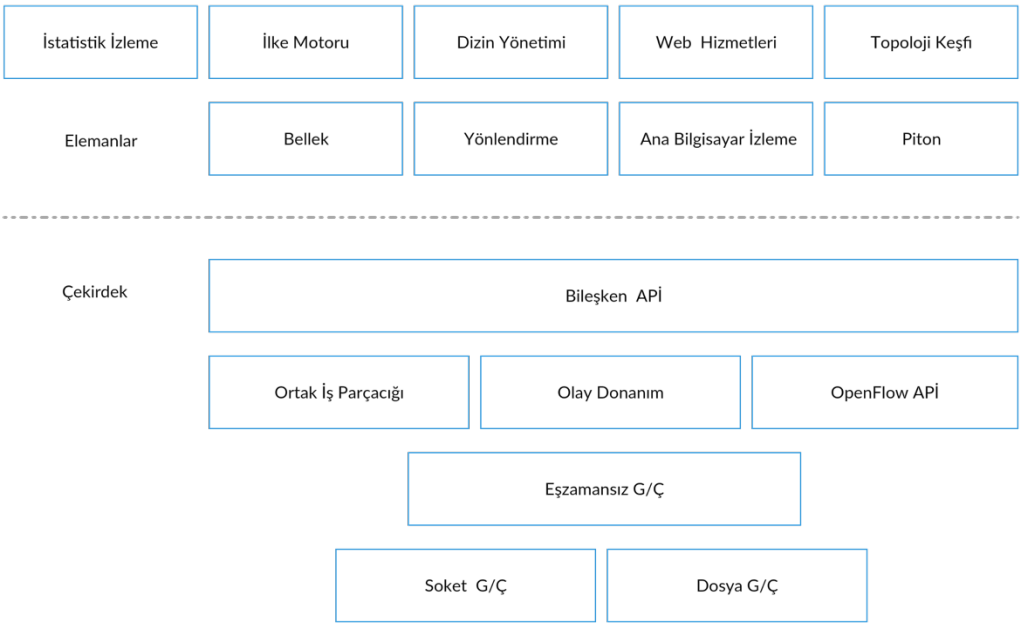
olarak ya da idari olarak dikte edilmiş merkezi bir ortamda akış sağlama maddesi olarak görev yapan SDN denetleyicisi olabilir. Bu şekilde, kontrol cihazı geleneksel olarak bir ağ OSS'sine maruz kalan bir ağın yönetim katmanını önleyebilir.

Bir veri merkezindeki ana bilgisayarlardaki yazılım anahtarları / yönlendiricileri için SDN denetleyicisi kritik bir yönetim arabirimidir. SDN kontrolörleri, analitik ve olay bildirimi gibi geçici ağ varlıklarından (ajan aracılığıyla), ilgili durumdan sorumlu olduklarından bazı yönetim hizmetleri (sağlama ve keşfetmeye ek olarak) sunarlar. Bu yönüyle SDN, ağ öğesi yönetimi (EMS) (Pelosi, 2006) hakkındaki görüşümüzde devrim yaratma potansiyeline sahiptir.

2. Nox/Pox

NOX / POX web sitesine göre (Gude, et al., 2008)NOX, Nicira tarafından geliştirilmiştir ve araştırma topluluğuna bağışlanarak böylelikle 2008'de açık kaynak olmuştur. Daha sonra UC Berkeley ve ICSI'nin katkılarıyla Stanford Üniversitesi'ndeki ON.LAB etkinliği aracılığıyla genişletilmiştir ve desteklenmiştir. NOX, OpenFlow'a (OF v1.0) bir C ++ API'si ve eş zamansız, olaya dayalı bir programlama modeli sağlar.

NOX hem ilkel denetleyici hem de SDN uygulamaları geliştirmek için bileşen tabanlı bir çerçevedir. OpenFlow'a özgü destek modülleri sağlar ancak genişletilebilir ve uzatılabilmektedir. NOX çekirdeği, bir bağlantı işleyicisi ve olay motoru dahil olmak üzere OpenFlow anahtarları ile etkileşimde bulunmak için yardımcı yöntemler ve API'ler sunar. Şekil 6'da gösterildiği gibi, ana bilgisayar izleme, yönlendirme, topoloji (LLDP) (Congdon, 2003) ve bileşen API için sarıcı olarak uygulanan bir Python arayüzü dahil olmak üzere, bu API'yi kullanan ek bileşenler mevcuttur.



Şekil 6:NOX Mimarisi

NOX genellikle ağ protokolü araştırması gibi SDN uygulamaları geliştirmek için akademik ağ araştırmalarında kullanılır. Yaygın akademik kullanımının önemli bir dolaylı etkisi, çeşitli programlama projeleri ve deneyler için başlangıç kodu olarak kullanılabilecek bir öğrenme anahtarı ve ağ çapında bir anahtarı taklit etmek için örnek kodun mevcut olmasıdır. Bazı popüler NOX uygulamaları SANE (Casado, et al., 2006) ve Ethane'dir. Ethane (Project, 2006), geleneksel bir erişim kontrol listesi düzeyinde merkezi, ağ çapında güvenlik için bir Stanford Üniversitesi araştırma uygulamasıdır. Her ikisi de, geçmişte benzer işlevleri uygulamak için önemli ölçüde daha fazla kod alan bu işlevleri uygulamak için önemli ölçüde gerekli olan kod satırlarını azaltarak SDN'in etkinliğini göstermiştir. Bu başarıya dayanarak, araştırmacılar bir NOX çekirdeğinin üzerine MPLS benzeri uygulamalar göstermekteler. POX, NOX'in (Python'daki NOX) daha yeni, Python tabanlı sürümüdür. Gelişiminin arkasındaki fikir, NOX'i C ++ köklerine geri döndürmek ve ayrı bir Python tabanlı platform geliştirmektir (Python 2.7). Sorgulama yapabilen bir topoloji grafiği ve sanallaştırma desteği içeren üst düzey bir SDN API'ye sahiptir.

a. Pox Denetim Birimi

POX, SDN uygulaması için geliştirilen birçok denetleyiciden biridir. POX, NOX'in kardeş projesidir ve NOX'ten daha hızlı hale gelmiştir. POX python tabanlı bir OpenFlow denetleyicisi ve açık kaynak kodlu bir geliştirme platformudur. POX'un

temel avantajı, ağ uygulamaları geliştirmek için Python tabanlı Yazılım Tanımlı Ağ oluşturma denetleyicisi olmasıdır. POX kullanmamızın nedeni hali hazırda mevcut, öğrenmesi ve uygulamaları geliştirmesi kolaydır. Kolay ve ücretsiz bir kurulum olan PyPy çalışma zamanı ile birlikte çalıştığı için her yerde çalışabilir. POX, bir arke tipik, modern Yazılım Tanımlı Denetleyici oluşturmaya adanmış denetleyicilerden biridir. ONOS, Floodlight, NOX, vb. gibi farklı SDN kontrol cihazları mevcuttur. POX'i diğer kontrol cihazlarına göre seçmemizin nedeni, farklı kontrol cihazlarının farklı bir gereksinimi olmasıdır. Ryu, python tabanlı SDN kontrol cihazlarından biridir, ancak kontrolörler arası iletişimi kuramaz. Gelecekteki çalışmalar için Ryu denetleyici verimli olmayacak ve çoklu denetleyici mimarisini desteklemeyecektir. ONOS, dağıtık mimariyi destekleyen, ancak çok fazla bağımlılığa sahip kontrol cihazlarından biridir. Hala az gelişmiştir ve üzerine yeni uygulamalar geliştirmeyi zorlaştırmaktadır. Floodlight, en yaygın kullanılan SDN denetleyicilerinden biridir, ancak API, Java tabanlıdır ve python kullanılarak geliştirilemez. POX, beraberinde gelen stok bileşenlerini kullanarak derhal temel bir SDN kontrol cihazı olarak kullanılabilir. POX çerçeveleri OpenFlow'un her iki tarafı için; biri anahtar tarafı ve diğeri kontrol cihazı tarafı içindir. Toplayıcı, diğer denetleyicilerin buna bağlanmasına ve buna OpenFlow komutları göndermesine izin veren bir anahtardır. Ancak, altındaki diğer OpenFlow anahtarlarını kontrol ederek bunu gerçekleştirir (Harsh, 2016).

POX, NOX'e göre aşağıdaki avantajları iddia etmektedir:

- POX bir Pythonik OpenFlow arayüzüne sahiptir.
- POX, yol seçimi, topoloji keşfi vb. için yeniden kullanılabilir örnek bileşenlere sahiptir.
- POX her yerde çalışır ve kolay dağıtım için kurulum gerektirmeyen PyPy çalışma zamanı ile birlikte gelir.
- POX özellikle Linux, Mac OS ve Windows'u hedefler.
- POX, GUI ve görselleştirme araçlarını NOX ile aynı şekilde destekler.
- POX, Python ile yazılmış NOX uygulamalarına kıyasla iyi bir performans sergilemektedir.

NOX ve POX şu anda OpenFlow v1.0 anahtarları ile iletişim kurmaktadır ve Open vSwitch için özel destek içermektedir (Nadeau & Gray, 2013).

F. SDN Zayıflıkları ve Zorlukları

Bu bölümün odağı, SDN ağları ve OpenFlow protokolü ile çalışmanın zayıflıkları ve zorlukları ile ilgilidir. SDN ve OpenFlow, ağ elemanlarının proto tipleme, dağıtım ve yönetimini basitleştirmenin bir yolunu sunar. Bununla birlikte, ağın güvenli olmayan ya da kullanılamayan bir durumda olmasına neden olabilecek bazı ilginç hususları (Lara, et al., First 2014) (Fernandez, 2013) da göz önünde bulundurmalıyız:

- Denetleyicinin kullanılabilirliği, dikkate alınması gereken ana unsurdur. Kurallarda değişiklik yapılması gerektiğinde anahtarlar ile denetleyici arasındaki sıkı bağımlılık bir sorun olabilir. Dahası, eğer ağ tasarımı yalnızca bir merkezi denetleyiciyi dikkate alırsa, "tek bir başarısızlık noktası" olabilir. Kullanılabilirliği sağlamak ve istenmeyen arızaları önlemek için dağıtılmış bir yaklaşım uygulanabilir. Kullanılabilirliği sağlamak ve potansiyel istenmeyen arızaları önlemek için dağıtılmış bir yaklaşım uygulanabilir. Ek olarak, sağlamlığı sağlamak için bir miktar yedekleme veya yedekleme çözümü kullanılabilir.
- Güvenlik de önemlidir. SDN'de denetleyici, ağın kritik bilgisine sahip bir bileşendir ve bu yönü denetleyiciyi olası saldırı ve tehditlere maruz bırakır. Ek olarak, kontrol cihazı ve anahtarlar arasındaki kanallar korunmasız olabilir. OpenFlow teknik özelliklerine göre, TLS protokolü ile güvenli iletişim kullanmak mümkündür; ancak kullanımı gerekmediğinden ağın tasarımına bağlıdır.
- Akış tablolarının tutarlılığı da potansiyel bir konudur. Birkaç kontrol cihazı aynı akış tablolarını yönetebildiğinden, örneğin bir üretim donanım denetleyicisi ve diğer bazı deneysel denetleyiciler, ikincisinin "zincirdeki en zayıf halka" olacağını izler. Sonuç olarak, daha düşük güvenlik kontrollerine tabi olabilirler, akış tablolarının tutarsız bir durumda olmasına neden olur. Akış vizörünün bir uygulaması bu potansiyel tehditlerden kaçınmak için uygun olabilir.
- Ağın ölçeklenebilirliği ayrıca denetleyiciye bağlıdır, potansiyel olarak "darboğaz" olabilir. Denetleyiciye çok fazla paket ulaştığında, ağda

performans sorunları ortaya çıkabilir. Kontrol düzleminin dağılımını hesaba katmanın önemli olduğunu takip eder.

- Ağın performansı aynı zamanda benimsenen kontrol modeliyle de ilişkilendirilebilir.Akış tablosu büyüklüğü sınırlı olduğundan, çok sayıda akışın yönetimi hala güçlü bir zorluktur. Ancak iyi tasarlanmış bir ağ, proaktif bir yaklaşımla performans sorunlarını azaltabilir. Aslında, proaktif yaklaşım, reaktif moddan daha iyi bir performansa ulaşır, çünkü kontrol cihazı ve anahtarlar arasında değiştirilen mesaj sayısını sınırlar.

Yukarıda tarif edilen hususlar topluluk araştırmacılar tarafından dikkate alınmıştır ve SDN için gelecekteki zorlukları temsil etmektedir (ONGARO, 2013-2014).

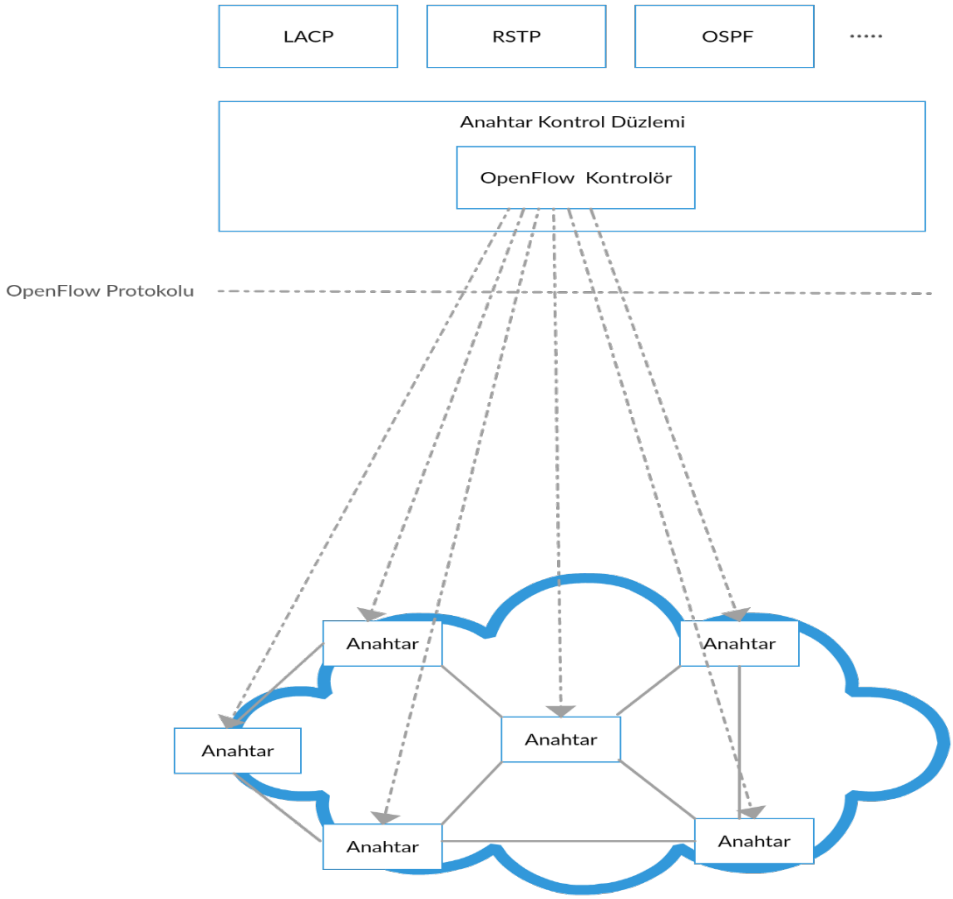
III. OPENFLOW

A. OpenFlow Tarihçesi

OpenFlow aslen Stanford Üniversitesi'ndeki ağ araştırmasının bir parçası olarak hayal edilmiş ve uygulanmıştır. Asıl odak noktası, araştırma ve deney için kullanılabilecek kampüs ağlarında deneysel protokoller oluşturulmasına izin vermektir. Bundan önce, üniversiteler sıfırdan kendi deney platformlarını oluşturmak zorunda kalmışlardır. Bir fikrin bu ilk çekirdeğinden gelişen, OpenFlow'un katman 2 ve katman 3 protokollerinin işlevselliğini tamamen ticari anahtarlarda ve yönlendiricilerde değiştirebileceği görüşüydü (Yazar, 2013). Bu yaklaşım genellikle temiz kayrak önerisi olarak adlandırılır. 2011 yılında, OpenFlow'un üretim ağlarında kullanımını ticarileştirmek, standardize etmek ve teşvik etmek için bir grup servis sağlayıcı tarafından Açık Ağ Kuruluşu (ONF) (Anon., 2019) adlı kar amacı gütmeyen bir konsorsiyum kurulmuştur. ONF, OpenFlow protokolünü ve diğer SDN ile ilgili çabaları teşvik etmek için kullanılan çok aktif bir pazarlama departmanına sahip olması nedeniyle yeni bir Standart Geliştirme Örgütüdür. Organizasyon, bu çabaların bir parçası olarak Açık Ağ Zirvesi adlı yıllık bir konferansa ev sahipliği yapmaktadır. Genel tabloda, ONF'un yazılım tanımlı ağlar fenomenine dikkat çekerek kredilendirilmesi gerekmektedir.

OpenFlow modelinin temel bileşenleri, Şekil 7'de gösterildiği gibi, en azından SDN genel tanımının bir parçası haline gelmiştir:

- Kontrol ve veri düzlemlerinin ayrılması (ONF durumunda, kontrol düzlemi mantıksal olarak merkezi bir kontrol sistemi üzerinde yönetilir).
- Durum oluşturma için denetleyici ve ağ öğesindeki bir aracı arasında standartlaştırılmış bir protokol kullanma (OpenFlow durumunda, iletme durumu).
- Modern ve genişletilebilir bir API ile ağ programlanabilirliğini merkezi bir bakış açısıyla sağlamak.



Şekil 7:OpenFlow mimarisi(Bazı kontrol düzlemi uygulamalarının,geleneksel kontrol düzlemi uygulamalarının davranışını taklit eden denetleyicinn ÜSTÜ üzerinde süreceği görüşünde)

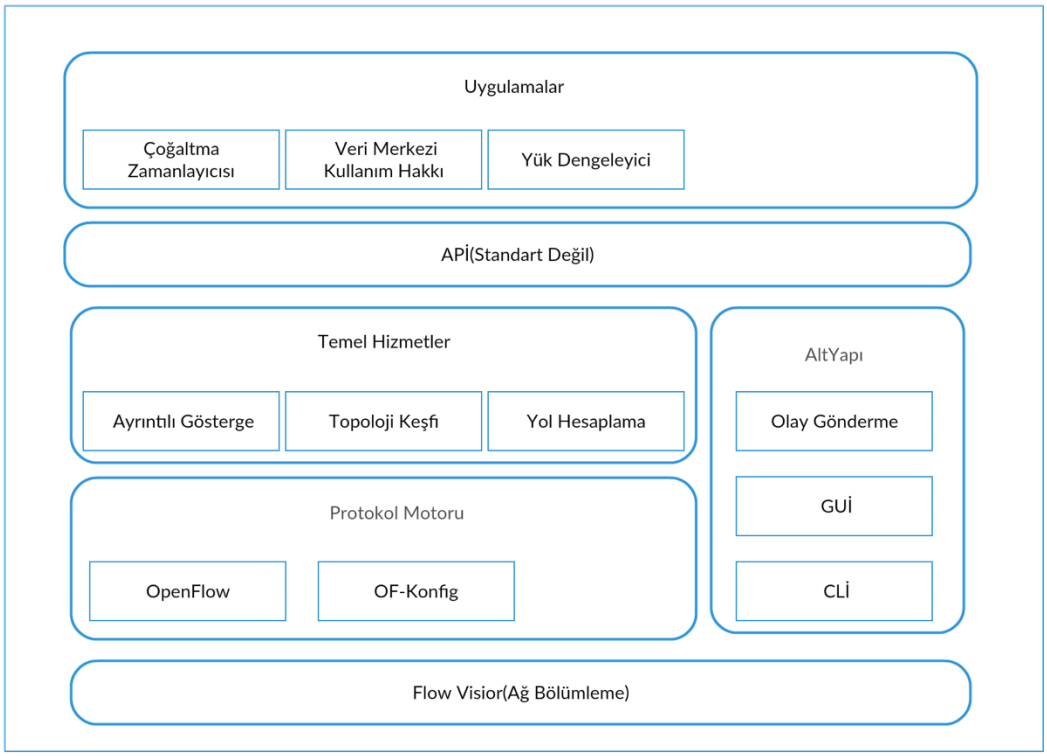
B. OpenFlow Mimarisi

OpenFlow, akışları başlatmak için standartlaştırılmış güneye bağlı bir protokolü (kontrol birimine eleman aracısı) sağlarken, kuzeyden (uygulamaya bakan) API veya doğu / batı API için standart yoktur.

Çoğu kontrol cihazındaki mevcut doğu / batı durum dağılımı, tek bir satıcının kontrolörlerinin federasyonuna izin veren ancak birlikte çalışabilir bir durum değiş tokuşuna izin vermeyen bir veri tabanı dağıtım modeline dayanmaktadır.

Mimarlık Çalışma Grubu (Foundation, 2016), en azından dolaylı olarak bunu ele almaya çalışmaktadır- SDN için genel bir SDN mimarisi tanımlamaktadır. ONF, SDN ve OpenFlow'un tanımı ile birlikte geçmişine de sahiptir. Bu standartlaştırılmış ara yüzler olmadan, soru ONF'nin SDN tanımı açıklığı ima edip etmediği sorusudur. Çoğu OpenFlow kontrol cihazı (Şekil 8) temel bir dizi uygulama servisi sunar: yol hesaplama, topoloji (topolojiyi katman 2 ile sınırlayan LLDP ile belirlenir) ve

şartlandırmadır. Config'i desteklemek için bir NETCONF sürücüsünü desteklemeleri gerekir.



Şekil 8:OpenFlow denetleyici bileşenleri(FlowVisor ve uygulamaları ayrı varlıklardır)

SDN mimarisi ve OpenFlow hakkında süregelen sorular, bir OpenFlow kontrol cihazı (ve OpenFlow'un çalıştığı ağ katmanı) tarafından sağlanan uygulama hizmetleri türlerinin, tüm potansiyel SDN uygulamaları için yeterli olup olmadığı hakkındadır. OpenFlow modeli etrafındaki makro konular üzerine araştırma (örneğin, sorun giderme, OpenFlow anlambilimiyle üst düzey politikaların ifade edilmesi ve kontrolörler ile unsurlar arasında bir doğrulama katmanı ihtiyacı) birçok akademik ve araştırma tesisinde ancak özellikle Açık Ağ Araştırma Merkezinde (ONRC) gerçekleştirilmektedir (Nadeau & Gray, 2013).

C. OpenFlow Protokolü

OpenFlow, kendi başına bir ürün olmamakla birlikte bir ürünün tek bir özelliği değil; bir protokoller kümesi (Göransson & Black, 2014)ve bir API'dir. Başka bir deyişle, kontrolör hangi öğelere (hangi sebeplerden dolayı) gideceğine dair talimatlar veren bir uygulama programı olmadan (muhtemelen birden fazla) hiçbir şey yapmaz.

OpenFlow protokolleri şu haliyle iki bölüme ayrılmıştır:

- Bir kontrol oturumu oluşturmak, akış modifikasyonlarını değiştirmek için bir mesaj yapısı (akış modları) ve istatistik toplamak ve bir anahtarın temel yapısını (portlar ve tablolar) tanımlamak için bir tel protokolü (şu an sürüm 1.3.x) oluşturmaktadır. Sürüm 1.1, birden fazla tabloyu, saklı eylem yürütmeyi ve meta veri geçişini destekleme yeteneği eklemiş - sonuçta akışları idare etmek için bir anahtar içerisinde mantıksal boru hattı işlemi yaratmıştır.
- Belirli bir denetleyiciye fiziksel anahtar bağlantı noktaları tahsis etmek için NETCONF'a (Yang veri modellerini kullanarak) dayalı config (şu anda sürüm 1.1) olan bir yapılandırma ve yönetim protokolü, yüksek kullanılabilirliği (etkin / bekleme) ve denetleyici bağlantı arızası davranışlarını tanımlar. OpenFlow, OpenFlow komutunun / kontrolünün temel çalışmasını yapılandırabilse de (henüz) bir öğeyi önyükleyemez veya koruyamaz (bir FCAPS bağlamında yönetemez).

2012 yılında, ONF birlikte çalışabilirliği ve uyumluluğunu test etmek için "plugfest"lerden daha resmi bir teste (Indiana Üniversitesi dış kaynaklı) geçmiştir. Bu, OpenFlow sonrası tel sürüm 1.0 ilkel kümesinin karmaşıklığından kaynaklanmaktaydı (Nadeau & Gray, 2019).

ONF (Anon., 2019), referans uygulaması oluşturmayı tartıştığı halde, bu yazı itibariyle, bu gerçekleşmemiştir (birçok açık kaynak denetleyici uygulaması vardır).

OpenFlow protokolleri doğrudan ağ dilimlemesini sağlamaz (bir öğeyi ayrı kontrol edilen port gruplarına veya bir ağı ayrı yönetim alanlarına bölme olanağı sağlayan çekici bir özellik).

Ancak, FlowVisor (Sherwood, et al., 2009) (birden çok denetleyici ve öğe arasında saydam bir proxy görevi görür) ve belirli satıcı uygulamaları (ayrı denetleyici oturumlarıyla birden çok sanal anahtarın oluşturulmasını sağlayan aracılar) gibi araçlar bunu mümkün kılar.

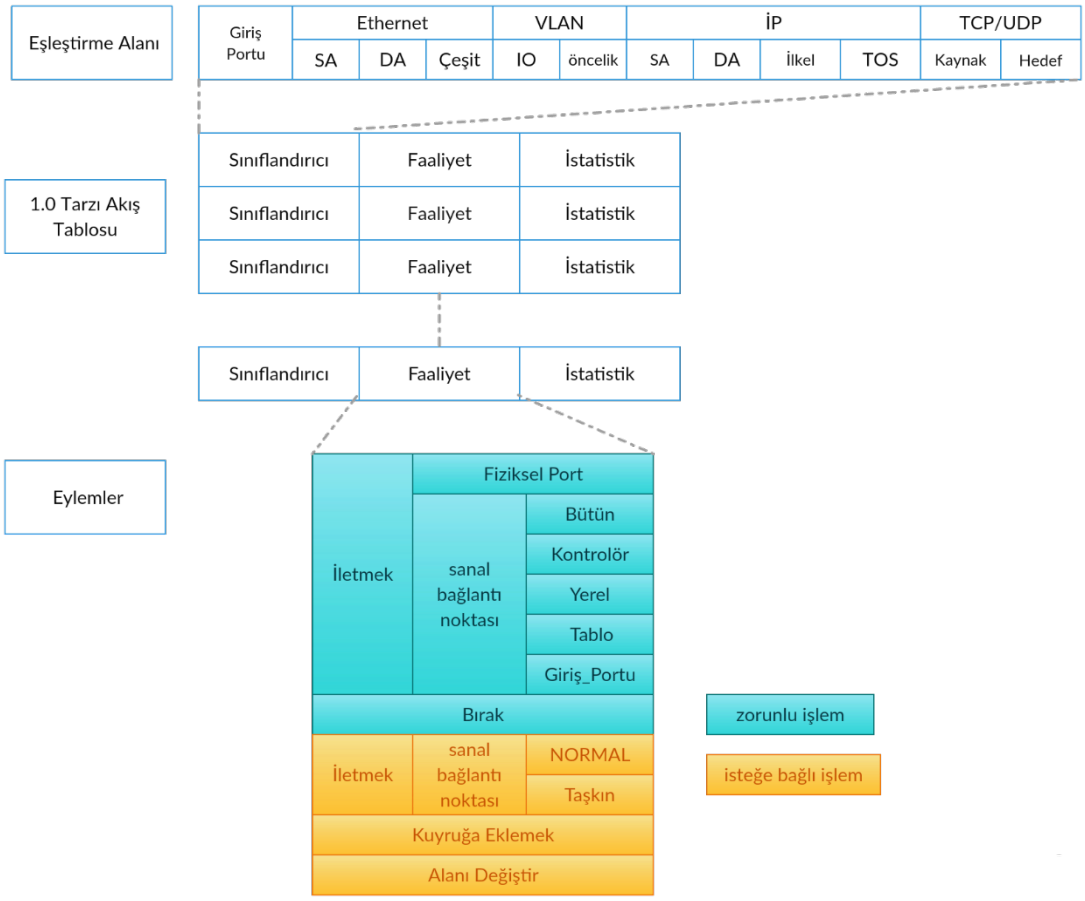
1. Tel Protokolü

İlk olarak, çeşitli üreticilerin protokol konfigürasyonunun katı ve standartlaştırılmamış semantikleri için geçici durumun (akış girişleri ağ üzerinde kalıcı bellekte saklanmamaktadır) ikame kavramını ortaya koymaktadır (McKeown, 2010).

Geçici durum ayrıca, ağ otomasyonunda geçmişteki denemelerin modellerini daha yavaş yapılandıran yapılandırma işlemlerini atlar.

Çoğu ağ mühendisi için, bu tür yapılandırmanın nihai sonucu yönlendirme durumu oluşturmaktır (dağıtılmış kontrol ortamında dağıtılmış ve öğrenilmiş olsa dahi). Aslında, çoğu için, uygun konfigürasyon testi, yönlendirme durumunu doğrulamaktır (yönlendirme (Anon., 2005), gönderme veya köprüleme tablolarına bakılmalıdır. Tabii ki, bu durum, yönetim yükünün bir kısmını-en azından bu durumun sürdürülmesi için (proaktif olunmak ve her zaman yönlendirme tablosunda belirli yönlendirme kurallarına sahip olunmak istenilmekte ise) yapılandırma standartlarının dağıtılmış yönetimini ağ öğelerine kıyasla denetleyicilere kaydırır.

İkinci olarak, bir OpenFlow akış girişinde, paket başlığının tamamı (en azından katman 2 ve katman 3 alanları), Şekil 9'da gösterildiği gibi eşleştirme ve değiştirme eylemleri için kullanılabilir. Alan eşleşmelerinin çoğu maskelenebilir. Bunlar, OpenFlow'un (Ching-Hao, et al., 2015) farklı sürümleri üzerinde gelişmiştir. Şekil 9, L2 + L3 + ACL iletme işlevselliğinin (hızlı yakınsama için bir sonraki sıçrama soyutlama ile) uygulanmasının karmaşıklığını göstermektedir. Masadan masaya desteklenen ilkellerin kombinasyonu, desteklenecek çok çeşitli olasılıkların birleşimine yol açar.



Şekil 9:OpenFlow(telli)sürüm 1.0 ilkeleri

Bu, dağıtılmış IP / MPLS modeline kıyasla operatör kontrolünün genişliği açısından çarpıcı bir farktır. (OpenFlow 11-tuple eşleşme alanına sahiptir). Kısa bir olasılık listesi şunları içerir:

- Eşleştirme talimatlarındaki maskeleme yeteneği nedeniyle, ağ IP hedefi yönlendirme davranışını taklit edebilir.
- Hem katman 2 hem de katman 3'te ağ, kaynak / hedef yönlendirme davranışı sergileyebilir.
- OpenFlow'un paket eşleştirme güçlü yönlerinin (şu anda) standartlaştırılmış bir eşdeğeri yoktur, bu da onu dağıtılmış kontrol ortamındaki ilke tabanlı yönlendirme veya diğer eşleşme / öne alma mekanizmalarının yerine çok güçlü bir ikame haline getirmektedir.

Son olarak, değişiklik eyleminin belirtisi var. Orijinal konsept, anahtarın (anahtarın üzerinde çalışan bir uygulama aracılığıyla) NAT veya güvenlik duvarı gibi hizmetleri gerçekleştiren bir servis cihazı gibi davranabilmesidir. Bunun, donanım tabanlı iletme sistemlerinde gerçekleştirilip gerçekleştirilemeyeceği fark edilemez. Bu

özellik, satıcı uygulamasına (desteklenen talimatlar, siparişleri ve satır oranı performansını korumak için bütçelenmiş işlem sayısı) bağlıdır. Bununla birlikte, tel protokolünün 1.3 sürümüne eklenen etiket manipülasyon eylemleriyle, bir openflow kontrollü elemanın bir MPLS LSR (Kaplan, 2008) (veya diğer geleneksel dağıtılmış platform fonksiyonları) gibi entegre platform davranışlarını kolayca taklit etmesi mümkündür.

OpenFlow protokolü, kontrol mesajları, akış eşleme alanları, sayaç kullanımı, istatistikler ve satıcıya özel uzantılar (genel veya özel olabilir) için bir DENEME (Izard, 2018) uzatması (genel veya özel olabilir) üzerinden genişletilebilir.

Tablo girişleri önceliklendirilebilir (üst üste binen girişler olması durumunda) ve zamanlanmış bir bitiş süresine sahip olabilir (bazı durumlarda temizleme işleminden zaman kazanılması ve kontrolör kaybı senaryolarında akışlar için bir üstün bir faydalılık belirlemesi).

OpenFlow, fiziksel, mantık ve rezerve edilmiş port (Foundation, 2012) türlerini destekler. Bu portlar giriş, çıkış veya çift yönlü yapılar olarak kullanılır.
Çok işlevli bir veri hattı oluşturmak için TABLO gerekmektedir (OpenFlow, rastgele GoTo siparişi ile 255 türlenmemiş tabloyu desteklemektedir).
Kalan RESERVED portları önemli (ve ilginç) davranışları mümkün kılar:

i. Yerel

Sadece bir çıkış portu olan bu mantıksal port, OpenFlow uygulamalarına, eleman ana bilgisayar işletim sisteminin erişim portlarına (ve dolayısıyla işlemelerine) izin verir.

ii. Normal

Yalnızca çıkışlı bir bağlantı noktası olan bu mantıksal bağlantı noktası, anahtar geleneksel bir Ethernet anahtarı gibi işlev görmesine izin verir (ilişkili sel / öğrenme davranışları yardımıyla). Protokolün işlevsel özelliklerine göre bu bağlantı noktası yalnızca bir Hibrit anahtar tarafından desteklenir.

iii. Sel

Yalnızca çıkışlı bir bağlantı noktası olan bu mantıksal bağlantı noktası, paketi tüm standart (ayrılmamış) bağlantı noktalarına göndermek için ağ öğesinin çoğaltma motorunu kullanır. AKIŞ, ALL giriş portunun içerdiği için ALL'den (başka bir ayrılmış portu) farklıdır. FLOOD, eleman paketi çoğaltma motorunu kullanır.

iv. Kontrolör

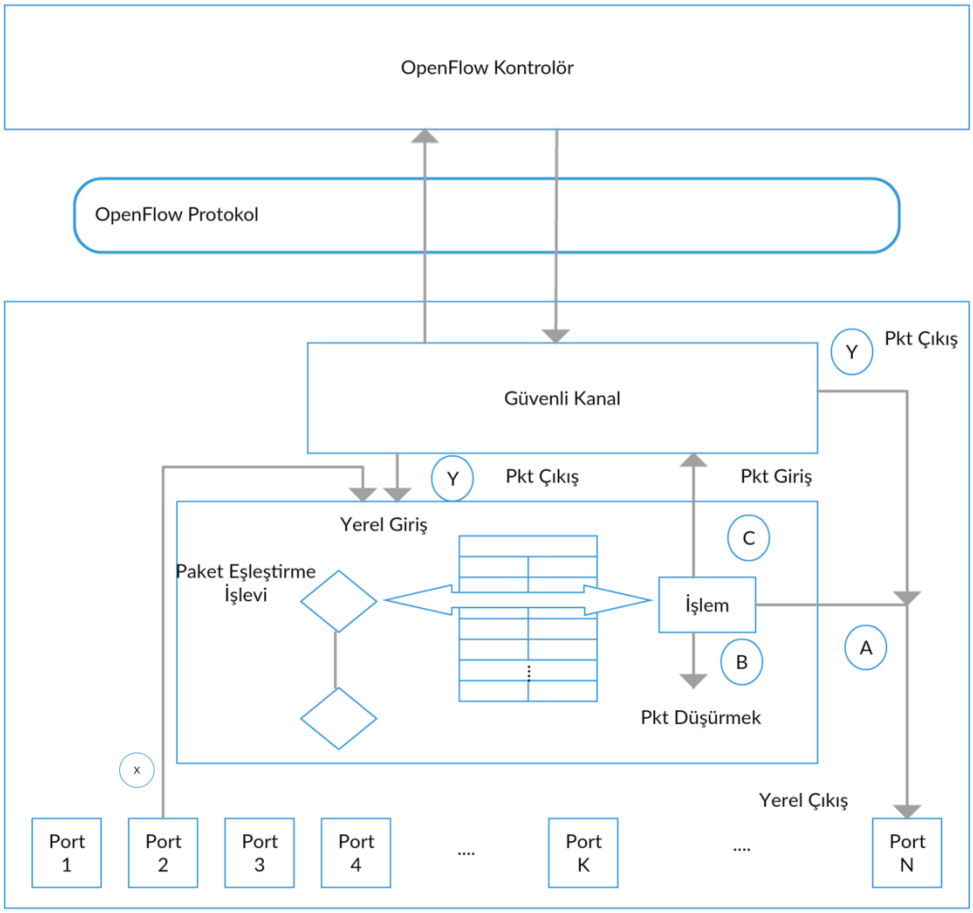
Akış kuralının paketleri (kontrol kanalı üzerinden) bir veri yolundan kontrol cihazına (ve tersine) iletmesini sağlar. Bu, PACKET_IN ve PACKET_OUT davranışını etkinleştirir (Foundation, 2012).

D. OpenFlow Anahtarı

Şekil 10, bir OpenFlow V.1.0 anahtarının temel fonksiyonlarını ve bunun bir kontrol cihazı ile ilişkisini göstermektedir. Bir paket anahtarında bekleneceği gibi, temel fonksiyonun bir bağlantı noktasına (Şekildeki bağlantı noktası 2 üzerindeki X yolu) gelen paketleri alıp başka bir bağlantı noktasından (Şekildeki N bağlantı noktası) yol boyunca gerekli paket değişikliklerini gerçekleştirerek iletmek olduğu görülmektedir. OpenFlow anahtarının benzersiz bir yönü, Şekil 11'de gösterilen paket eşleştirme işlevinde şekillendirilmiştir. Bitişik tablo bir akış tablosudur. Şekil 10'daki geniş, gri, çift ok karar mantığında başlar, bu tablodaki belirli bir girdiye olan bir eşleşmeyi gösterir ve bu anda eşleşen paketi sağdaki bir işlem kutusuna yönlendirir. Bu işlem kutusu, gelen paketin elden çıkarılması için üç temel seçeneğe sahiptir (Göransson & Black, 2014):

- A. Paketi önce büyük olasılıkla belirli başlık alanlarını değiştirerek yerel bir bağlantı noktasına iletir.
- B. Paketi bırakır.
- C. Paketi kontrol ünitesine iletir.

Bu üç temel paket yolu, Şekil 10'da gösterilmektedir. C yolu kullanılması durumunda, paket kontrol cihazına şekilde gösterilen güvenli kanal üzerinden iletilir. Anahtara verilecek bir kontrol mesajı veya bir veri paketi olması durumunda, kontrolör de bu aynı güvenli kanalı ters yönde kullanır. Denetleyici, anahtardan dışarı iletilecek bir veri paketi olduğunda, OpenFlow PACKET_OUT mesajını kullanır. Şekil 10'de, kontrol cihazından gelen bu tür bir veri paketinin, her ikisi de Y ile gösterilen OpenFlow mantığı boyunca iki farklı yol alabildiği görülmektedir. En sağdaki durumda, kontrol cihazı doğrudan çıkış portunu belirler ve paket bu N portuna geçirilir. En sağdaki durumda, kontrol cihazı doğrudan çıkış portunu belirler ve paket örnekteki N portuna geçirilir (Göransson & Black, 2014).

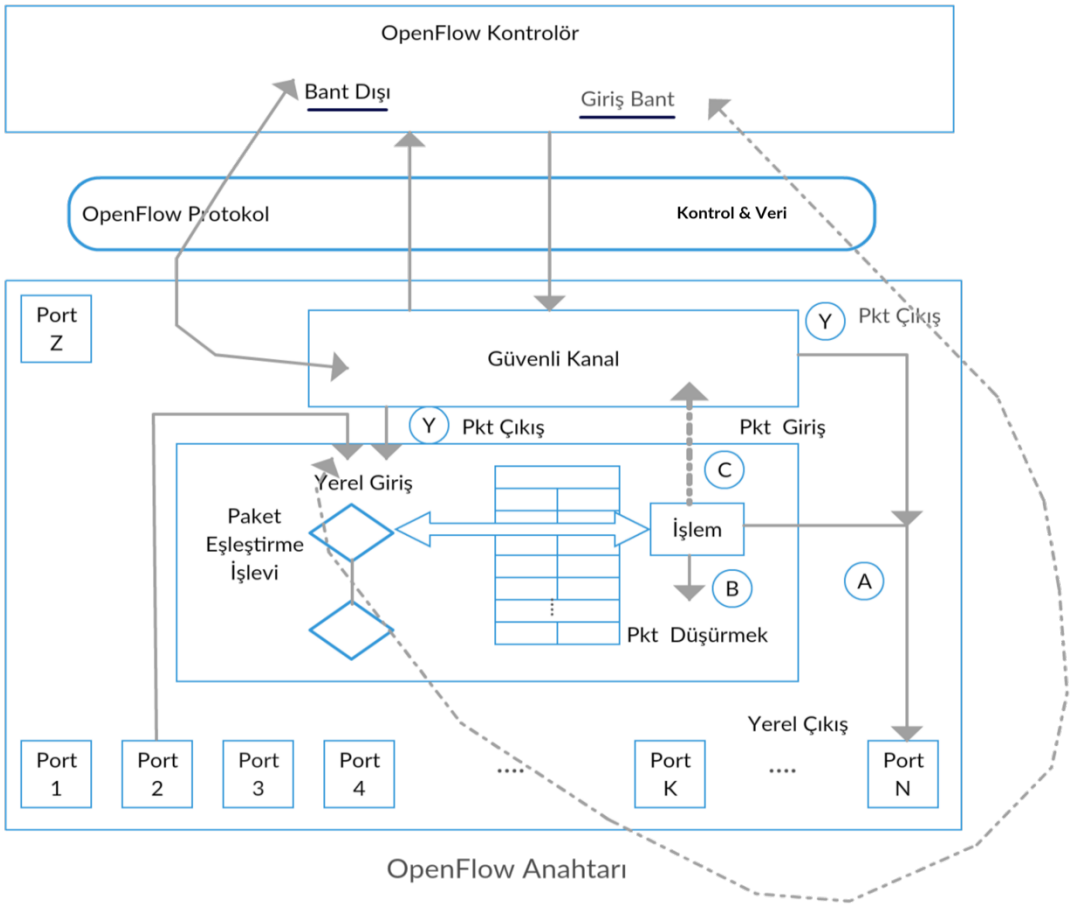


Şekil 10 :OpenFlow V.1.0 Anahtarı

En soldaki Y yolunda, kontrol cihazı yönlendirme kararını paket eşleştirme mantığına ertelemek istediğini belirtir. Belirli bir OpenFlow anahtarı uygulaması, ya sadece OpenFlow ya da OpenFlow hibritidir. Yalnızca OpenFlow anahtarı, paketleri yalnızca yukarıda açıklanan OpenFlow mantığına göre ileten anahtardır. Bir OpenFlow hibrid, paketleri bir Ethernet anahtarı veya IP yönlendiricisi olarak eski moduna da değiştirebilen bir anahtardır. Hibrit kasa, tamamen bağımsız bir geleneksel anahtarın yanında bulunan bir OpenFlow anahtarı olarak görülebilir. Böyle bir melez anahtar, paketleri OpenFlow işlemesine veya geleneksel paket işlemeye yönlendiren bir ön işleme sınıflandırma mekanizması gerektirir. Melez anahtarların saf OpenFlow uygulamalarına geçiş sırasında norm olması muhtemeldir *(Göransson & Black, 2014)*.

1. Kontrolör-Anahtar Güvenli Kanalı

Güvenli kanal, OpenFlow denetleyicisi ile OpenFlow cihazı arasındaki iletişim için kullanılan yoldur. Genel olarak, bu iletişim şifrelenmemiş TCP bağlantılarına izin verilmesine rağmen, TLS (Dierks & Rescorla, 2008) tabanlı asimetrik şifreleme ile güvence altına alınmıştır. Bu bağlantılar bant içi veya bant dışı olabilir. Şekil 11, güvenli kanalın bu iki varyantını göstermektedir. Bant dışı örnekte, güvenli kanal bağlantısının anahtara, OpenFlow veri düzlemi tarafından anahtarlanmayan Z portundan girdiği görülmektedir. Bazı eski ağ yığını, OpenFlow mesajlarını güvenli kanal üzerinden tüm OpenFlow mesajlarının ayrıştırıldığı ve işlendiği anahtardaki güvenli kanal işlemine iletir.



Şekil 11:OpenFlow Kontrolör-Anahtar Güvenli Kanal

Bu nedenle, bant dışı güvenli kanal yalnızca bir OpenFlow hibrit anahtar durumunda geçerlidir.

Bant içi örnekte, OpenFlow veri düzleminin bir parçası olan K bağlantı noktasındaki denetleyiciden gelen OpenFlow mesajları görülmektedir. Bu durumda, bu paketler şekilde gösterilen OpenFlow paket eşleştirme mantığı ile ele alınacaktır. Akış

tabloları, bu OpenFlow trafiğinin LOCAL sanal portuna iletilmesi ve böylece iletilerin güvenli kanal işlemine geçirilmesiyle sonuçlanacak şekilde oluşturulmuş olacaktır. Kontrolör ve kontrol ettiği tüm anahtarlar tamamen bir veri merkezi gibi sıkı kontrol edilen bir ortama yerleştirildiğinde, kanalı korumak için TLS tabanlı şifrelemeyi kullanmamanın akıllıca olabileceğini dikkate almak gerekir. Bunun nedeni, bu tür bir güvenlik sistemi kullanılarak bir performans ek yükünün gerçekleşmesidir ve bunun gerekli olmadığı durumlarda, bu performans azalmasının olmaması tercih edilir (Göransson & Black, 2014).

IV. YÖNTEM BİLİM

A. VirtualBox

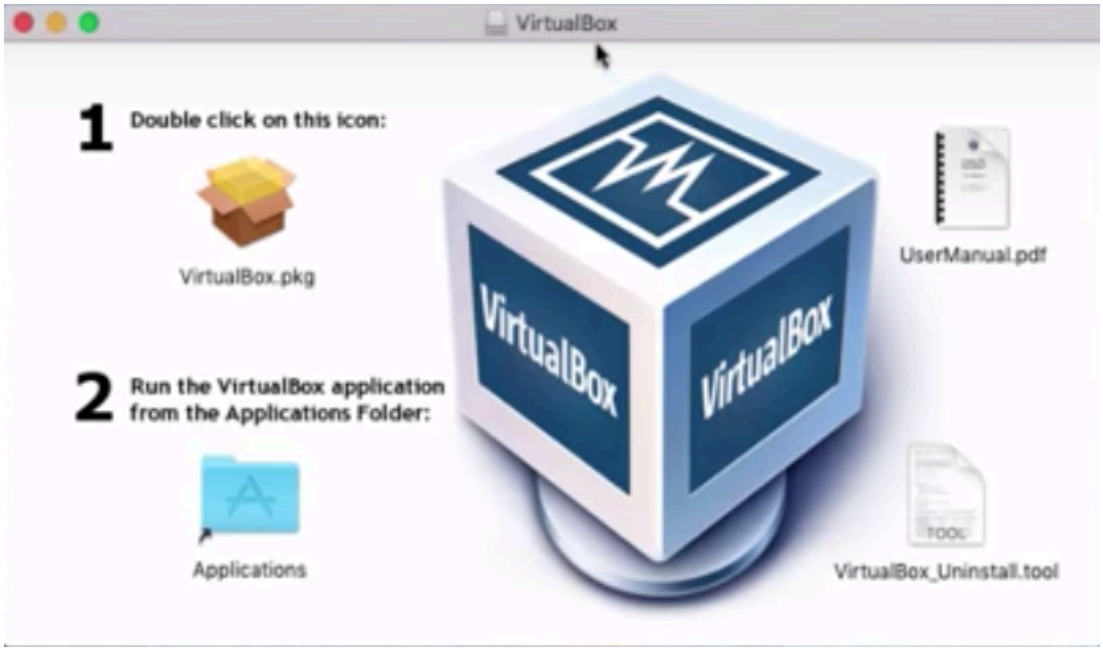
Oracle VirtualBox bir platformlar arası sanallaştırma uygulamasıdır. Windows, Mac, Linux veya Solaris işletim sistemleri kullanılmaktaysa, mevcut Intel veya AMD tabanlı bilgisayarlara kurulabilir. VirtualBox, ana bilgisayar işletim sisteminin bir penceresinde bir "konuk" işletim sistemi (sanal makine) oluşturabilir ve çalıştırabilir. Sanal makine, ana bilgisayar işletim sistemindeki zararlı değişiklikleri riske atmadan yeni yazılımı denemek için bağımsız bir ortam sunar (technous.net, 2018).

B. VirtualBox Kurulumu

Geçerli işletim sistemininin ikili dosyasını indirmek için VirtualBox web sitesine şekil 12'de gösterildiği gibi giriş yapılır. Ana makine Mac üzerinde çalıştığından, Mac ana bilgisayarlarından Mac seçeneği seçilmelidir.

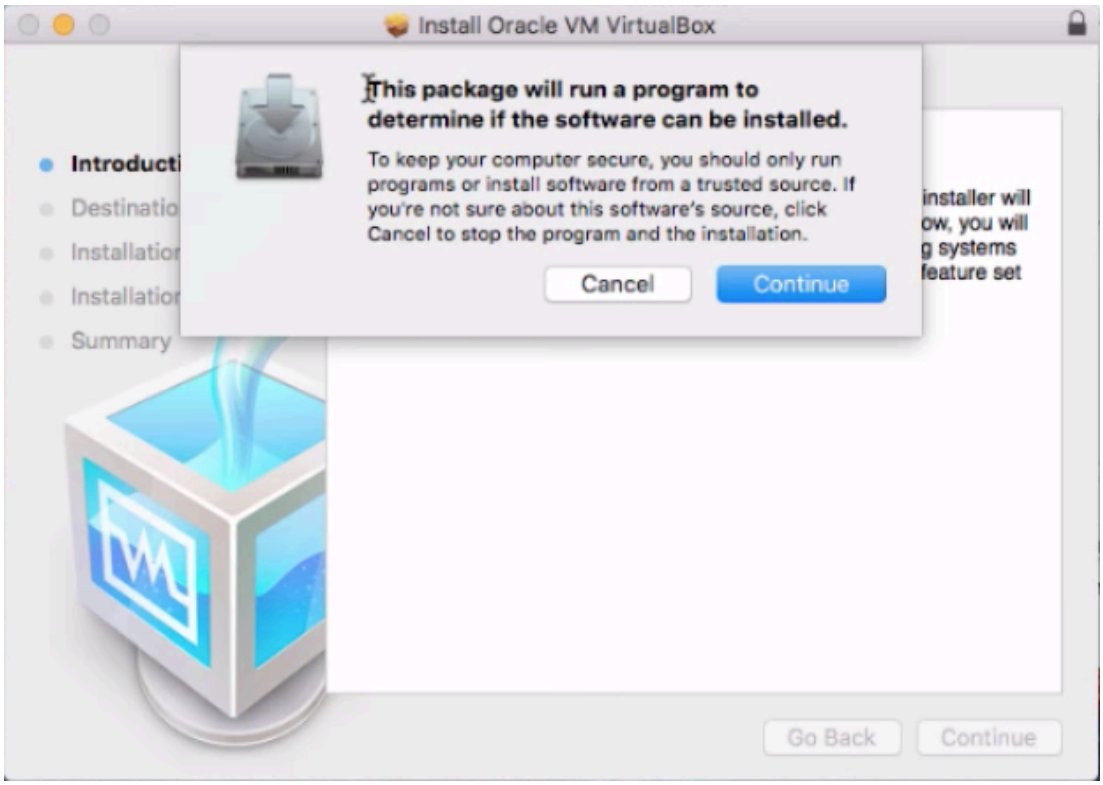
Şekil 12::VirtualBox Indirmek için

Daha sonra indirilen virtualbox dosyası şekil 13'te gösterildiği gibi üzerine iki kere tıklanarak çalıştırılmaktadır.



Şekil 13: VirtualBox Kurulumuna Başlamak

İlk aşamada sorulan soruya şekil 14'te görüldüğü gibi 'continue' seçimi seçilerek devam edilmelidir.

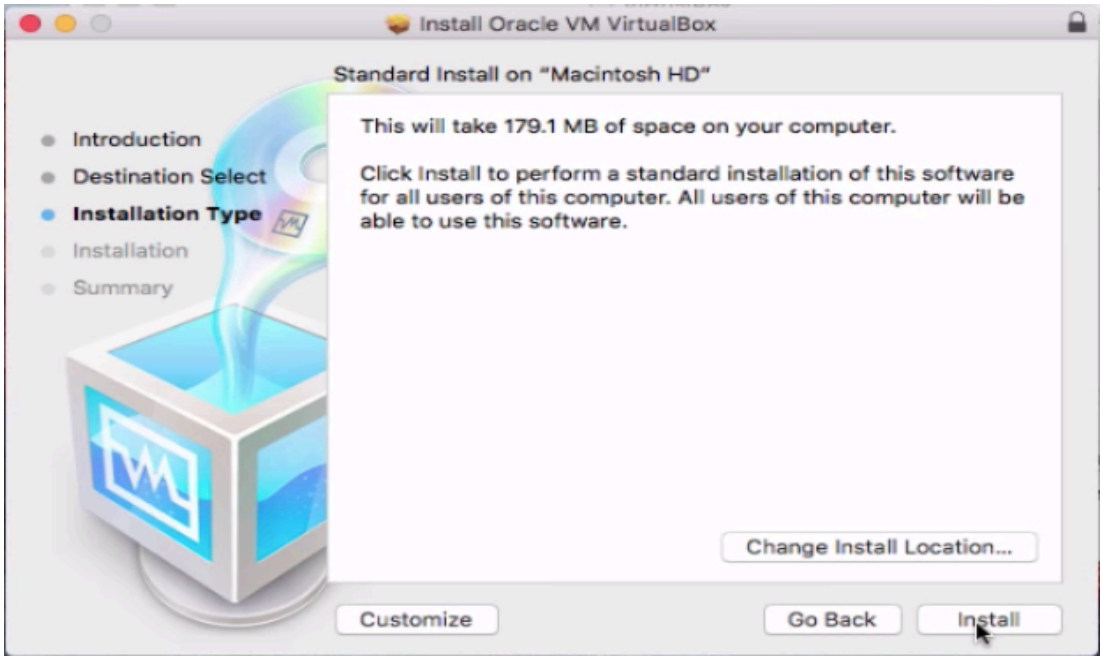


Şekil 14:ilk aşamada sorulan soru

Şekil 15'te gösterildiği gibi 'Introduction' bölümünde de 'continue' seçilerek devam edilmiştir.

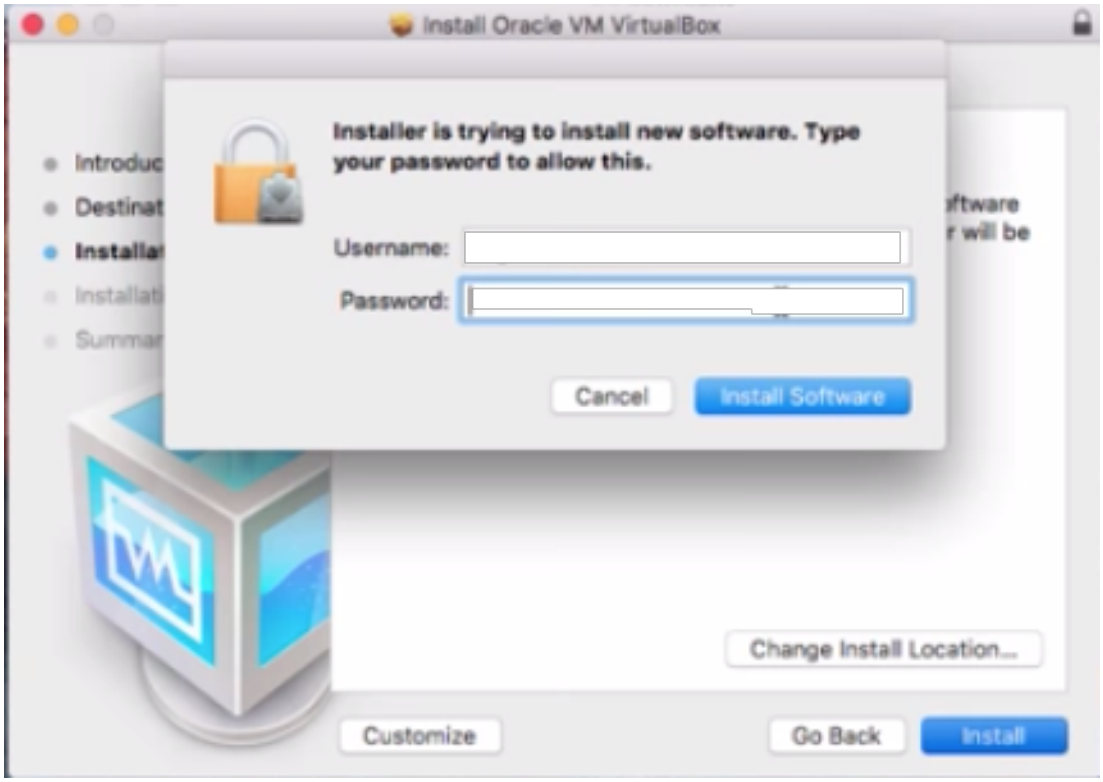
Şekil 15:Tanıtım bölümünde devam tuşunun seçilmesi

Şekil 16'da gösterildiği gibi standart kurulumu Mac ta düzenlemek için 'install' tuşu seçilmelidir.



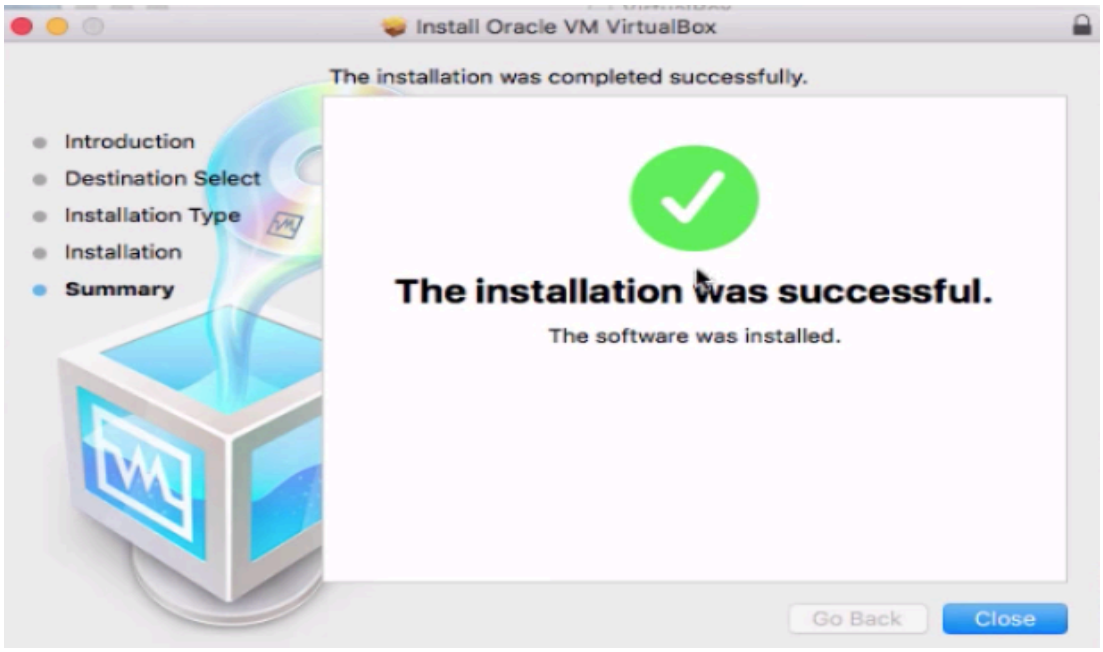
Şekil 16:VirtualBox'ın standart kurulumunun başlaması

Kuruluma devam edebilmek için kullanıcı, şekil 17'de gösterildiği gibi sisteme şifresini girmelidir.



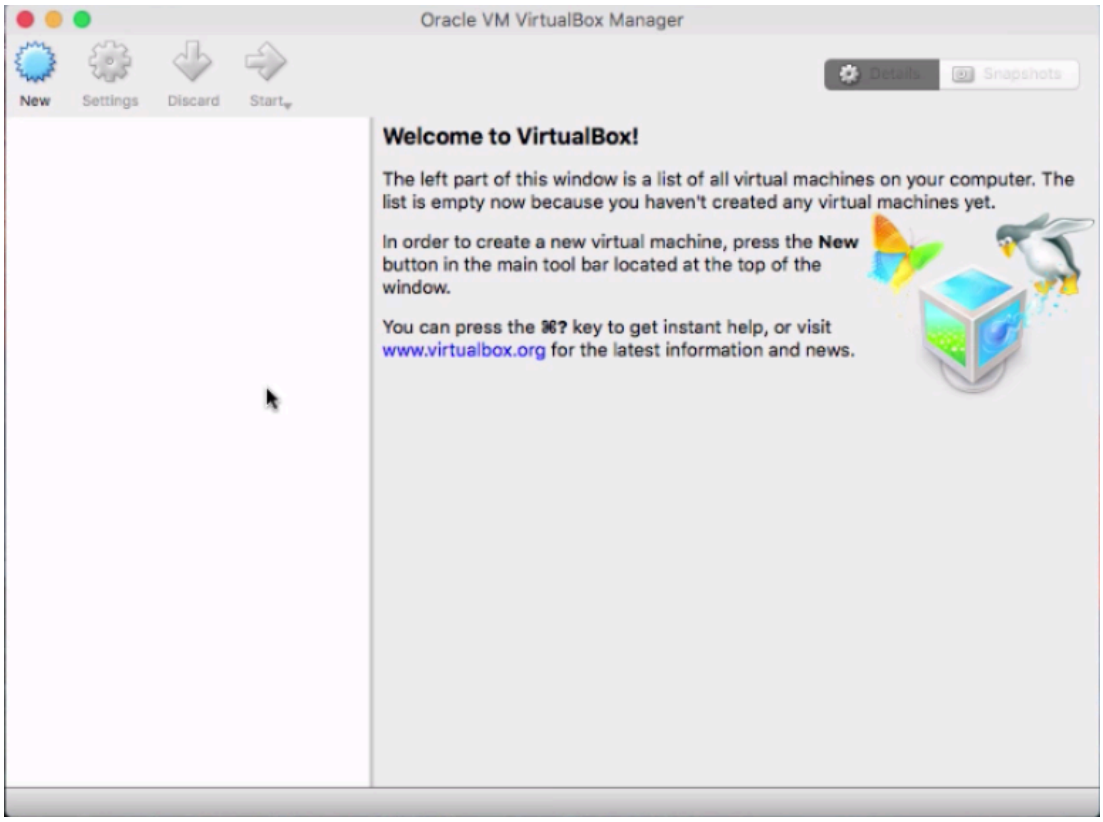
Şekil 17:Kullanıcı ve sistemin şifresinin girilmesi

Sonra şekil 18'de gösterildiği gibi kurulum bitene kadar beklenmelidir.



Şekil 18:VirtualBox kurulumunun tamamlanması

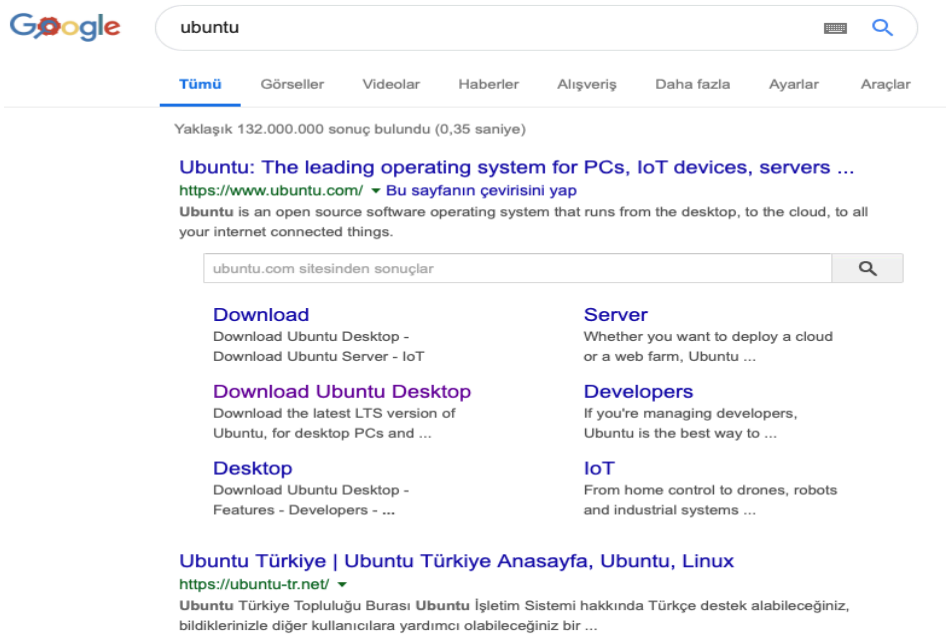
Kurulum bittikten sonra Virtualbox, şekil 19'da gösterildiği gibi Virtualbox Application veya Launchpad sekmesinden açılmaktadır (technous.net, 2018).



Şekil 19:VirtualBox'ın başlatılması

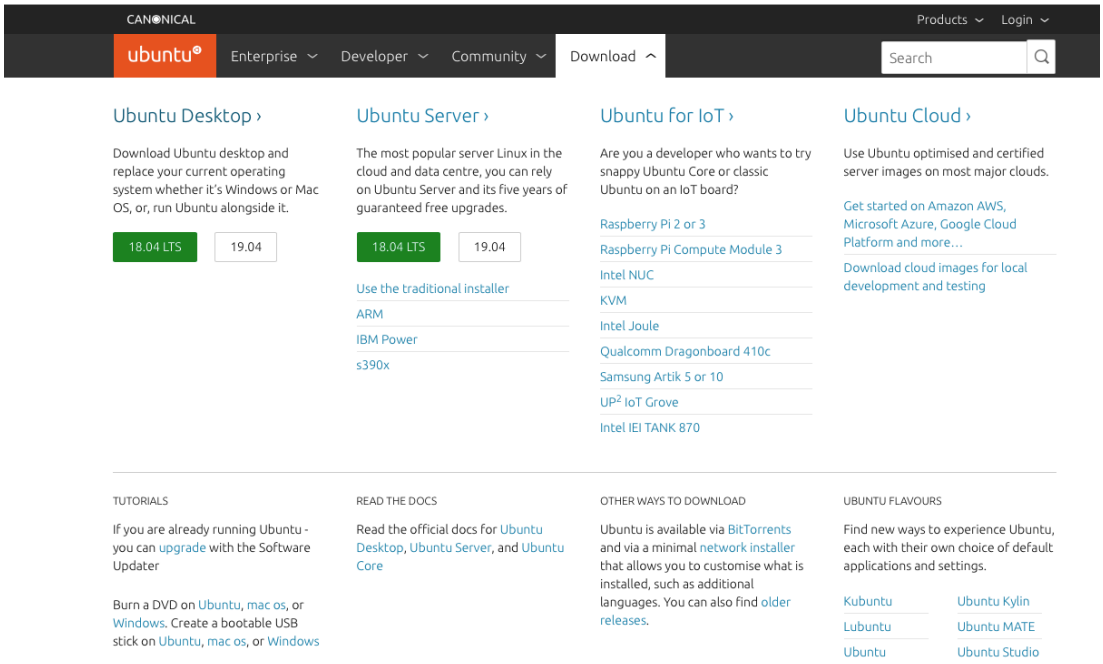
1. Sanal Makine Oluşturmak

Sanala makine oluşturmak için öncelikle, şekil 20 ‘da gösterildiği gibi Google arama moturunda ‘Ubuntu’ yazarak, ‘Ubuntu’ sitesine ulaşılmalıdır.



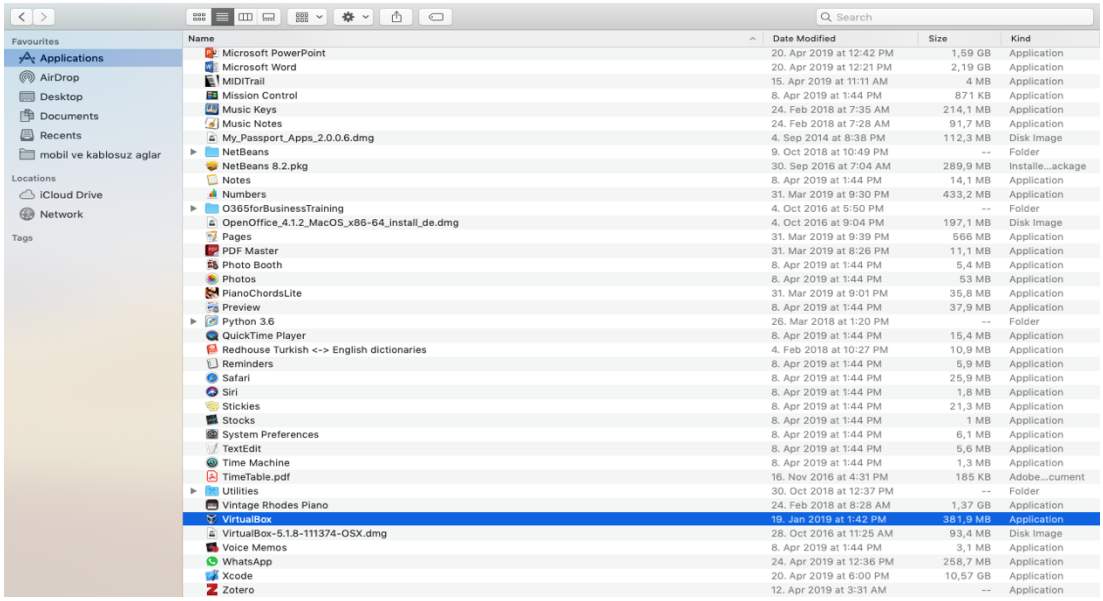
Şekil 20:Ubuntu'nun nasıl indirileceğini anlatan görsel

Siteden şekil 21'de görüldüğü gibi 'download' sekmesinden 'Ubuntu Desktop' seçilmiştir.



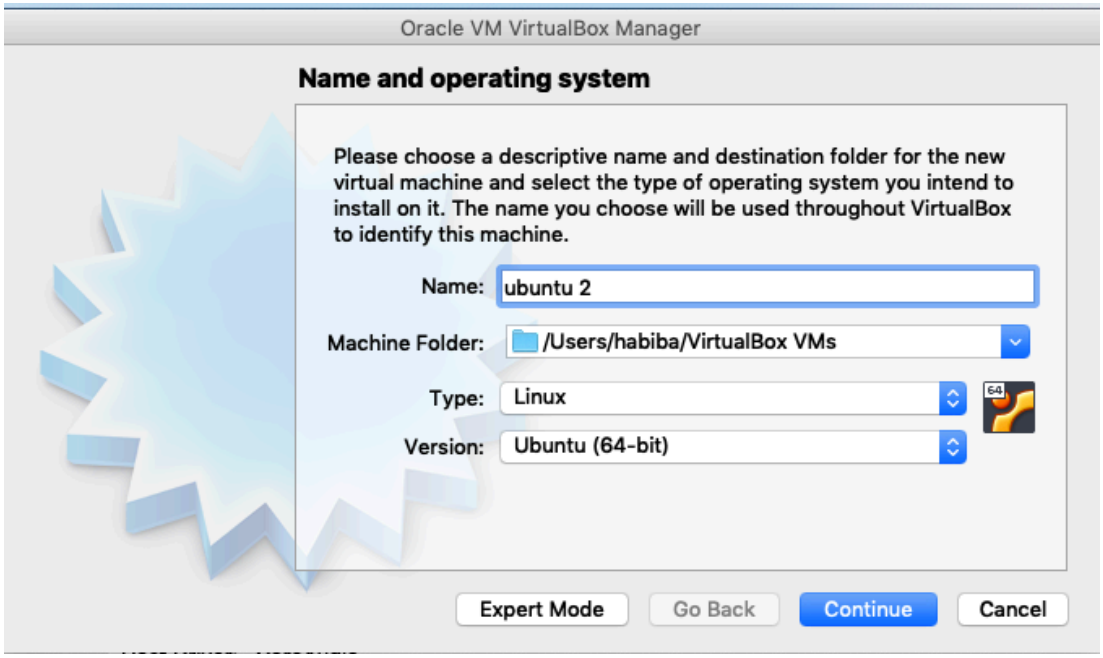
Şekil 21:Ubuntu işletim sistemini indirmek

Şekil 22'de görüldüğü gibi 'application' sekmesinde açılan listeden 'virtual box' seçilerek açılmıştır.



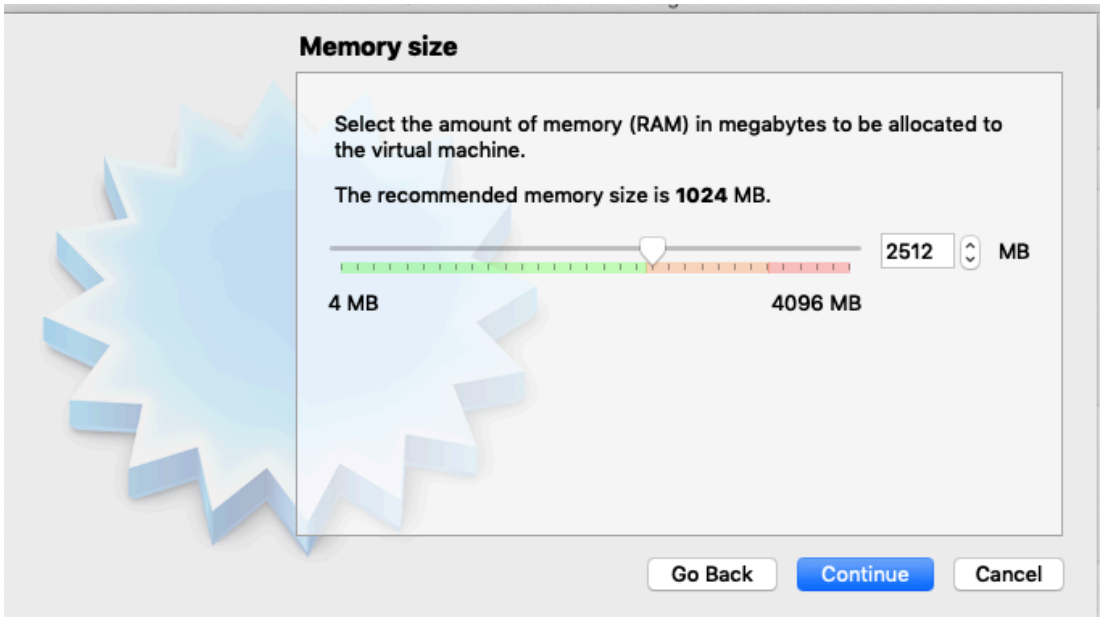
Şekil 22:VirtualBox'ın nasıl açıldığını göstermektedir

Şekil 23'te gösterildiği gibi 'New' sekmesine tıkladıktan sonra yeni bir pencere açılmıştır. Açılan pencerede yeni sanal makine için isim kısmına 'ubuntu 2' yazıldıktan sonra sistem türü otomatik 'Linux' ve versiyon seçeneği de 'ubuntu 64-bit' olarak seçilerek devam edilmiştir.



Şekil 23:Sanal makineye bir isim verilmesi

Daha sonra şekil 24'te gösterildiği gibi hafıza boyutu 2512MB olarak atanıp devam edilmiştir.



Şekil 24:Hafıza boyutu

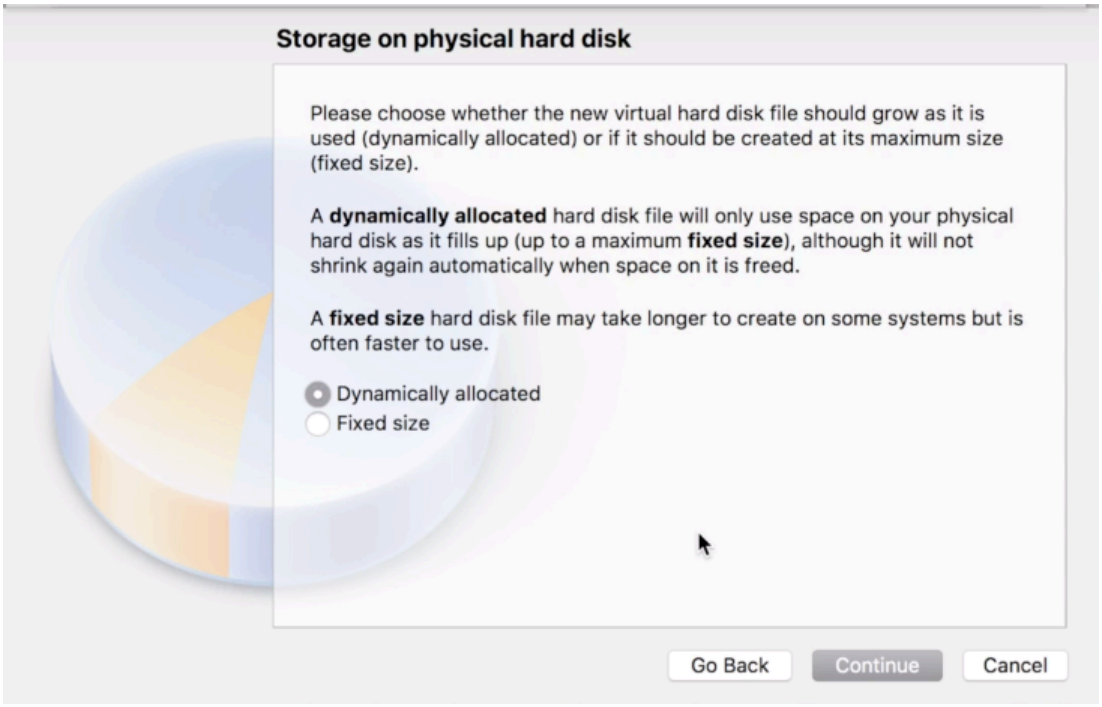
Şekil 25'te gösterildiği gibi bir sonraki adımda sabit disk eklenmesi hakkında 'Do not add a virtual hard disk' seçeneği işaretlendikten sonra 'create' düğmesi tıklanmıştır.

Şekil 25:Sanal sabit disk eklenmesi

Şekil 26'da gösterildiği gibi sonraki adımda sabit diskin dosya türü VDI (VirtualBox Disk image) işaretlendikten sonra devam edilmiştir.

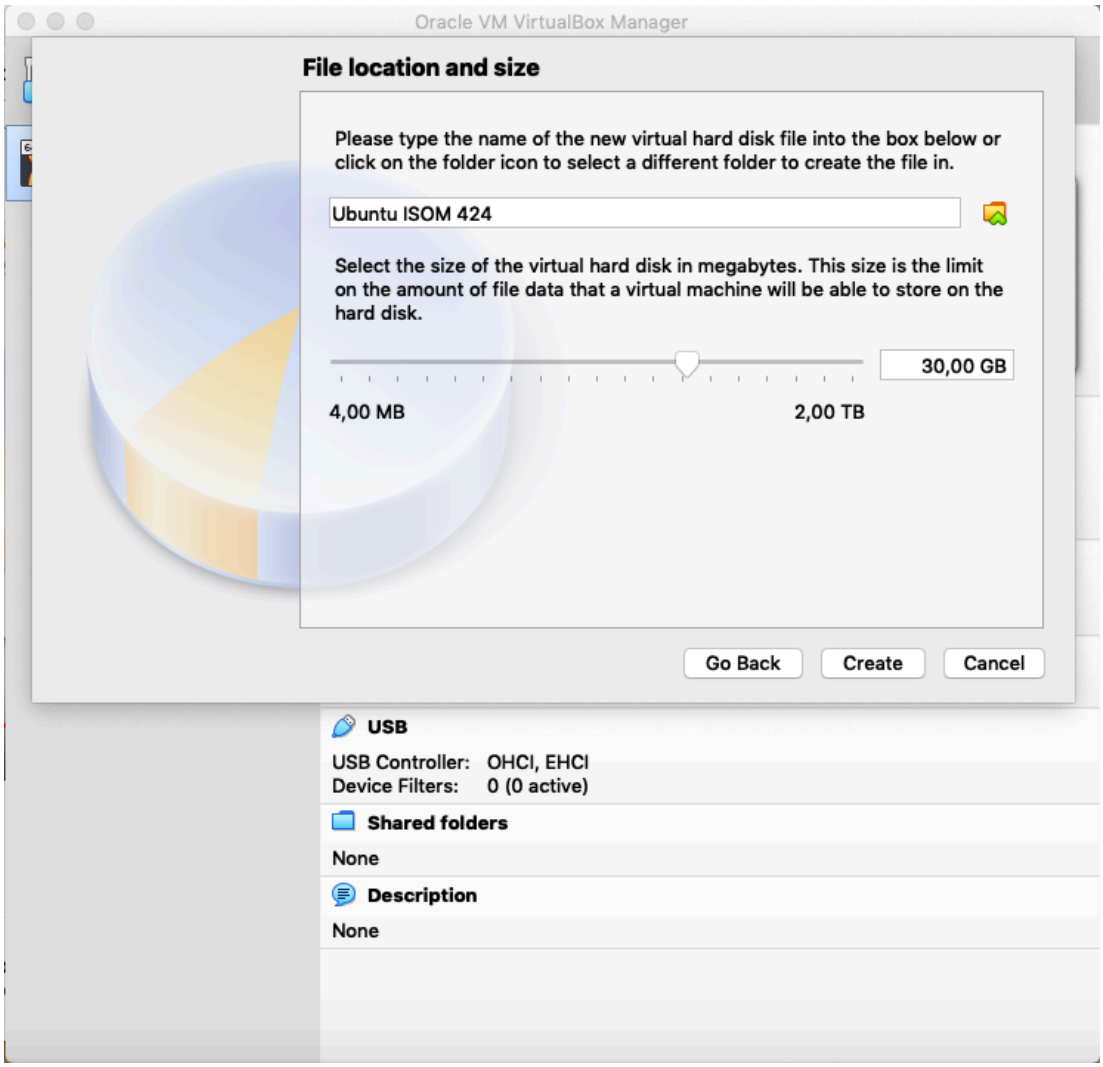
Şekil 26:Sabit disk dosya türü

Şekil 27'de gösterildiği gibi fiziksel sabit diskte depolama 'Dynamically allocated' olarak seçilmiştir.



Şekil 27:Fiziksel sabit diskte depolama

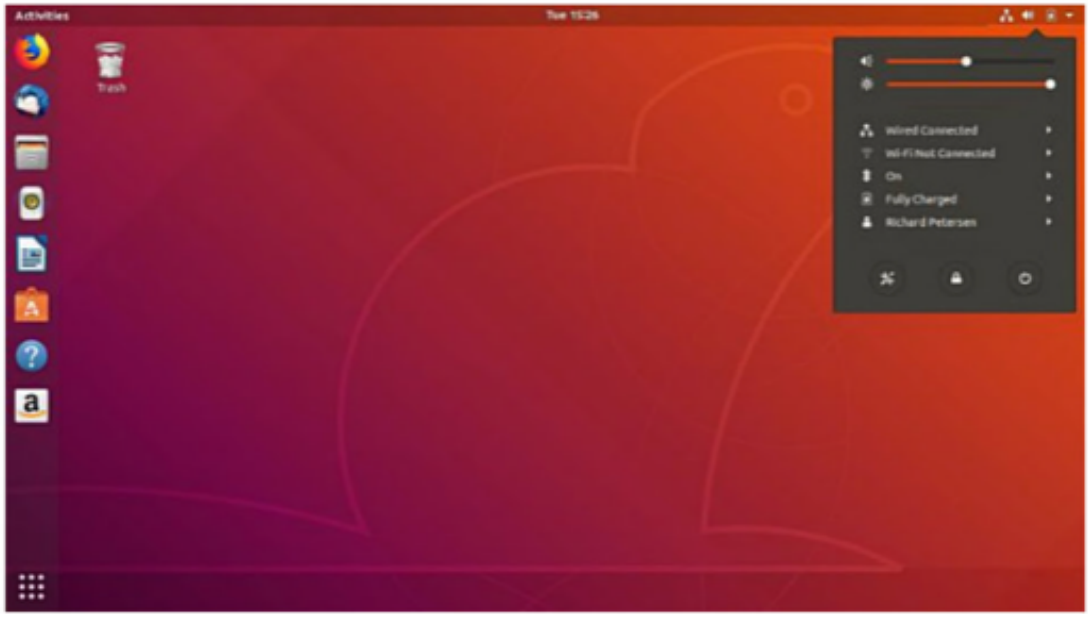
Şekil 28'de gösterildiği gibi dosya konumu ve boyutu bölümünde ise 30GB olarak atanmış ve 'create' tuşu ile sanal bir makine oluşturulmuştur.



Şekil 28:Dosya konum ve boyutu

1. Ubuntu 18.04 LTS

Ubuntu 18.04, birçok yeni özelliğin yanı sıra çok sayıda küçük değişiklik getirmiştir. Linux 4.4 çekirdeğini temel alan uzun vadeli bir destek sürümüdür. Ubuntu depoları, Ubuntu masaüstü (GNOME), Kubuntu (KDE), MATE ubuntu ve GNOME Ubuntu (GNOME3) dahil olmak üzere birçok büyük masaüstünü desteklemektedir. Ubuntu'yu, aynı depoların farklı masaüstlerini desteklediği çoklu masaüstü sistemi olarak düşünülebilir. Şekil 29'da gösterildiği gibi Ubuntu masaüstünde GNOME kullanıcı ara yüzü, bir Dock, dash ve gösterge menülerine sahiptir. Ubuntu kurulmu sadece ubuntu masaüstü olarak adlandırılan GNOME, GNOME 3.18 tabanlı varsayılan masaüstüdür (Petersen, 2018).



Şekil 29: Ubuntu 18.04'un Masaüstü Görünümü

Ubuntu 18.04 ayrıca minimal kurulum özelliğine sahiptir. tam bir uygulama kümesi yerine yalnızca web tarayıcısı ve birkaç yardımcı program yüklenir. Şekil 30 ‘da gösterildiği gibi "Güncellemeler ve diğer yazılımlar" ekranındaki kurulum işlemleri sırasında minimum kurulum seçenekleri seçilebilir. "Hangi uygulamalarla başlamak istersiniz?" başlığı altında, iki seçenek vardır: Normal ve minimum kurulum. normal yükleme seçeneği varsayılan olarak seçilir. En düşük seviyede, en az yükleme seçeneğini seçilmiştir (Petersen, 2018).

Şekil 30:Ubuntu 18.04 Masaüstü minimum kurulum

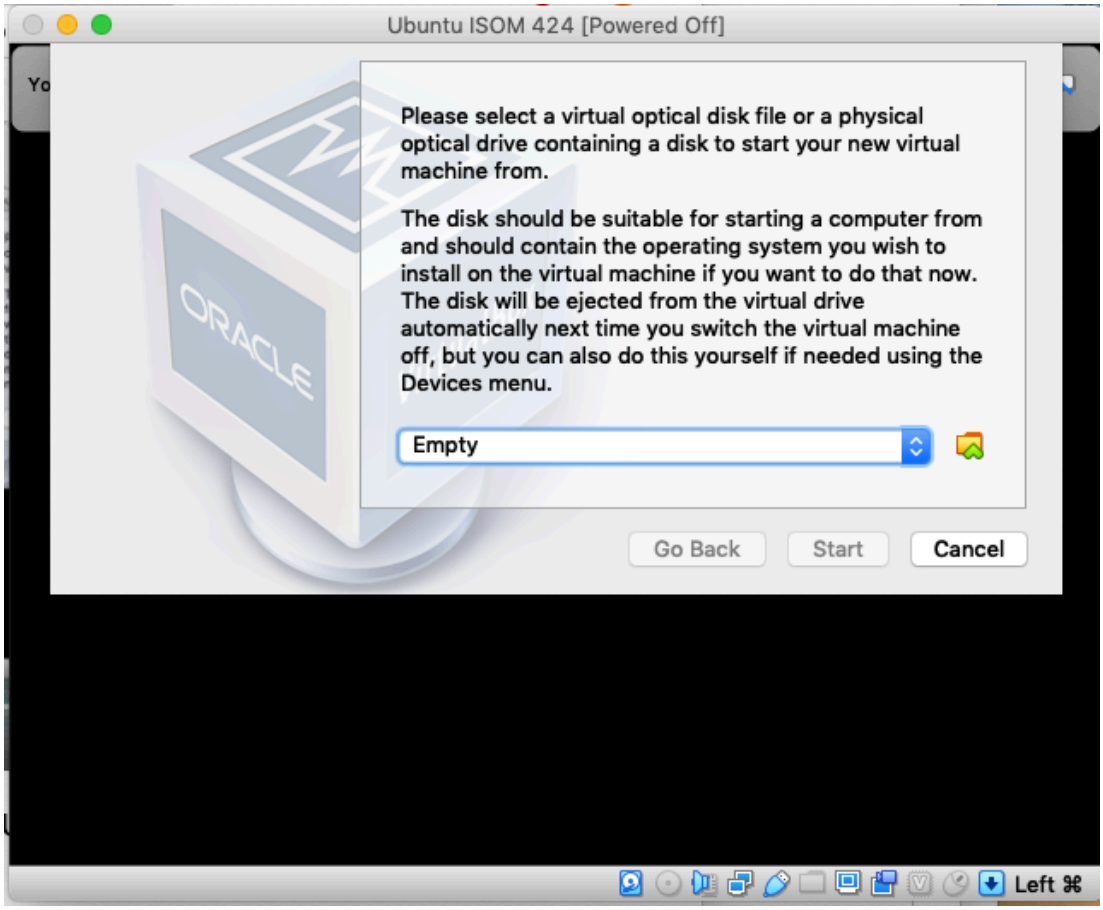
a. Ubuntu 18.04'ün en önemli değişiklikleri

18.04 sürümü için birkaç önemli değişiklik yapılmıştır. Ubuntu masaüstünde artık unity yerine GNOME kullanılmaktadır. Bu nedenle artık bir GNOME'u andırmamaktadır. GNOME'un daha yeni bir sürümü için gnome-session, GNOME'un sade bir versiyonu için vanilla-gnome-desktop'ı kurabilirsiniz. Ubuntu masaüstünde, bütünlük üzerine başlatıcıya benzer şekilde çalışan bir takma ad uzantısı uygulanmaktadır. Ubuntu ağ cihazlarını yönetmek için netplan.io kullanır. İfdown, ifup komutları kullanılmamakla birlikte istenildiği zaman yüklenebilmektedir. Desteklenen SMB protokolü, pencere değişikliklerini onaylamak için şu anda version 2.1 veya daha üst sürümleri kullanılmaktadır. Bu, masaüstü ağ taramasının çalışmayabileceği anlamına gelir. Masaüstünde, belirli bir ana bilgisayar kullanabilir ve bunun yerine sunucu işlemlerine bağlanılarak erişim yöntemleri paylaşılabilir. Bir takas dosyası yerine artık bir takas bellek bölümü kullanılmakla birlikte istenildiğinde bir takas bölümünü de kullanabilir. Önceki bir sürümden yeni bir yükseltme yapılmaktaysa, eski takas bölümünü hala kullanılabilmektedir. Ubuntu masaüstünde sadece web tarayıcısı ve standart araçların yüklendiği minimal kurulum seçeneği mevcuttur. Sonrasında arzu edilen uygulamalar yüklenilebilmektedir. Yeni Wayland ekran sunucusu, her giriş yapıldığında bir seçenek olarak sunulur (Wayland'da Ubuntu). Wayland'ın halihazırda stabil versiyonunun olmadığı uyarısı yapmakla birlikte, gelecek sürümlerde varsayılan olacağı kesindir. Şu anda varsayılan sunucu eski Xorg sunucusudur. Ubuntu, sunucu ve masaüstü sürümleri için 32 bit versiyonlarını artık sunmamaktadır. Fakat Kubuntu, ubuntu-mate, Xubuntu ve budgie ubuntu gibi diğer Ubuntu türleri 32bit versiyonları sunmaktadırlar. Ubuntu sunucusu ağ bağlantılarını uygulamak için systemd-networkd kullanmaktadır. Ubuntu masaüstü versiyonunda halihazırda bir ağ yöneticisi kullanılmaktadır. Her iki ağ yöneticisi de karşılıklı yer değiştirilerek kullanılabilmektedir. Ağ zaman sunucusu olarak ntpd yerine chrony kullanılır. Canonical, güvenlik güncellemeleri için canlı bir düzeltme eki hizmeti sunmaktadır, böylelikle güncellemenin etkinleşmesi için sistemin yeniden başlatılması gerekmemektedir. Bu, masaüstünden ziyade sürekli çalışması gereken sunucular için bir sorundur (Petersen, 2018).

Ek depo Ubuntu yazılımına entegre edilmiştir. Kullanılacak kanal seçilebilmektedir. Masaüstü temalarını ve ikon stillerini değiştirmek için GNOME tweaks'ı kurulmalı ve görünüm sekmesi kullanılmalıdır. Gelecekteki varsayılan

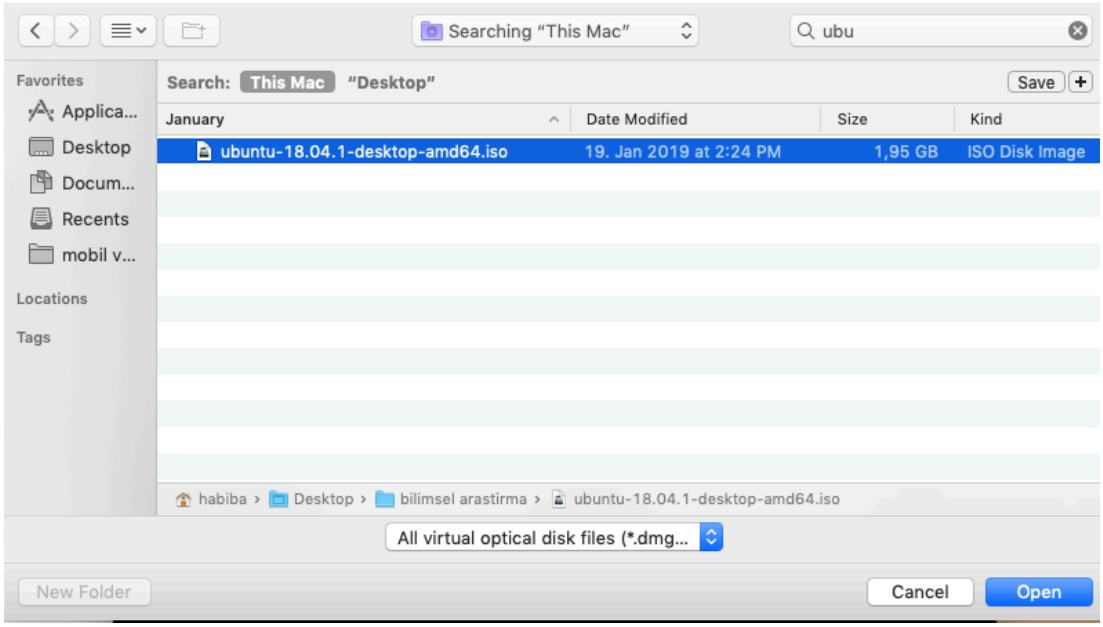
masaüstü teması topluluk temasıdır. Ubuntu yazılımına communitheme olarak indirilebilir (Petersen, 2018).

Şekil 31'de gösterildiği gibi Sanal makine oluşturduktan sonra sanal makine üzerine tıklanmaktadır.



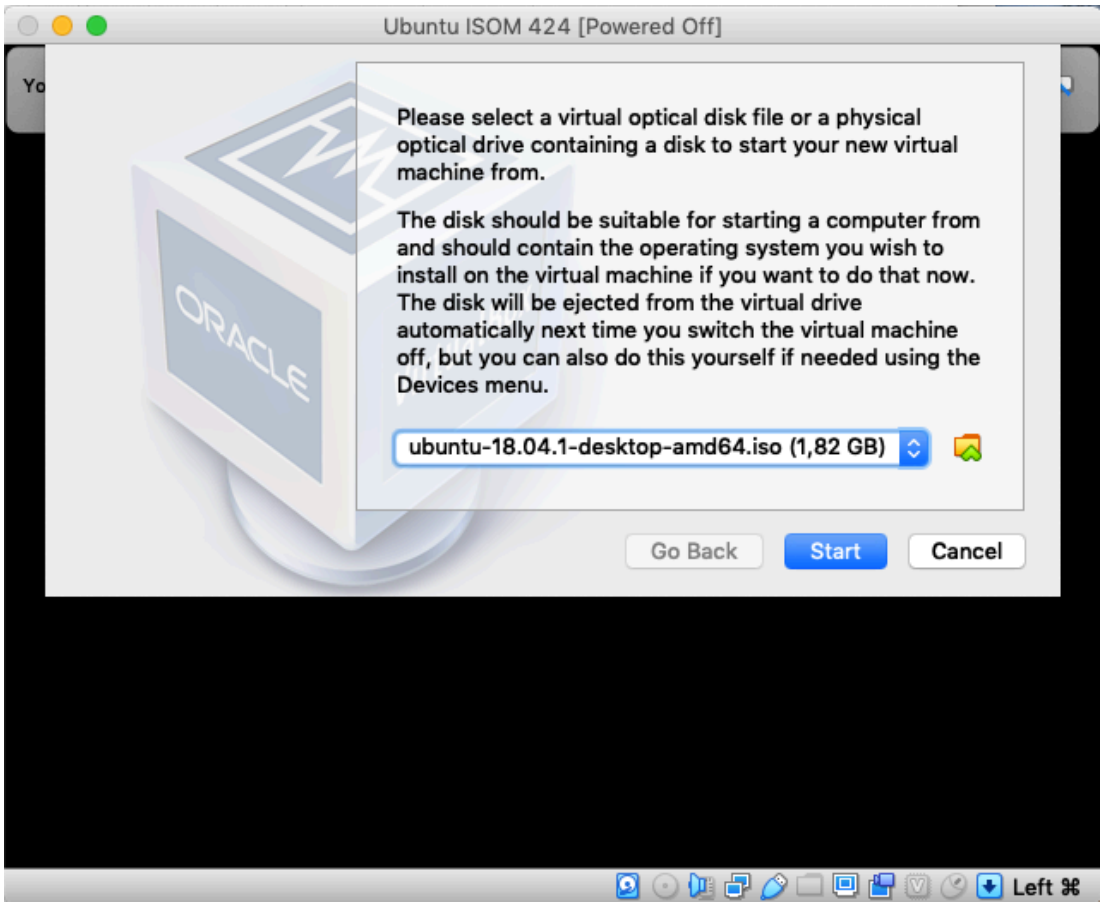
Şekil 31:İşletim sisteminin yüklenmesinden önceki aşama

Şekil 32'de gösterildiği gibi işletim sisteminin görüntüsünü sanal makineye aç diyerek yüklenmektedir.



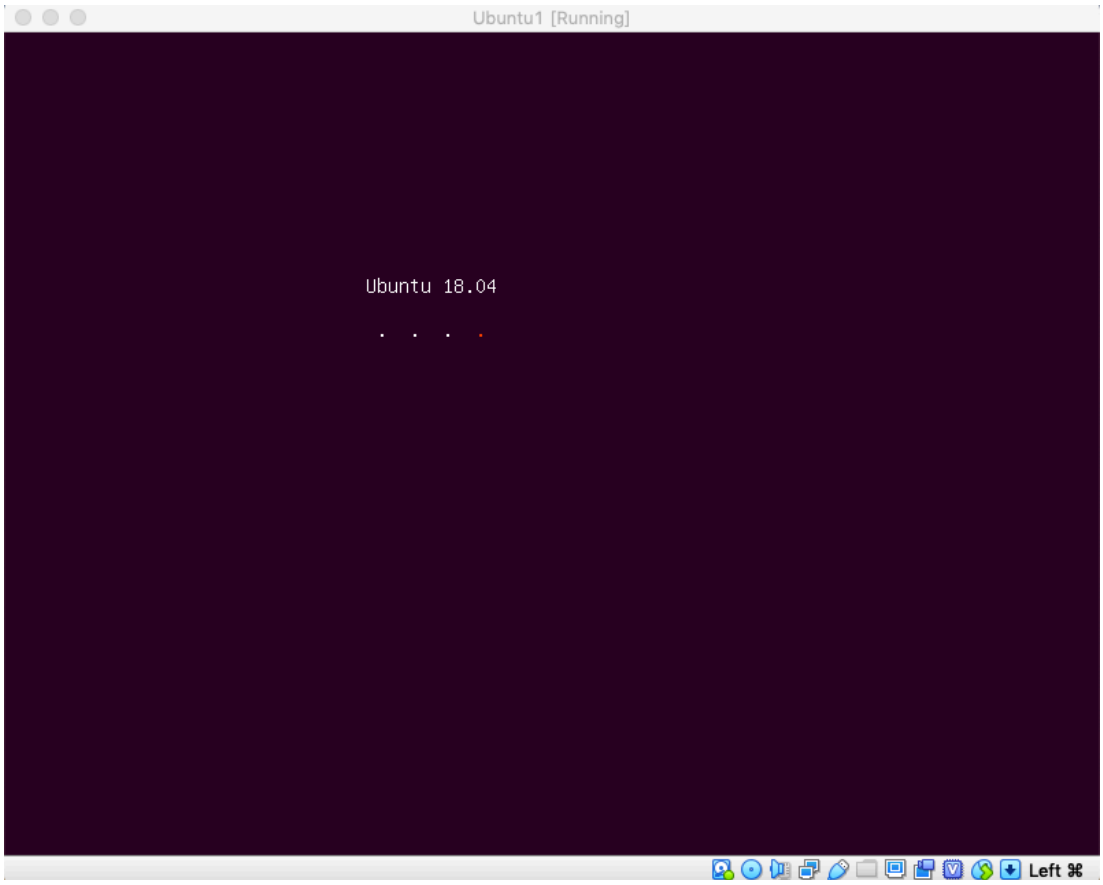
Şekil 32:Ubuntu'nun görüntüsünü yüklemek

Ve daha sonra şekil 33'te gösterildiği gibi start düğmesi seçilmektedir.



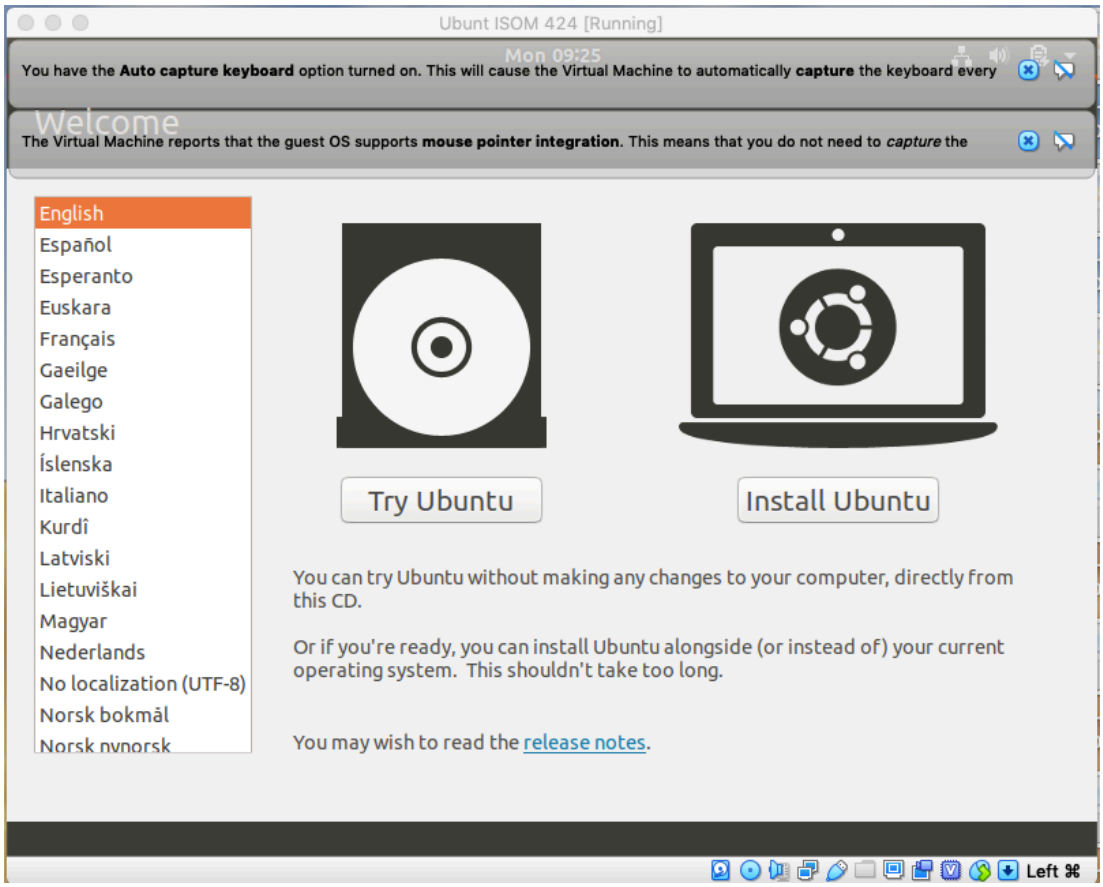
Şekil 33:İşletim sisteminin görüntüsünü seçtikten sonraki aşama

Sonra şekil 34'te gösterildiği gibi ubuntu'nun başlaması için bir süre beklenmektedir.



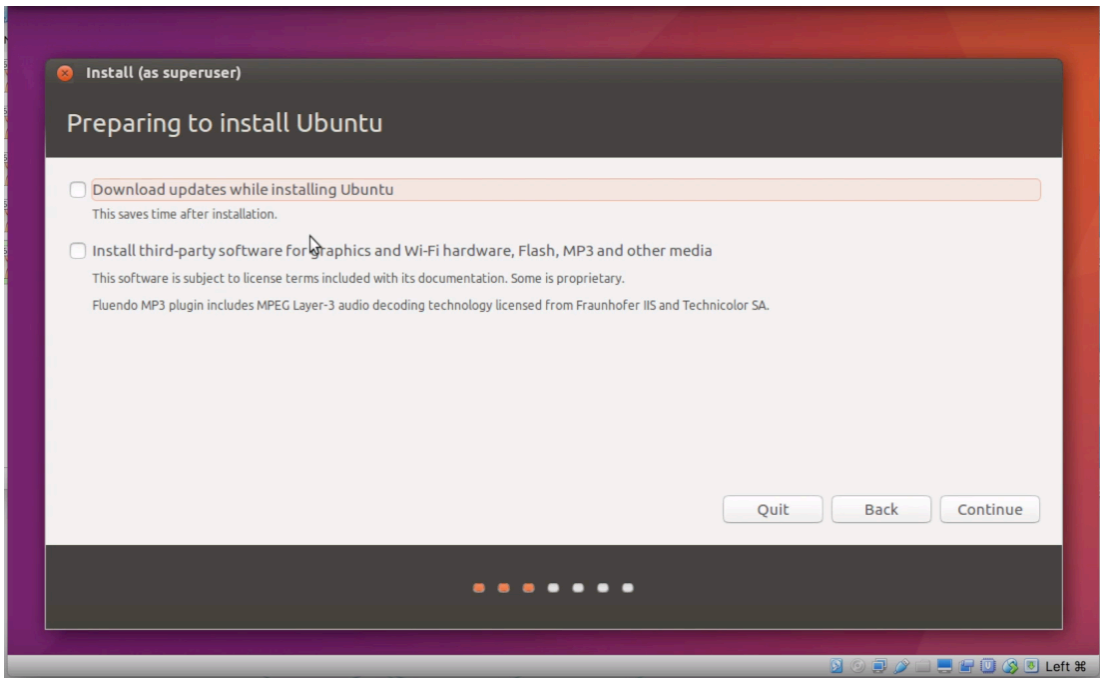
Şekil 34:İşletim sisteminin başlama aşaması

Ve şekil 35 ‘te gösterildiği gibi işletim sistemi başladıktan sonra install ubuntu seçeneği seçilmektedir.



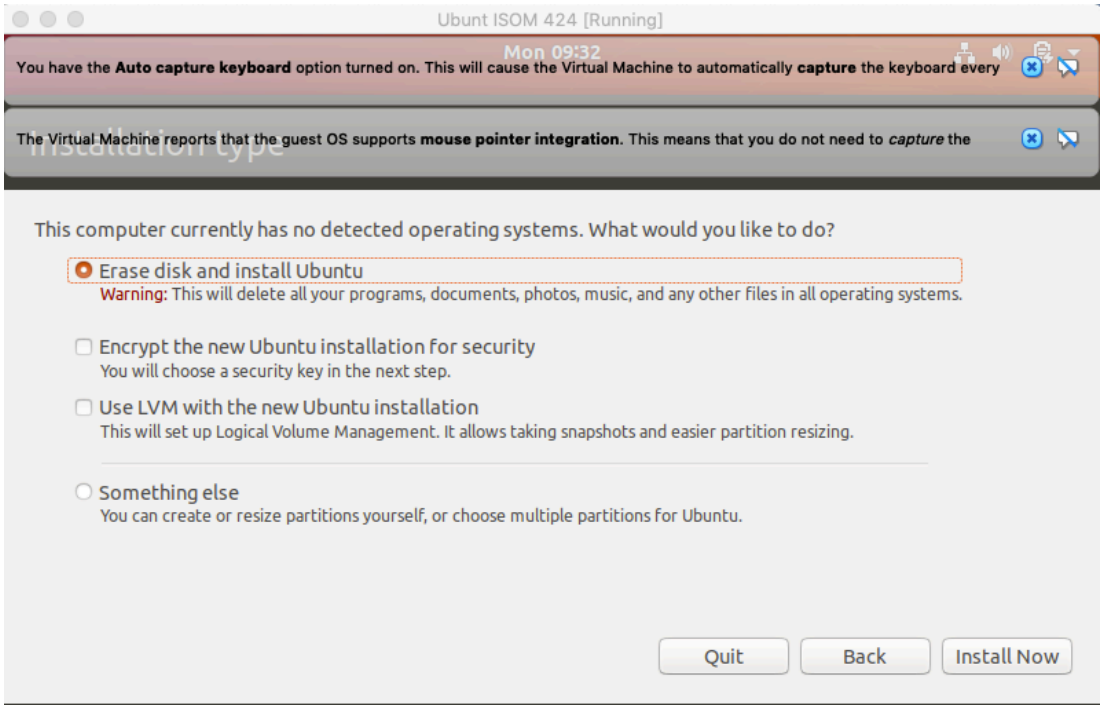
Şekil 35:İşletim sisteminin kurulumu için olan seçenekler

Şekil 36 'da gösterildiği gibi bir sonraki adımda verilen iki seçenekten hiç birini seçmeden devam seçilmiştir.



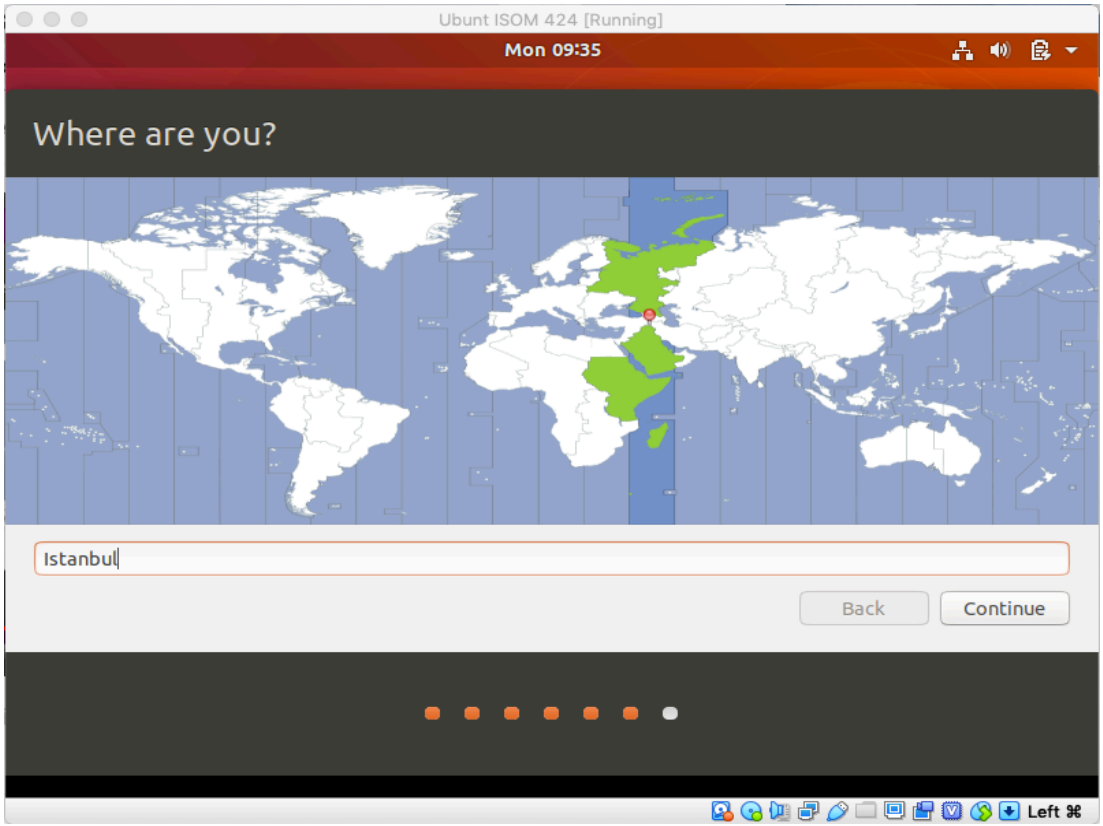
Şekil 36:Ubuntu kurulumundaki seçenekler

Sonrasında ise Şekil 37'de gösterildiği gibi kurulum türlerinden 'Erase disk and install ubuntu' seçeneği seçildikten sonra 'install now' düğmesi seçilmektedir.



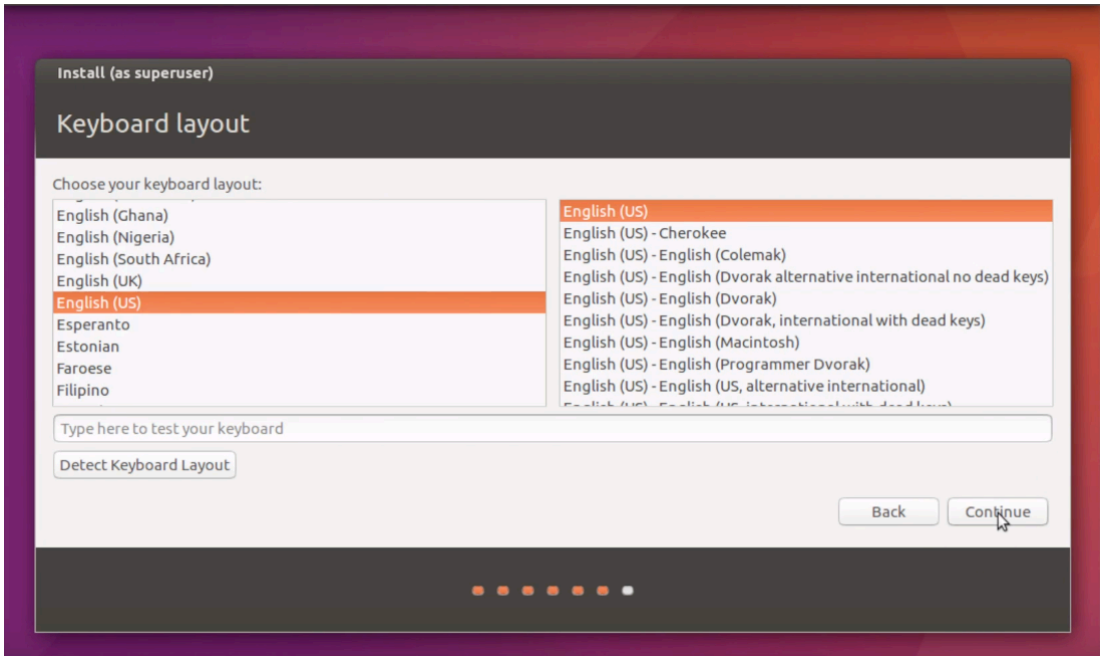
Şekil 37:Ubuntu kurulum türleri

Daha sonra Şekil 38'de gösterildiği gibi bulunduğumuz konum seçilerek 'continue' seçilmiştir.



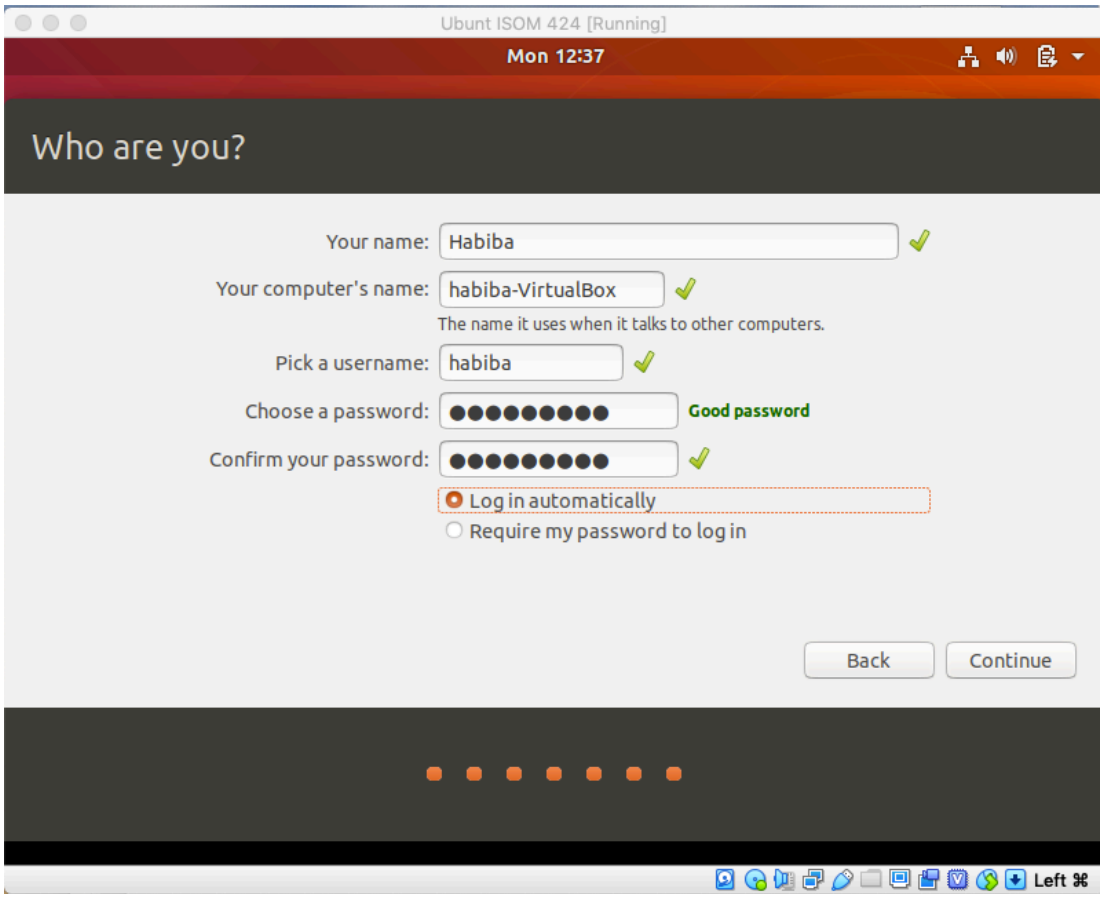
Şekil 38: Konum seçimi

Konumdan sonra şekil 39'da gösterildiği gibi klavye düzeni ayarlanmaktadır.



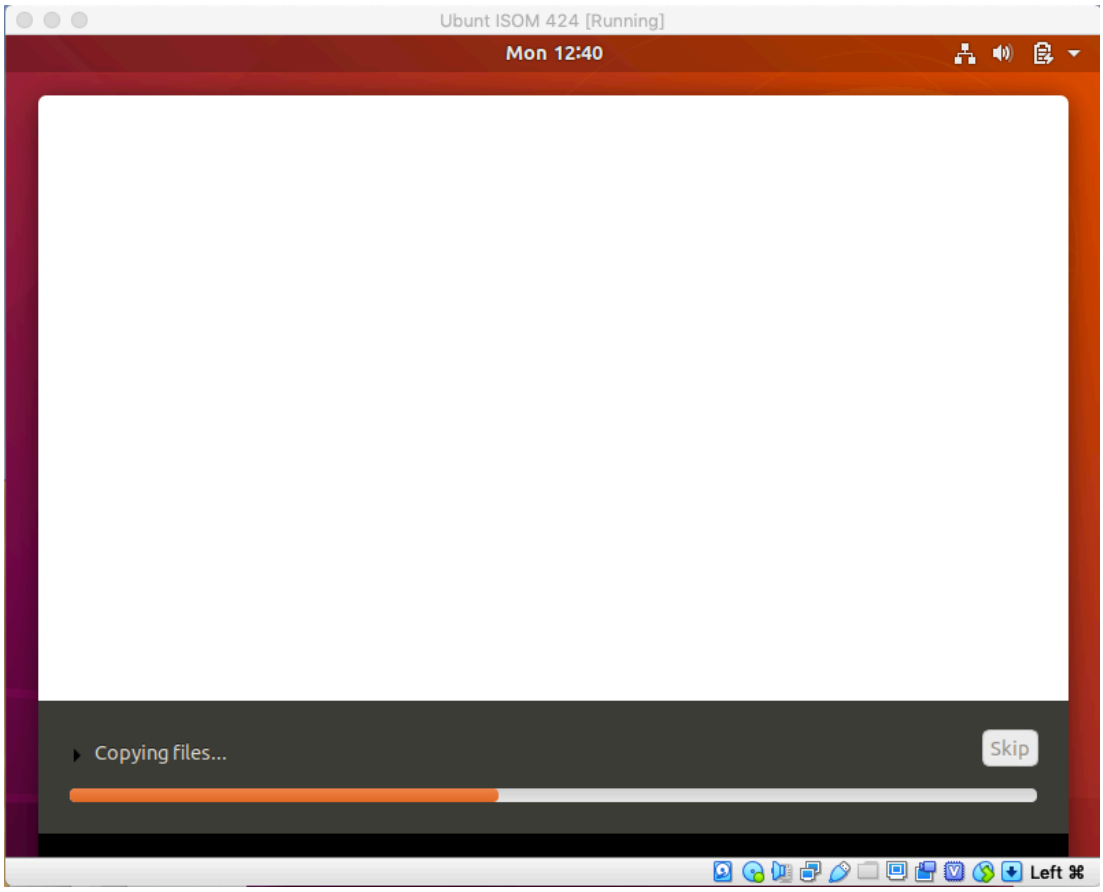
Şekil 39:Klavye düzeni

Şekil 40'da gösterildiği gibi bir sonraki aşamada isim ve bilgisayar adı yazıldıktan sonra bir kullanıcı ismi ve şifre girilerek devam edilmektedir.



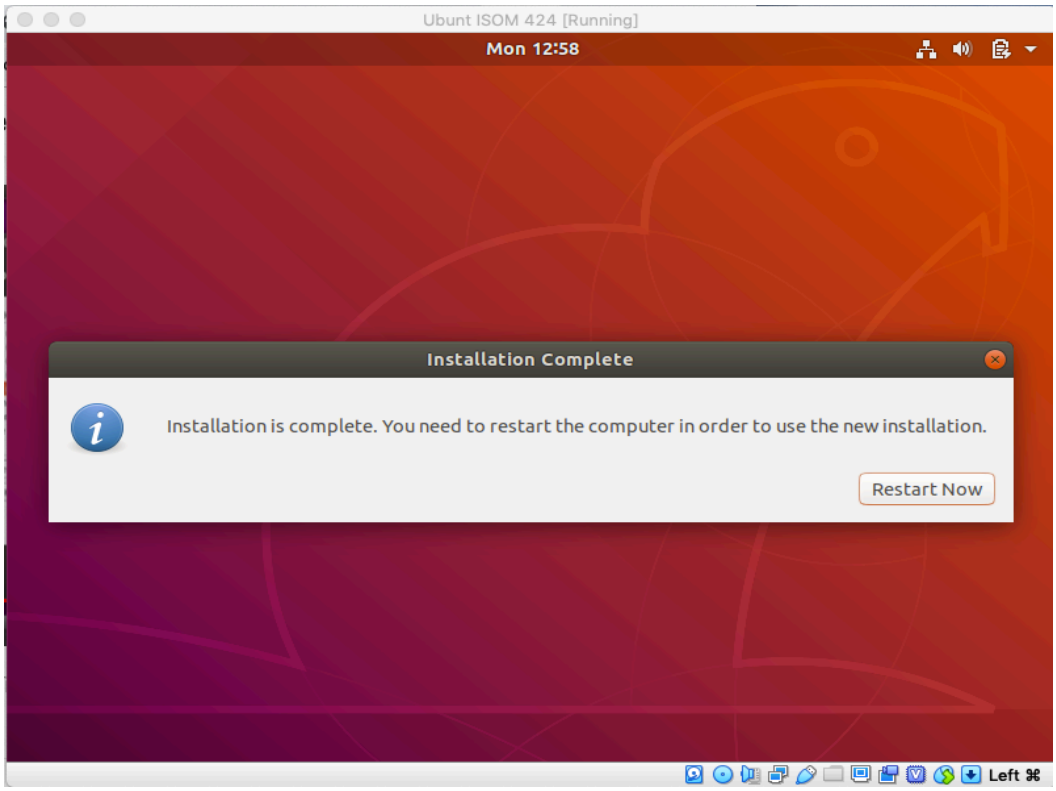
Şekil 40:Kimlik bilgileri

Şekil 41'de gösterildiği gibi kurulum işlemi biraz zaman gerektirmektedir.



Şekil 41:Bekleme süresi

Şekil 42'de gösterildiği gibi kısa bir süre sonra işletim sistemi kullanıma hazır duruma gelmiştir.

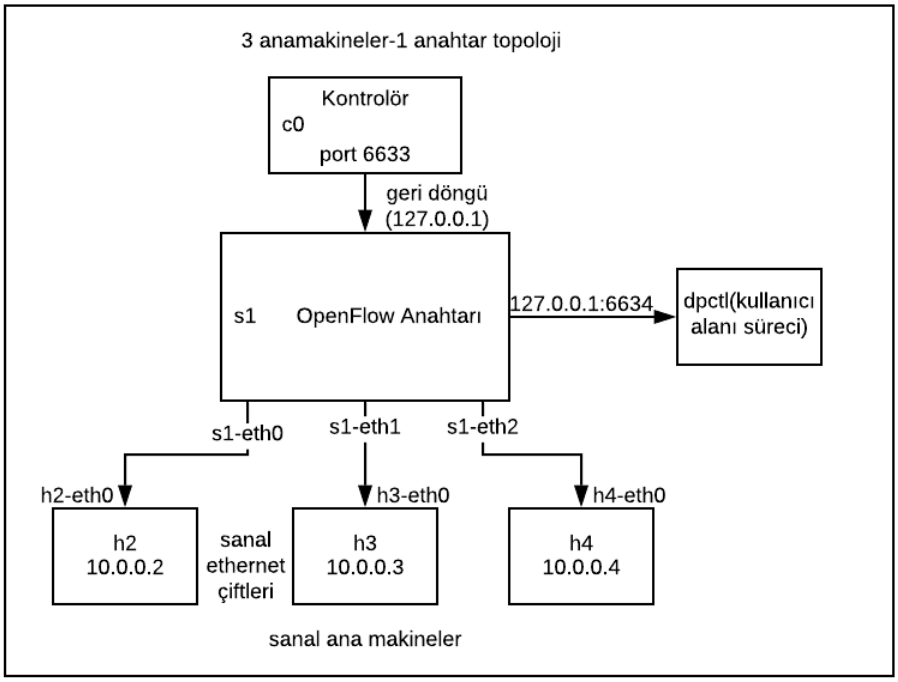


Şekil 42:İşletim sisteminin kurulum aşaması tamamlandıktan sonra Ubuntu masaüstü görünümü

C. Mininet

Tek bir Linux çekirdeğindeki uç ana bilgisayar, anahtarlar, yönlendiriciler ve bağlantılar koleksiyonunu simüle eden bir ağ emülatörüdür. Aynı çekirdeği, sistemi ve kullanıcı kodunu çalıştıran tek bir sistemin eksiksiz bir ağ gibi görünmesini sağlamak için hafif sanallaştırma kullanılmaktadır. Mininet, genellikle bir simülasyon, doğrulama, test aracı ve kaynak olarak kullanıldığı için açık kaynak SDN topluluğu için önemlidir. Mininet, GitHub'da barındırılan açık kaynaklı bir projedir. Serbestçe kullanılabilen kaynak kodunu, komut dosyalarını ve belgeleri kontrol etmekle ilgileniliyorsa GitHub'a bakılmalıdır. Bir Mininet sunucusu gerçek bir makine gibi davranır ve genellikle aynı kodu çalıştırır. Bu şekilde, bir Mininet sunucusu, isteğe bağlı programların takılabildiği ve çalıştırılabildiği bir makinenin kabuğunu temsil eder. Bu özel programlar, programın gerçek bir Ethernet gibi göründüğü paketleri kullanarak gönderebilir, alabilir ve işleyebilir, ancak aslında sanal bir anahtar / arabirimdir. Paketler, Mininet ana bilgisayarlarına nasıl yapılandırıldıklarına bağlı olarak gerçek bir Ethernet anahtarı veya yönlendirici gibi görünen sanal anahtarlar tarafından işlenir. Aslında, Mininet anahtarlarının Cisco ve diğer ticari sürümleri olan ağlara benzer, kuyruk derinliği, işleme disiplini ve poliçe işleme gibi ticari, amaca

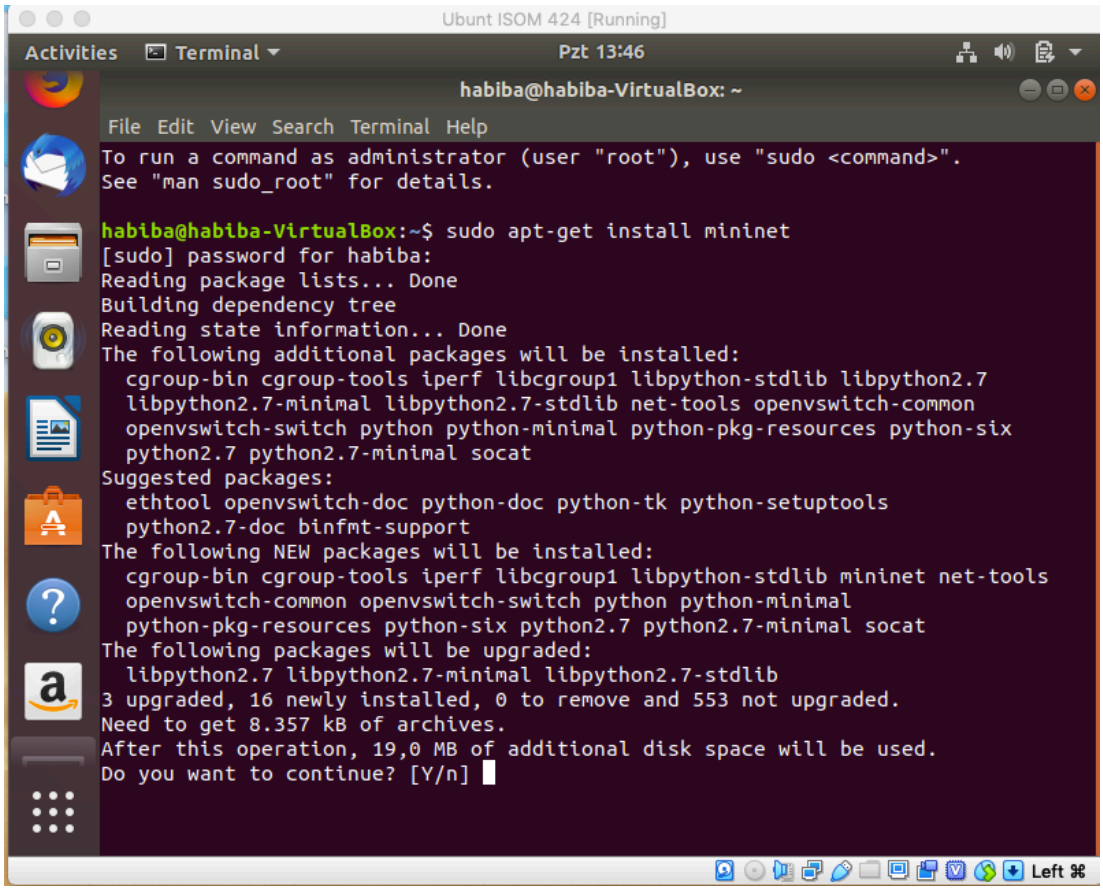
yönelik yapılmış anahtarlarının kilit özelliklerini oldukça doğru bir şekilde taklit ederek kullanılabilir. Bu yaklaşımın çok güzel bir yan etkisi, Mininet tarafından barındırılan bir ağın ölçülen performansının genellikle gerçek (taklit edilmemiş) anahtarlara, yönlendiricilere ve ana bilgisayarlara yaklaşması gerektiğidir. Şekil 43, üç ana bilgisayardan, sanal bir Open-Flow anahtarından ve bir OpenFlow kontrol cihazından oluşan basit bir Mininet ağını göstermektedir. Tüm bileşenler daha sonra erişilebilirlik için özel net-10 IP adresleri atanan sanal Ethernet bağlantıları üzerinden bağlanır. Mininet, bahsedildiği gibi, neredeyse rastgele boyut ve sıradaki çok karmaşık topolojileri destekler; böylece bir anahtar ve ekli ana bilgisayarları konfigürasyona kopyalayıp yapıştırabilir, bunları yeniden adlandırabilir ve yeni anahtarı mevcut olana ekleyebilir ve hızlıca iki anahtar ve altı ana bilgisayardan oluşan bir ağa sahip olabilmektedir. Mininet'in deney yapmak için yaygın olarak kullanılmasının bir nedeni, BGP araştırması için kullanılabilecek daha büyük İnternet benzeri topolojileri oldukça karmaşık ve gerçekçi olduğu kanıtlanmış özel topolojiler yaratılmasına izin vermesidir. Mininet'in bir başka harika özelliği de, paket iletmenin tamamen özelleştirilmesine olanak vermesidir. Belirtildiği gibi, piyasada satılan anahtarları yaklaşık olarak gösteren ana programlardan birçok örnek vardır. Bunlara ek olarak, OpenFlow protokolü kullanılarak programlanabilir ana bilgisayarlar yardımıyla bazı yeni ve yenilikçi deneyler gerçekleştirilmiştir (Castillo, 2018).



Şekil 43:Basit bir örnek Mininet ağı

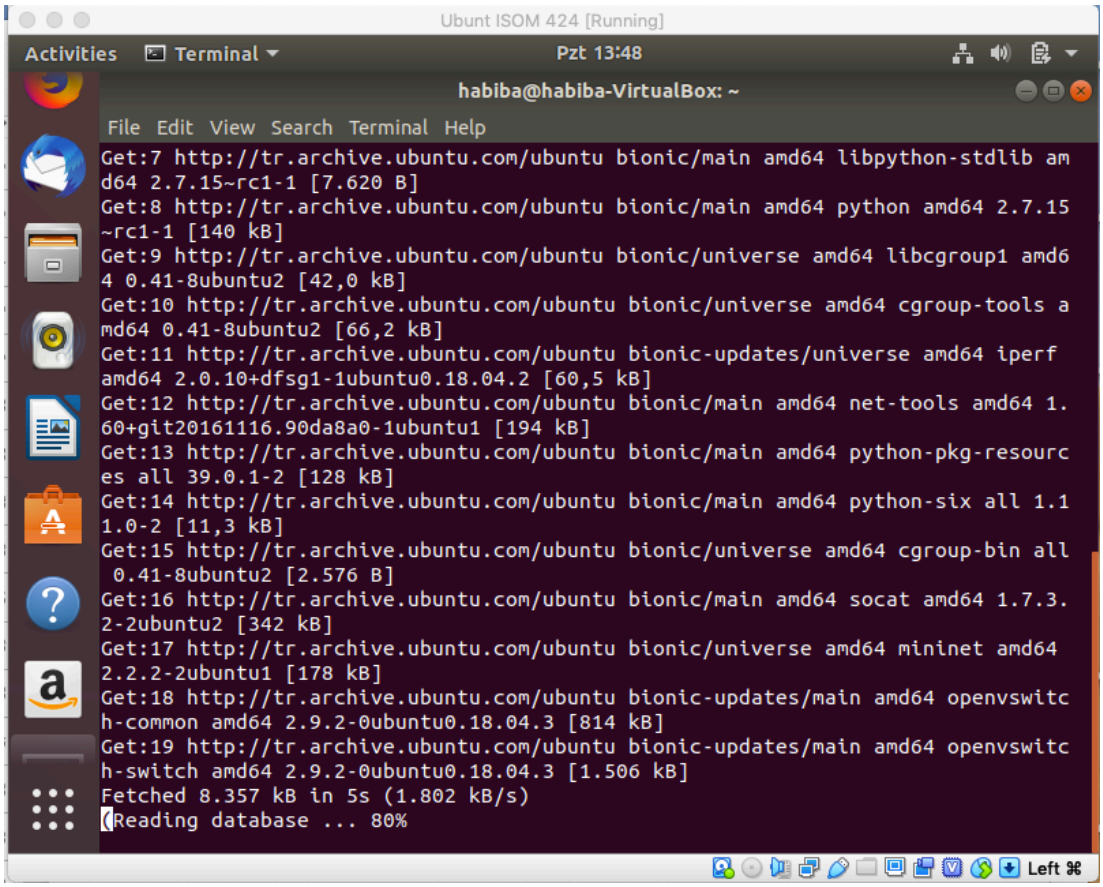
1. Mininet Kurulumu

İlk önce ubuntu'da terminal açılıp, şekil 44'te gösterildiği gibi terminal'de 'sudo apt-get install mininet' komutu yazılarak Enter tuşuna basıldıktan sonra Sistem şifresi girilerek Mininet kurulumuna başlanılmaktadır.



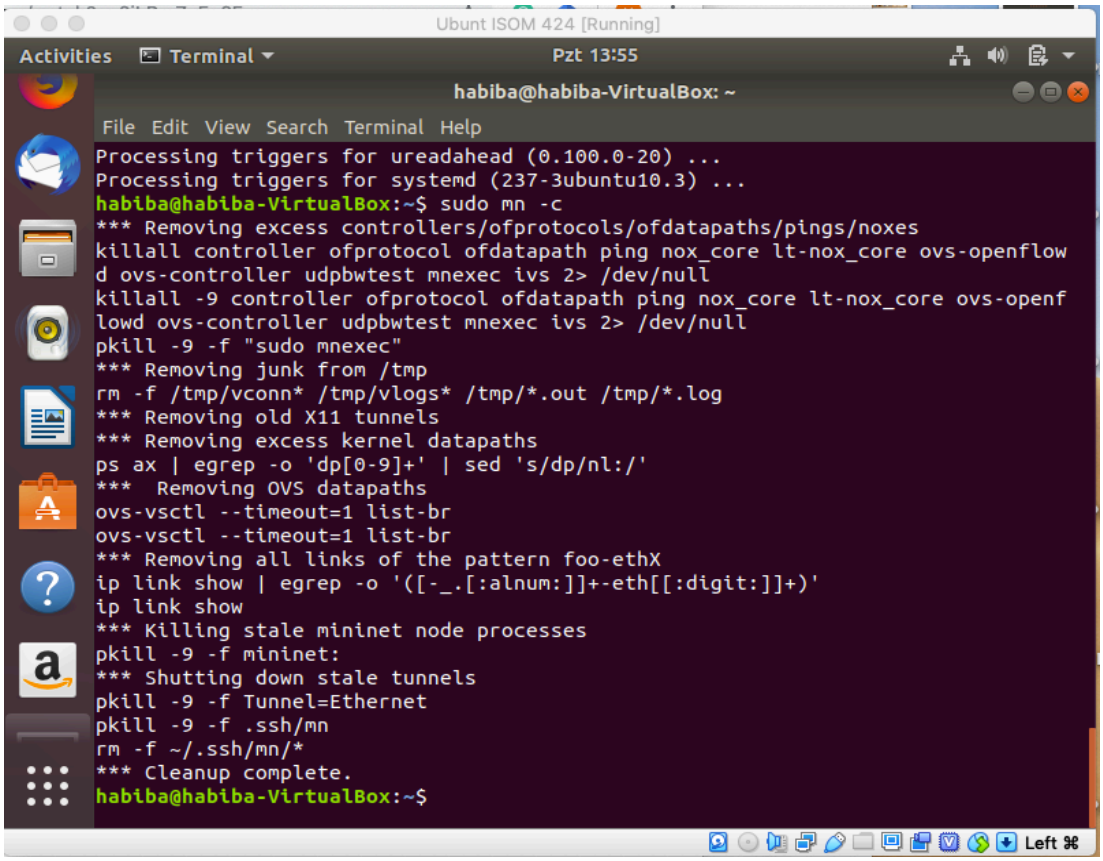
Şekil 44:Mininet kurulumu

Sonrasında şekil 45'te gösterildiği gibi devam etmek ister misiniz sorusuna evet cevabı verilmektedir.



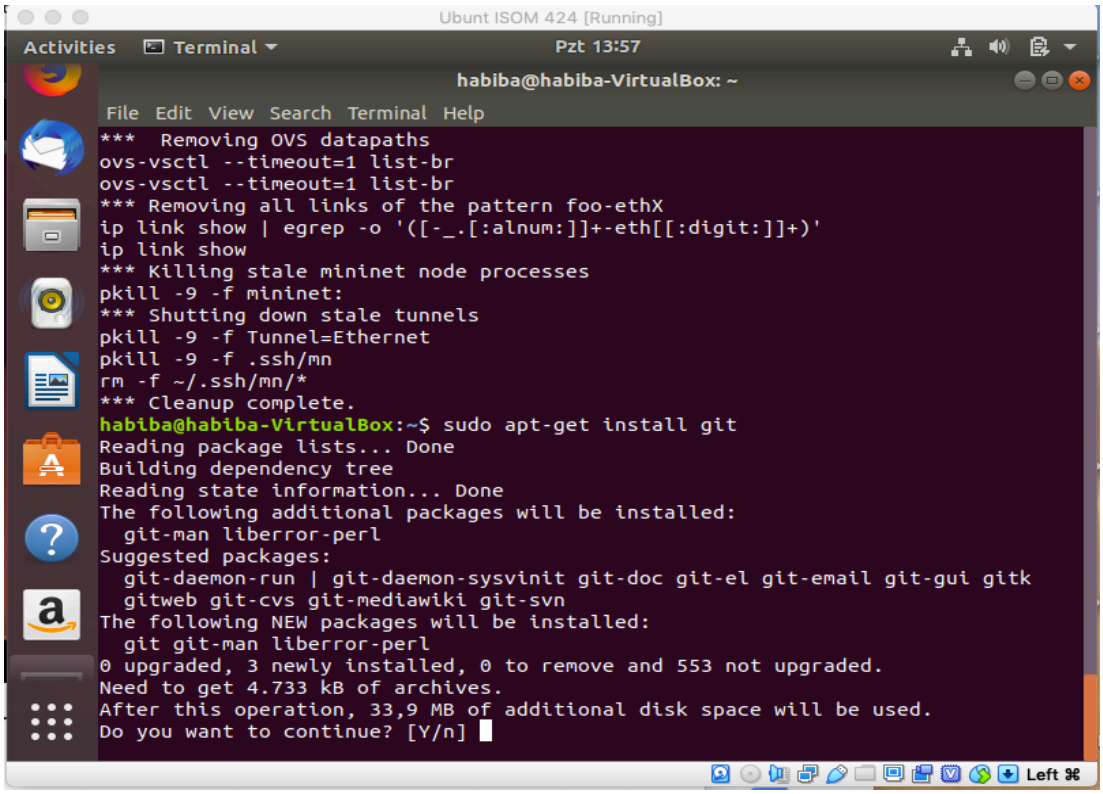
Şekil 45:Mininet kurulumuna devam etmek istedikten sonra

Şekil 46'da gösterildiği gibi Kurulum tamamlandıktan sonra 'sudo mn -c' komutu gereksiz dosyaları silmek için yazılmaktadır.



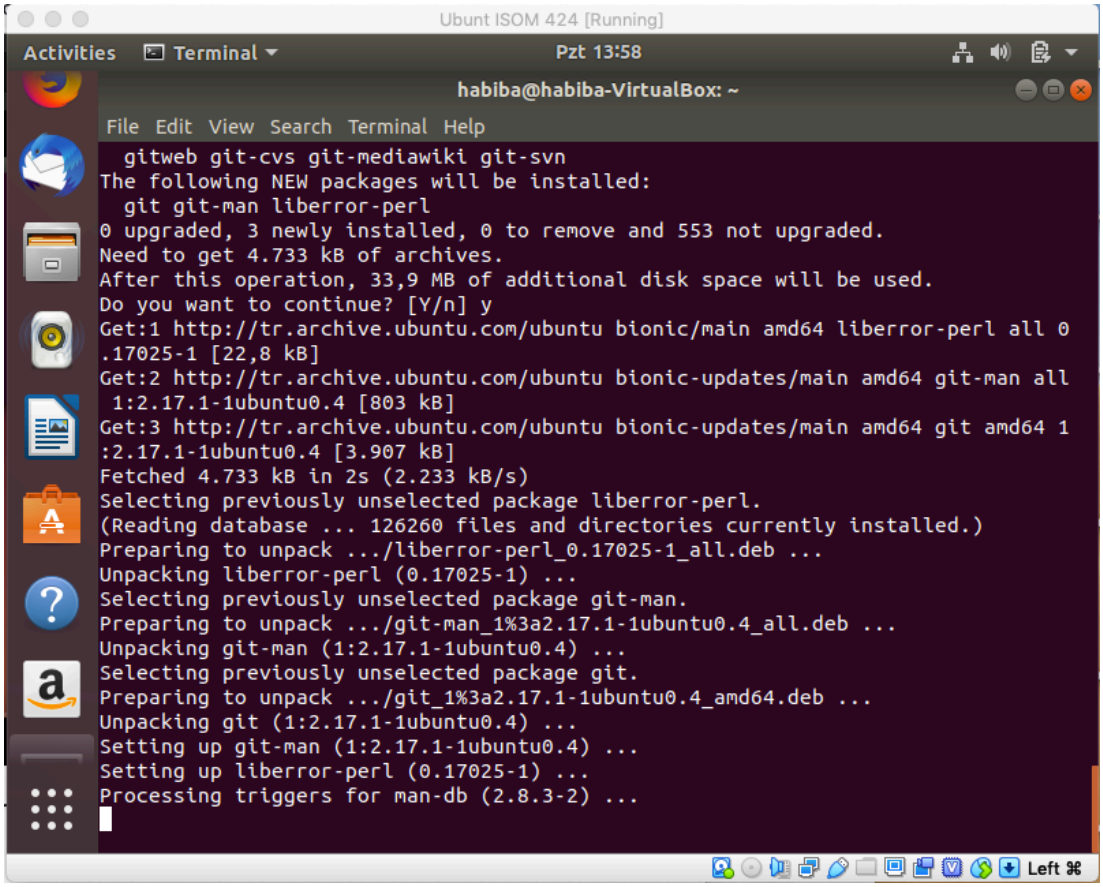
Şekil 46:Gereksiz dosyaları silmek

Daha sonra şekil 47'de gösterildiği gibi gerekli dosyaları indirmek için bir depoya ihtiyaç olduğundan Git deposu kurulmaktadır.



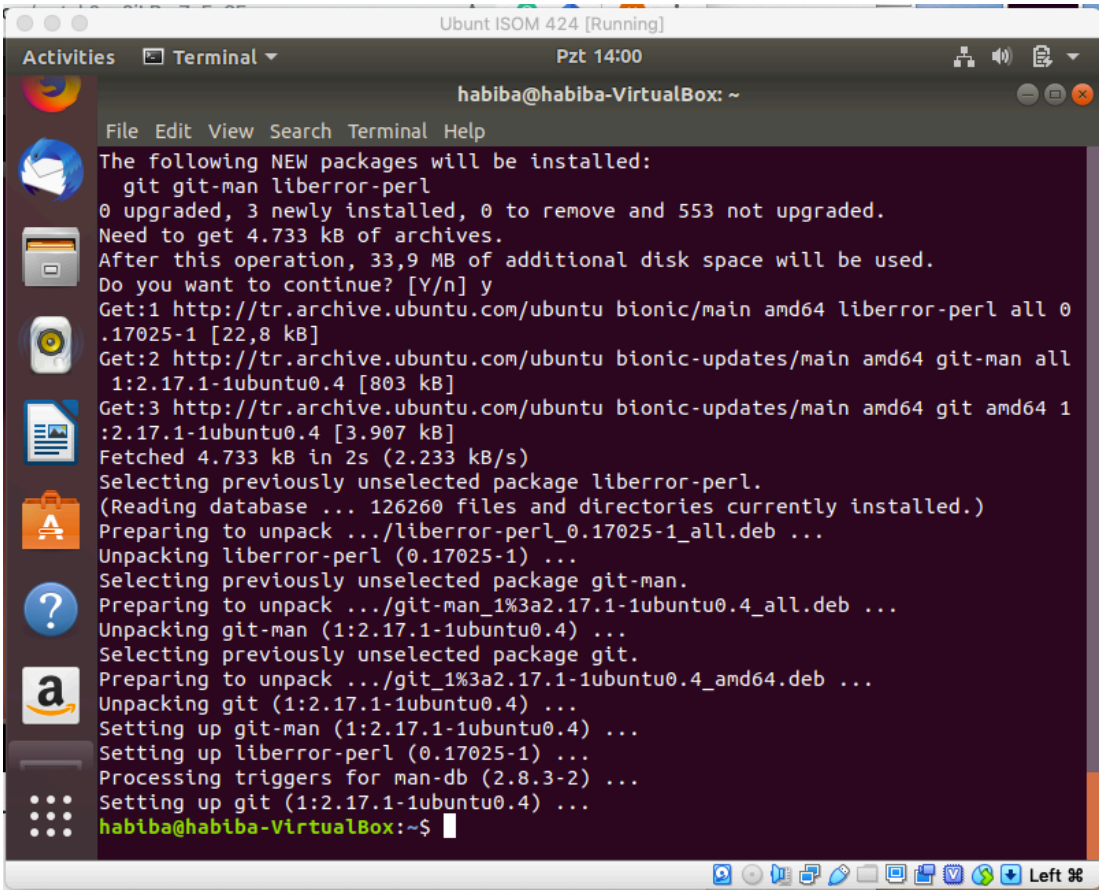
Şekil 47:Git kurulumu

Şekil 48'de gösterildiği gibi tekrar devam etmek ister misiniz sorusuna işlemi tamamlamak için evet cevabı verilmektedir.



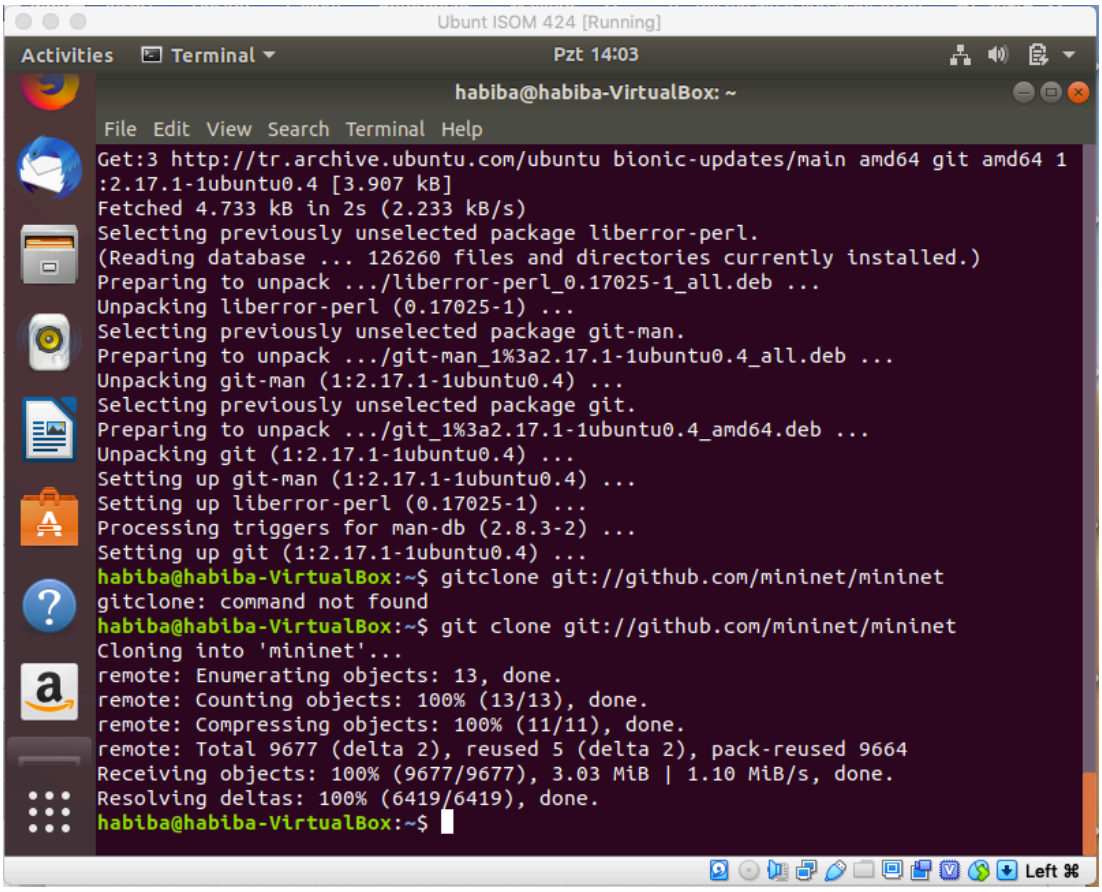
Şekil 48:Git kurulumu'nun devamı

Git kurulumu tamamlandıktan sonra şekil 49'da gösterilen bilgiler verilmektedir.



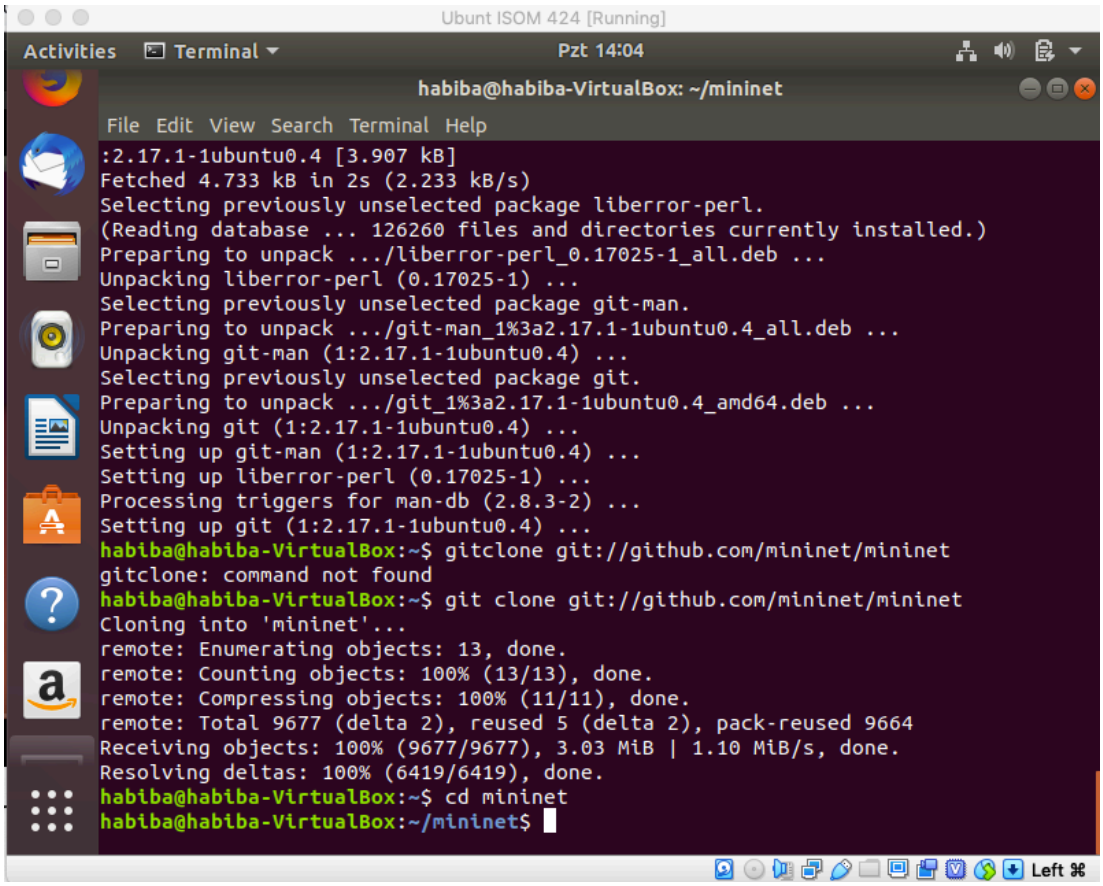
Şekil 49:Git kurulumu tamamlandıktan sonra verilen bilgiler

Daha sonra şekil 50'da gösterildiği gibi 'git clone git://github.com/mininet/mininet' komutu kullanılarak git deposundan bulunan mininet hakkındaki bilgiler mininet'e kopyalanmaktadır.



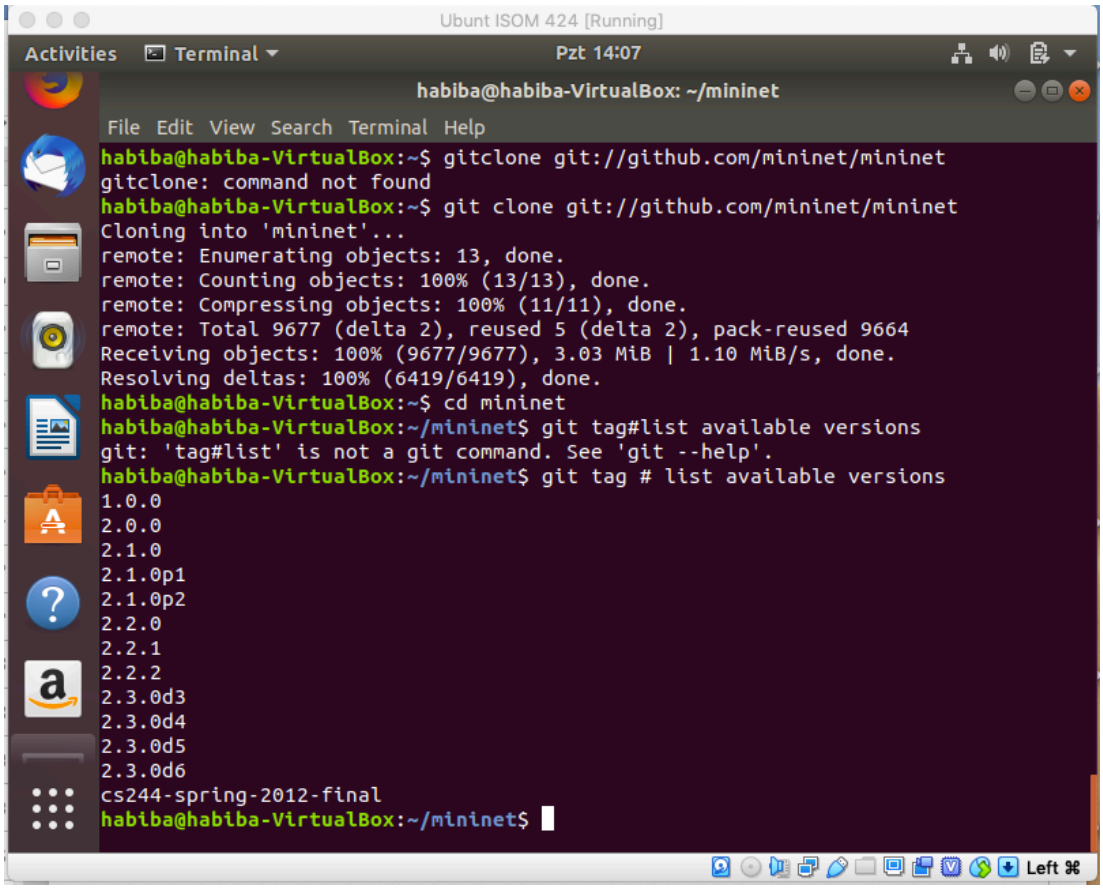
Şekil 50:Mininet bilgilerinin git deposundan alınması

Şekil 51'de gösterildiği gibi Linux'ta cd komutu, dizin değiştir komutu olarak bilinir. Mevcut çalışma dizinini değiştirmek için kullanılır. Linux'ta bir alt dizinde hareket etmek içinde kullanılmaktadır.



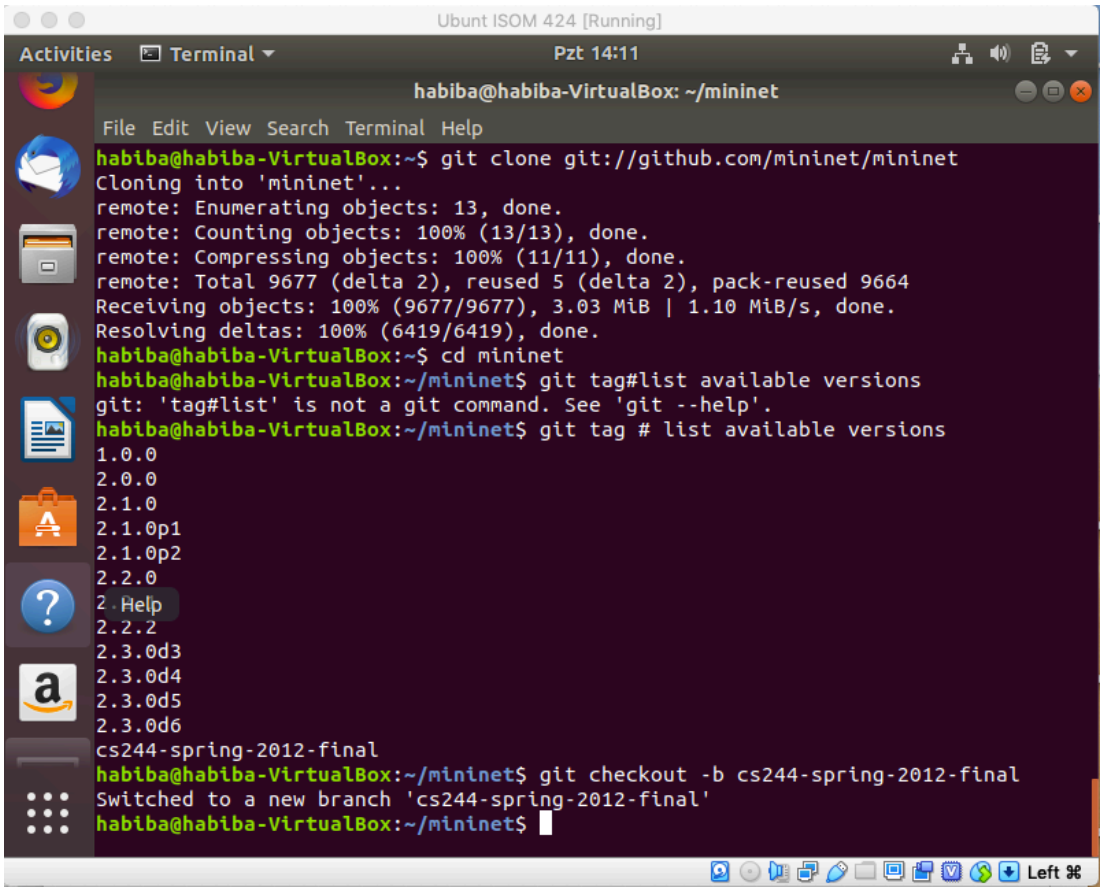
Şekil 51:Mininet dizinine girmek için cd komutu kullanımı

Sonrasında mininet'in mevcut sürümlerini görmek için 'git tag # list available version' komutu kullanılmıştır. Şekil 52'de mevcut sürümler gösterilmektedir.



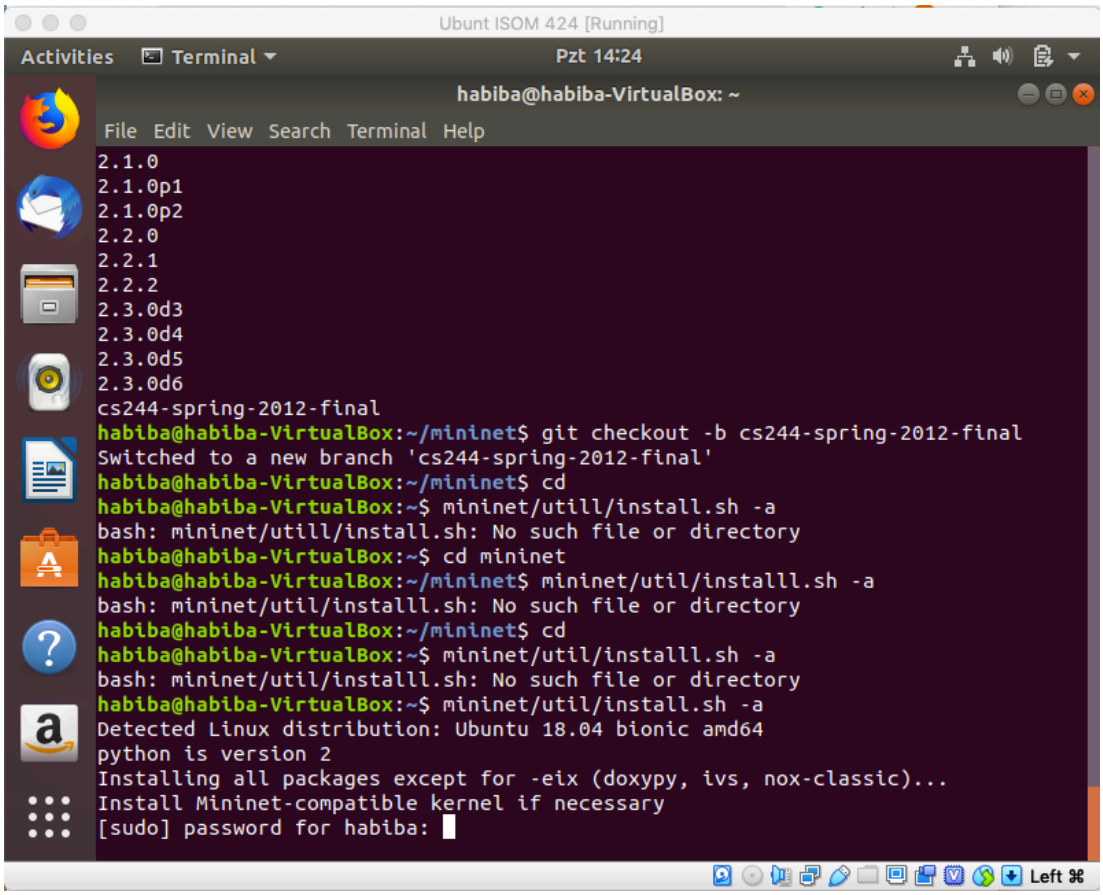
Şekil 52:Mininet'in mevcut sürümleri

Şekil 53'te gösterildiği gibi Mevcut versiyonların en son sürümünü 'git checkout -b cs244-spring-2012-final' komutu ile kurulmaktadır.



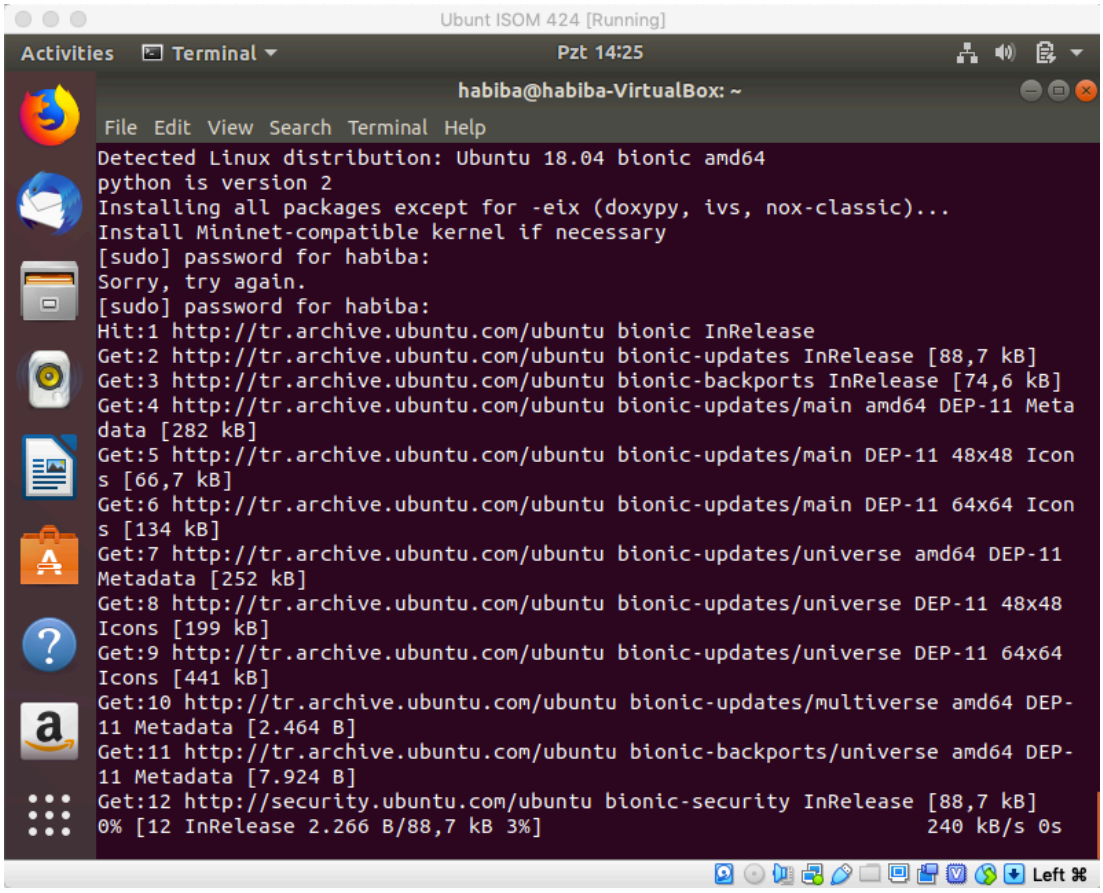
Şekil 53: Mevcut versiyonların en son sürümü

Şekil 54'te gösterildiği gibi bu aşamada 'mininet/util/install.sh -a' komutu kullanılmaktadır. Bu komut Mininet dizininde bulunan tüm özellikleri yüklemektedir. -a: Open vSwitch gibi bağımlılıklar ve OpenFlow wireshark dissector ve POX gibi ilaveler de dahil olmak üzere Mininet VM'de bulunan tüm özellikleri yüklemektedir.



Şekil 54:Mininet'te bulunan tüm özelliklerin yüklenmesi

Daha sonra şekil 55'te gösterildiği gibi 'mininet/utill/install.sh -a ubuntu 18.04 trusty i386' komutu kullanılarak Ubuntu 18.04 versiyonunda bulunan tüm özellikler yüklenmektedir (Mininet setup from start to finish on Ubuntu 14.04 virtual machine, 2015).



Şekil 55:Ubuntu 18.04'te bulunan tüm özelliklerin yüklenmesi

D. Wireshark Yazılımı

Wireshark mevcut olan en iyi açık kaynaklı ağ analizörüdür. Ticari ağ analizörleriyle karşılaştırılabilir özelliklerle doludur ve geniş, çeşitli yazar koleksiyonuyla sürekli olarak yeni geliştirmeler yapılmaktadır. Wireshark, tüm ağ araç takımları için kararlı ve kullanışlı bir bileşendir ve yeni özellikler ve hata düzeltmeleri her zaman geliştirilmektedir. Wireshark'ın ilk günlerinden beri (hafif ve ince olarak adlandırıldığı zamanlardan) çok ilerleme kaydedilmiştir; Uygulama şuanda karşılaştırılabilirliği olan (ve bazı açılardan) ticari dinleme yazılımlarından daha iyi performans göstermektedir (Orebaugh, et al., 2007).

1. Wireshark Nedir?

wireshark bir ağ analizörüdür. Ağdan paketleri okur, kod çözer ve anlaşılması kolay bir biçimde sunar. Wireshark'ın en iyi yönlerinden bazıları açık kaynaklı olması, aktif olarak sürdürülmesi ve serbestçe alçaltmanın yapılabilmesidir. Wireshark'ın diğer önemli yönlerinden bazıları:

- Gnu'nun UNIX (GNU) Genel kamu lisansı (GPL) açık kaynaklı lisansı altında dağıtılmaktadır.
- Karışık ve karışık olmayan modlarda çalışmaktadır.
- Ağdan veri alabilir veya bir yakalama dosyasından okuyabilir.
- Okunması kolay ve yapılandırılabilir bir GUI'ye sahiptir.
- Zengin ekran filtresi özelliklerine sahiptir.
- tcpdump format yakalama filtrelerini destekler. Bir iletim kontrol protokolü (TCP) oturumu yeniden yapılandıran ve bilgi değişimi için, genişletilmiş ikili kodlanmış ondalık kod değişimi kodunu (EBCDIC), Onaltılık (onaltılık) dökümü veya C dizileri için Amerikan standart kodunda (ASCII) görüntüleyen bir özelliğe sahiptir.
- Önceden derlenmiş ikili dosyalarda ve kaynak kodda bulunur.
- Benzersiz bilgi ve bilgi işlem sistemi (UNIX) tabanlı işletim sistemleri (OS) ve pencereler dahil olmak üzere 20'den fazla platformda çalışır ve Mac OS X için kullanılabilen üçüncü taraf paketler vardır.
- 750'den fazla protokolü desteklemekte ve açık kaynak kodlu olduğu için yenileri sık sık kullanılmaktadır.
- 25'ten fazla üründen yakalama dosyalarını okuyabilir.
- Yakalama dosyalarını çeşitli biçimlerde (örneğin, libpcap, ağ ortakları sniffer, Microsoft Ağ İzleyicisi (NetMon) ve sun snop) kaydedebilir.
- tshark adlı ağ analiz cihazının komut satırı sürümünü kapsamaktadır.
- editcap, mergecap ve text2pcap gibi çeşitli destekleyici programlar kapsamaktadır.
- Çıktı, düz metin veya postscript olarak kaydedilebilir veya yazdırılabilir (Orebaugh, et al., 2007).

a. Wireshark kurulumu

Daha sonra ağı analiz etmek için wireshark yazılımı kullanılmaktadır. Aynı zamanda Ubuntu sisteminde önceden yüklenmiş olduğu için wireshark yazılımı ‘sudo & wireshark’ komutu kullanılarak açılmaktadır. Eğer yüklenmemişse aynı komut yüklemek için kullanılabilir.

E. Pox Kontrolörünü Ayarlamak

Pox kontrolörü iki yöntemle ayarlanmaktadır.

- Bu yöntem de ilk olarak cd pox/pox yazarak pox klasörün'a girilmektedir daha sonra mevcut dosyalar listelenmektedir. Sonrasında cd forwarding kullanılarak forwarding klasörünün'e girilmektedir. Sonraki aşamada forwarding klasöründe mevcut olan dosyalar listelenmektedir. En son aşamada 'sudo ~/pox/pox.py forwarding.l2_learning' komutu pox kontrolörünü öğrenmesi için ayarlamaktadır.
- İlk önce nano editor'a su komutu ve sistem şifresi ile giriş yapılmaktadır. Daha sonra 'nano/usr/bin' komutu kullanılarak nano editoruna giriş yapılmıştır. Bu aşama dan Sonra

```
nano editorda  #!/bin/sh
echo "Pox Controller  Habiba Amed"
sleep 2
```

komutu yazılmıştır. Sonra pox kontrolörünü paketleri ayıklaması için ayarlanmaktadır. 'Cd /home/habiba/pox && ./pox.py forwarding.l2_pairs info.packet_dump samples.pretty_log log.level --DEBUG' (Mininet Part 2, 2017)

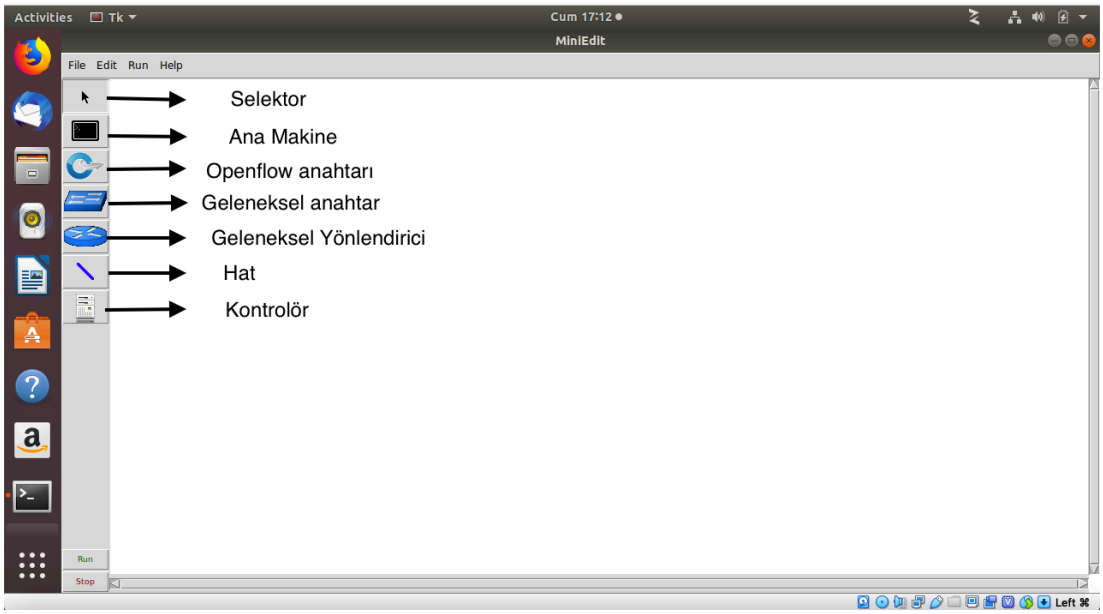
Ayarlama tamamlandıktan sonra yapılan değişiklikleri kaydettikten hemen sonra komut satırına geri dönüş yapılmaktadır. Ardından komut satırında 'chmod a+x' yazılmıştır. Unix benzeri işletim sistemlerinde, chmod komutu bir dosyanın erişim modunu değiştirmek için kullanılır. İsmi, değişim modunun kısaltmasıdır. 'a' all kelimesinin kısaltımı olup tüm gruptaki kullanıcıların dosyaya erişim hakkı vermektedir. 'X' ise dosyayı çalıştırma veya dizin olması durumunda dosyayı arama izni vermektedir (khan, n.d.). Tüm ayarlama aşaması tamamlandıktan sonra test etmek için komut satırında pox yazıldıktan hemen sonra pox çalışmaya başlamıştır.

F. Miniedite Bir Yazılım Tanımlı Ağ Tasarlamak

Bu tez kapsamında, tek kontrolörlu yazılım tanımlı ağ, üç kontrolörlü yazılım tanımlı ağ, aynı zamanda bir sonraki başlıkta açıklanan komut satırında bir yazılım tanımlı ağ tasarlamak olmak üzere üç tür yazılım tanımlı ağ tasarlanmıştır.

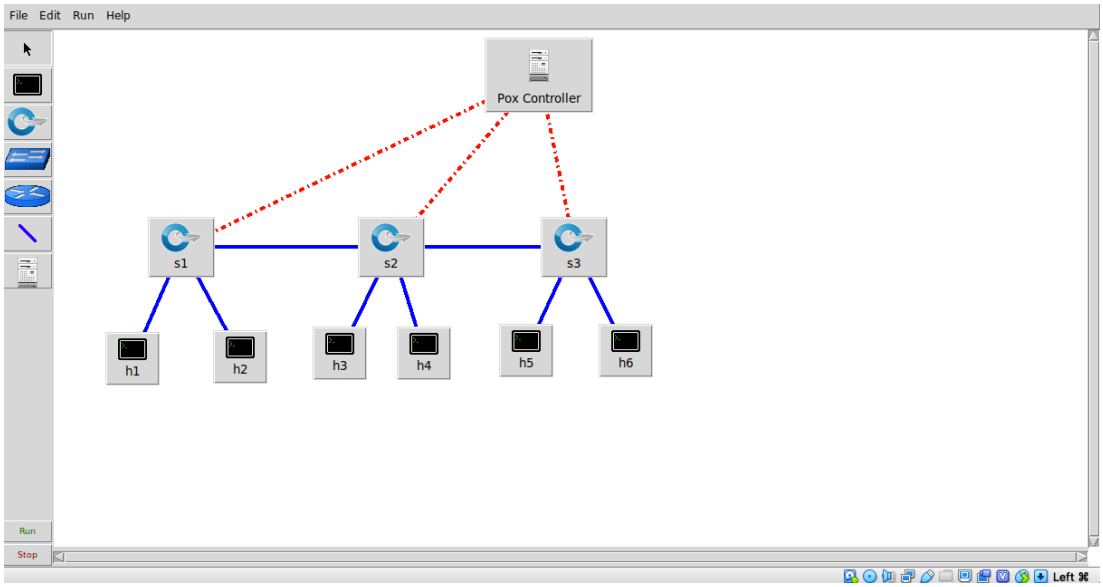
Miniedite bir yazılım tanımlı ağ tasarlamak için ilk önce ubuntu işletim sisteminde uygulama bölümünden terminal açılmıştır. Sonra miniedit.py dosyası mininet klasöründen home klasörüne götürülmüştür. Ardından komut satırında './miniedit.py' komutu çalıştırılmaktadır.

Daha sonra ise şekil 56‘da gösterildiği gibi selektörü seçerek ardından anahtar sekmesine tıklanmıştır.



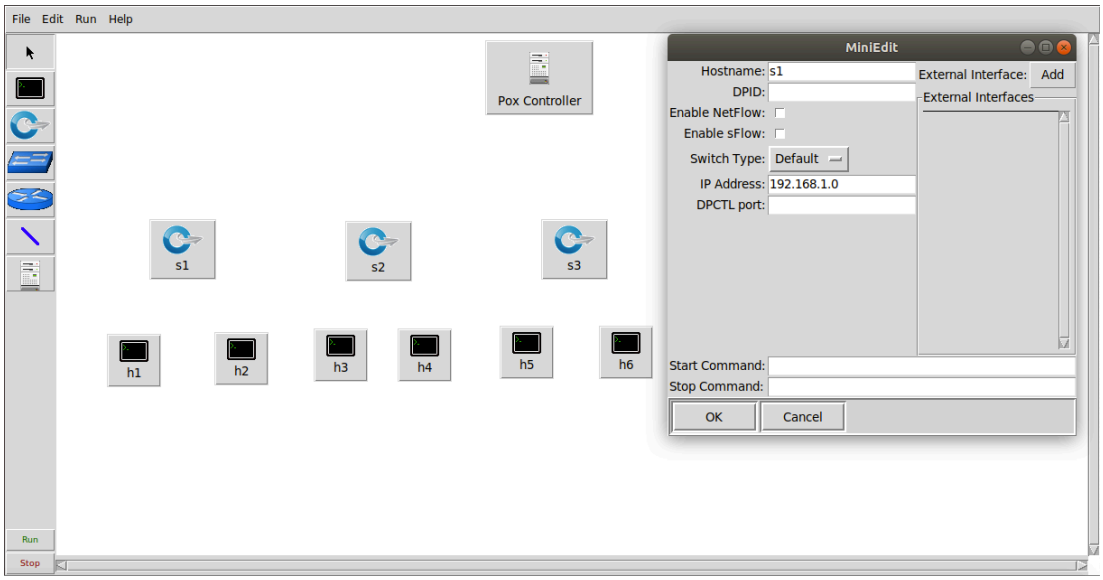
Şekil 56:Miniedit'te görünüm ve sekmeler

Şekil 57’de görüldüğü gibi üç anahtar yerleştirildikten sonra altı adet ana makine dizayna eklenmiştir ve ardından bir adet kontrolör yerleştirilmiştir.



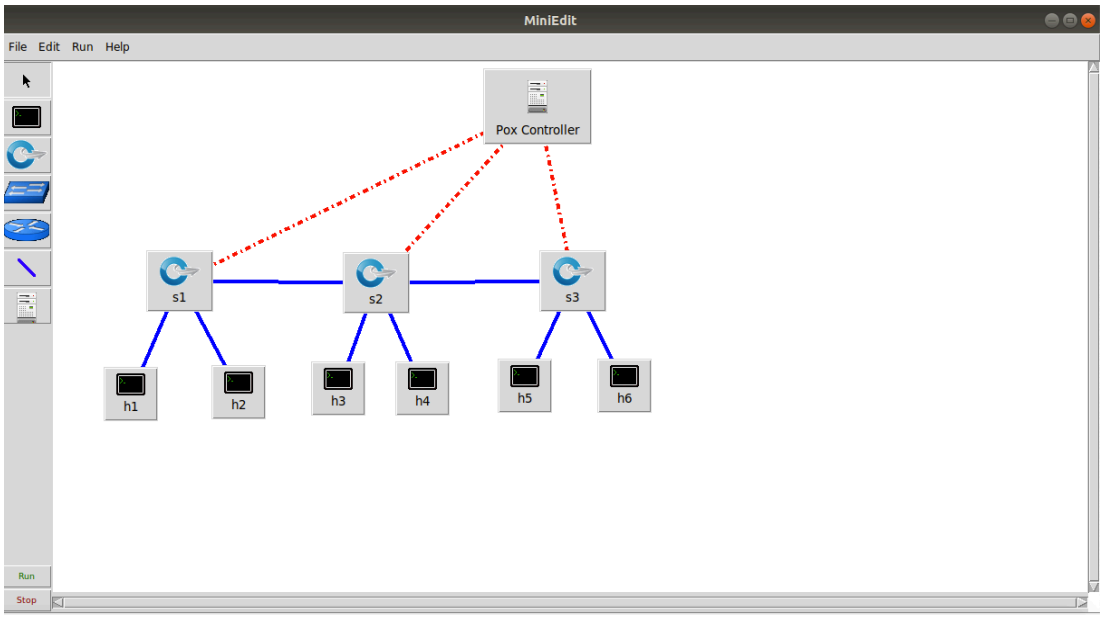
Şekil 57:Tek kontrolörlü yazılım tanımlı ağ

Bir sonraki adımda kontrolöre POX cont ismi verilerek kontrolör tipi OVS kontrolör olarak seçilmiştir. Daha sonra bir numaralı anahtarın üzerine sağ tıklayarak şekil 58’de gösterildiği gibi bir yönlendirici İP adresi 192.168.1.0 verilmektedir. Bu işlem sırasıyla her üç anahtar içinde tekrarlanmaktadır.



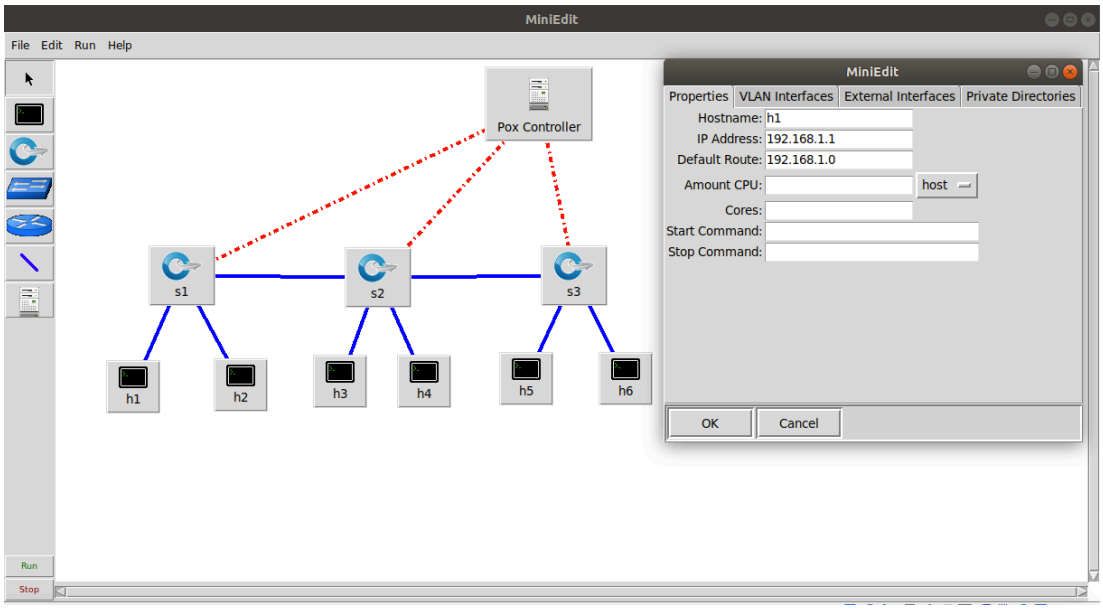
Şekil 58:Anahtarlara İP adres verilmesi

Sonrasında ise şekil 59'da gösterildiği gibi line sekmesi üzerine tıklayarak tüm ağ cihazları birbirine bağlanmıştır.



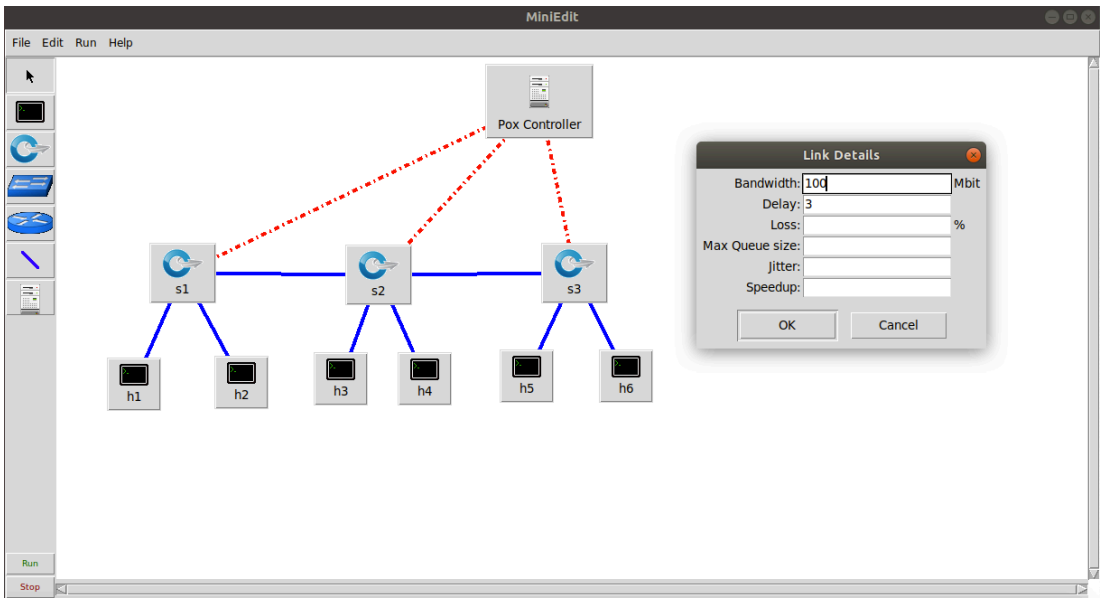
Şekil 59:Yazılım Tanımlı Ağın son hali

Sonra şekil 60'da gösterildiği gibi ana makinelere IP adres verilmektedir.



Şekil 60:Ana makinelere ip adres verilmesi

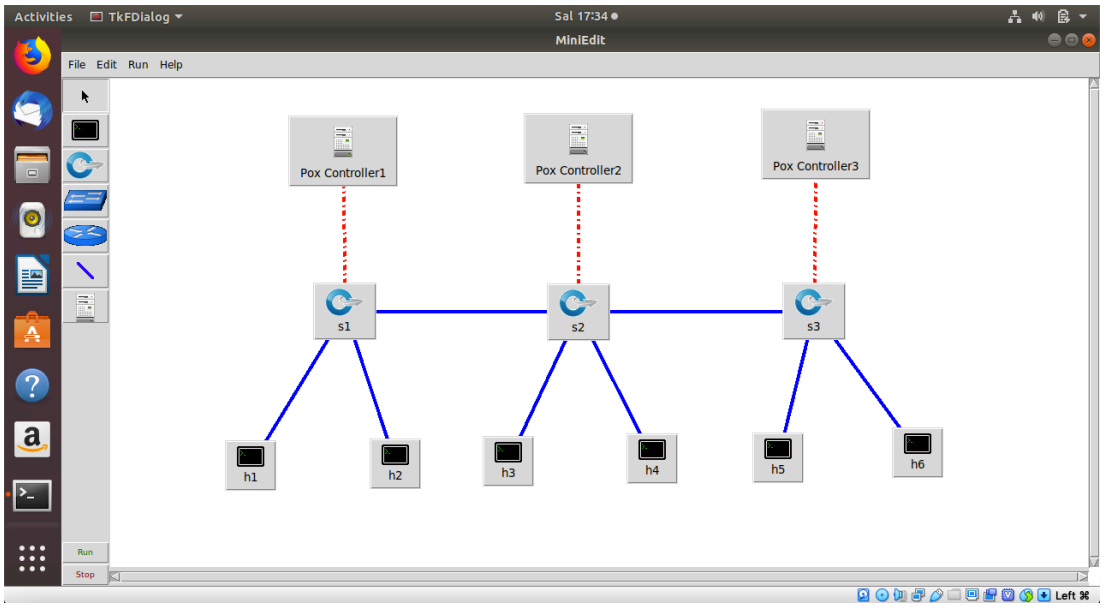
Bu işlem her altı ana makinede yapıldıktan sonra şekil 61'de gösterildiği gibi ilk anahtar ve ilk ana makinenin bağlantısı üzerine sağ tıklayarak bant genişliğini 100 MBit ve gecikmeye 3 yazılmıştır.



Şekil 61:Bağlantılara değer atanması

Bu işlem her altı linke de farklı gecikme değerleri verilerek yapılmıştır. Tasarım tamamlandıktan sonra python kodları demoson3.py ve mininet dosyası demoson.mn olarak kaydedilmiştir. Şekil 62'de gösterildiği gibi çok kontrolörlü yazılım tanımlı ağ da aynı şekilde tasarlanmış olup tek fark kontrolörlere farklı ad verilmesi olmuştur. Her iki tasarımı çalıştırmak için . py dosyasını komut satırında yazmak lazımdır. Burada .py dosyasına nano demeson3.py biçiminde , nano editörüne gidilmekte ve nano editörde paket oranı saniyede kilo bit cinsinden eklenmektedir, yapılan

değişiklikleri kaydettikten sonra ‘sudo python ./demoson3.py’ yazarak ilk tasarım çalıştırılmıştır.



Şekil 62:Çok kontrolü Yazılım Tanım Ağ

Burada kontrolör isminde bir sorun ortaya çıkmıştır ve sorun kelimeler arasındaki aralık ile ilgilidir. Aynı zamanda kontrolör türü de remote olarak python kodu’na eklenip düzeltme bittikten sonra tekrar demoson3.py çalıştırılmıştır. İkinci tasarım çoklu kontrolörlü olup denemeson.mn mininet dizaynı ve denson.py python kodlarıdır. Her iki dizaynı çalıştırmadan önce Pox kontrolörünü çalıştırmak gerekmektedir (Mininet Part 2, 2017).

G. Komut Satırında Bir Yazılım Tanımlı Ağ Tasarlamak

Komut satırında bir yazılım tanımlı ağ tasarlamak için ilk önce kontrolörü başlatarak sonrasında komut satırında aşağıdaki komutu yazmak gerekmektedir.

Sudo mn --topo tree, depth=5,fanout=5 –controller=remote,ip=127.0.0.1,port=6634 –mac (Team, 2018)

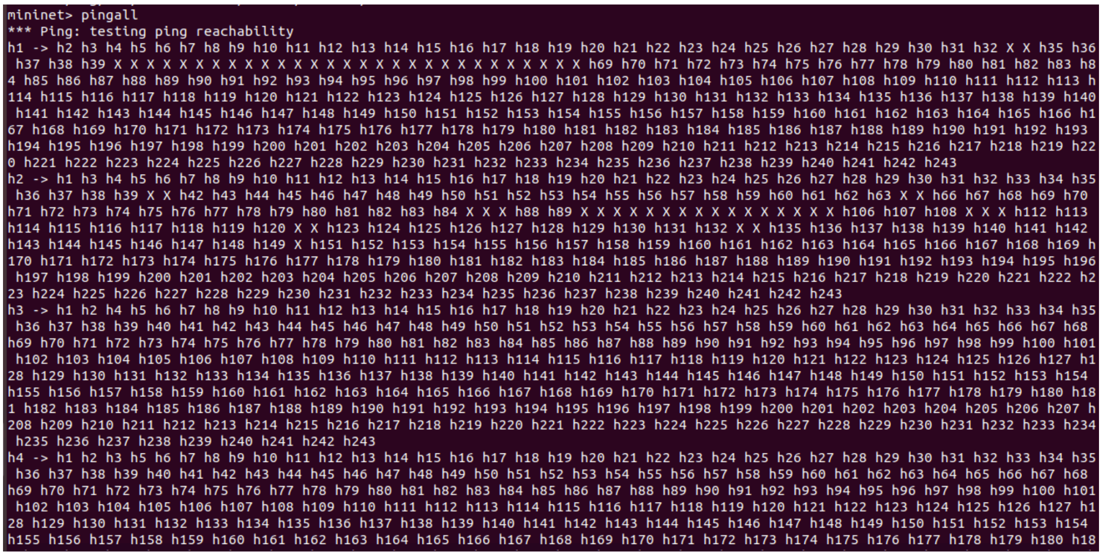
Yada şu komutla yeni bir ağ oluşturulabilir.

$ sudo mm --switch = ovsk, protocols = OpenFlow13 --controller = remote --topo = tree, depth = 5, fanout = 5 --ip base = 172.17.0.2

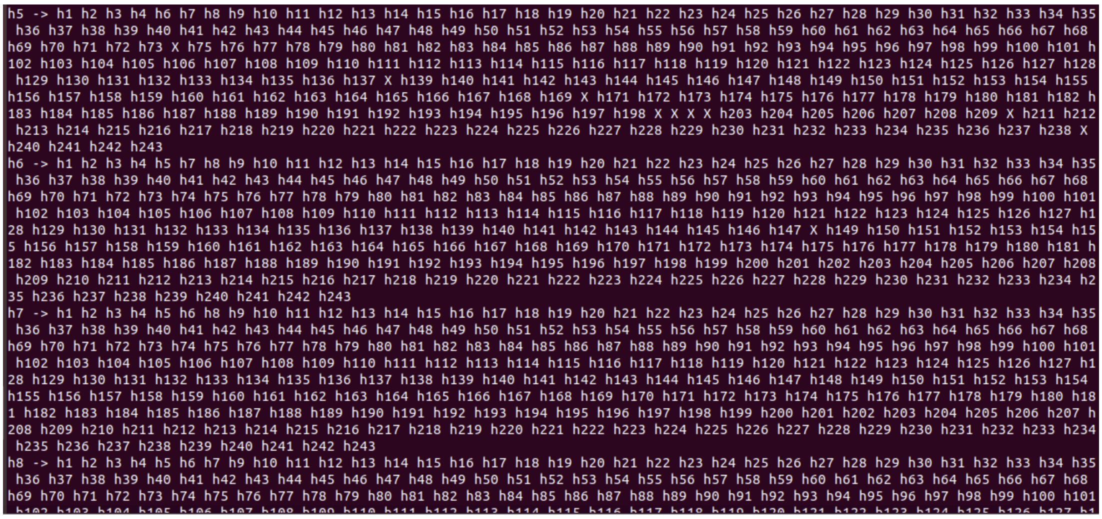
V. BULGULAR

A. Pingall Komutuna Göre Sonuçlar

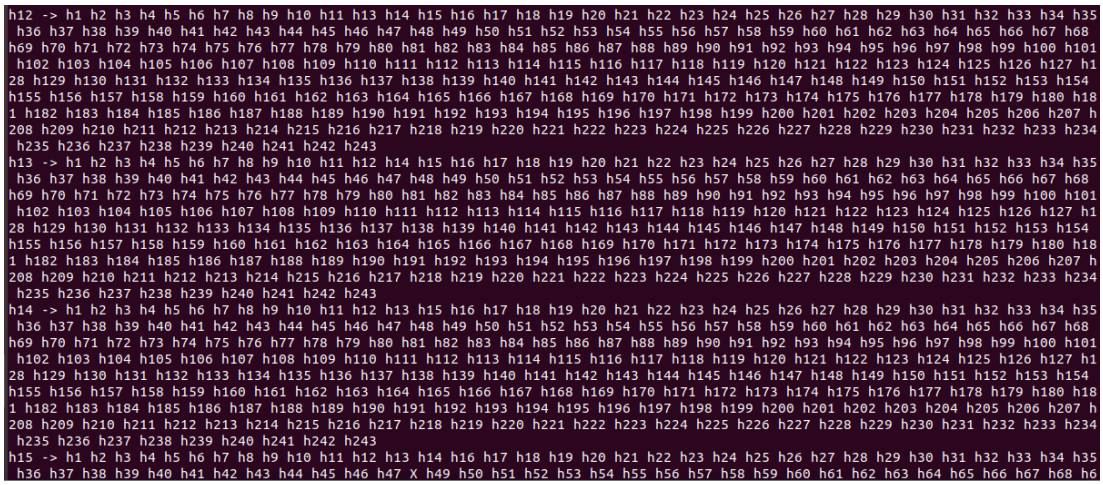
Burada şekil 63'te gösterildiği gibi ağ bağlantısı genel olarak test edilmiştir.



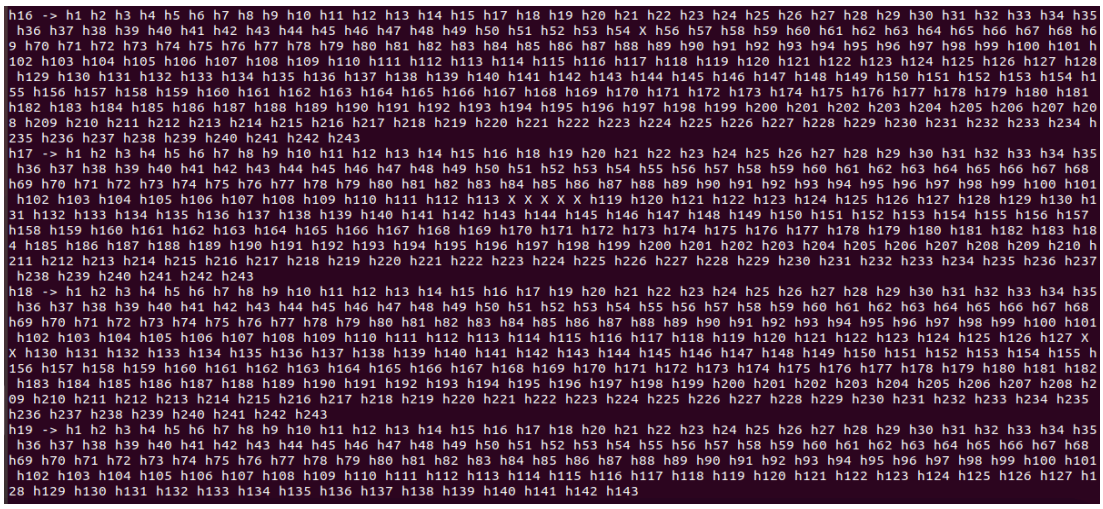
Şekil 63:h1 ana makinesinde Pingall komutunun sonucu



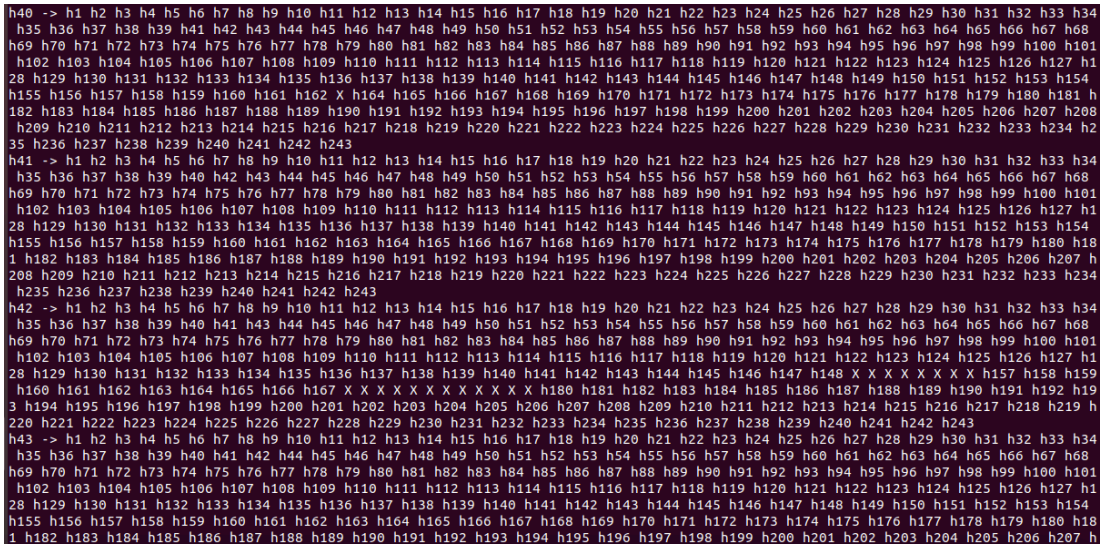
Şekil 64:h5 ana makinesinde Pingall komutunun sonucu



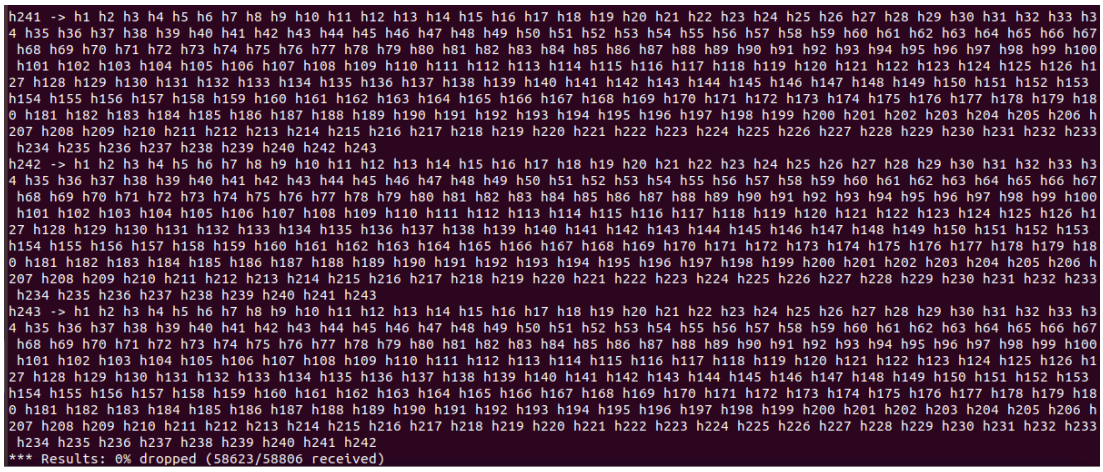
Şekil 65:h12 ana makinesinde Pingall komutunun sonucu



Şekil 66:h16 ana makinesinde Pingall komutunun sonucu



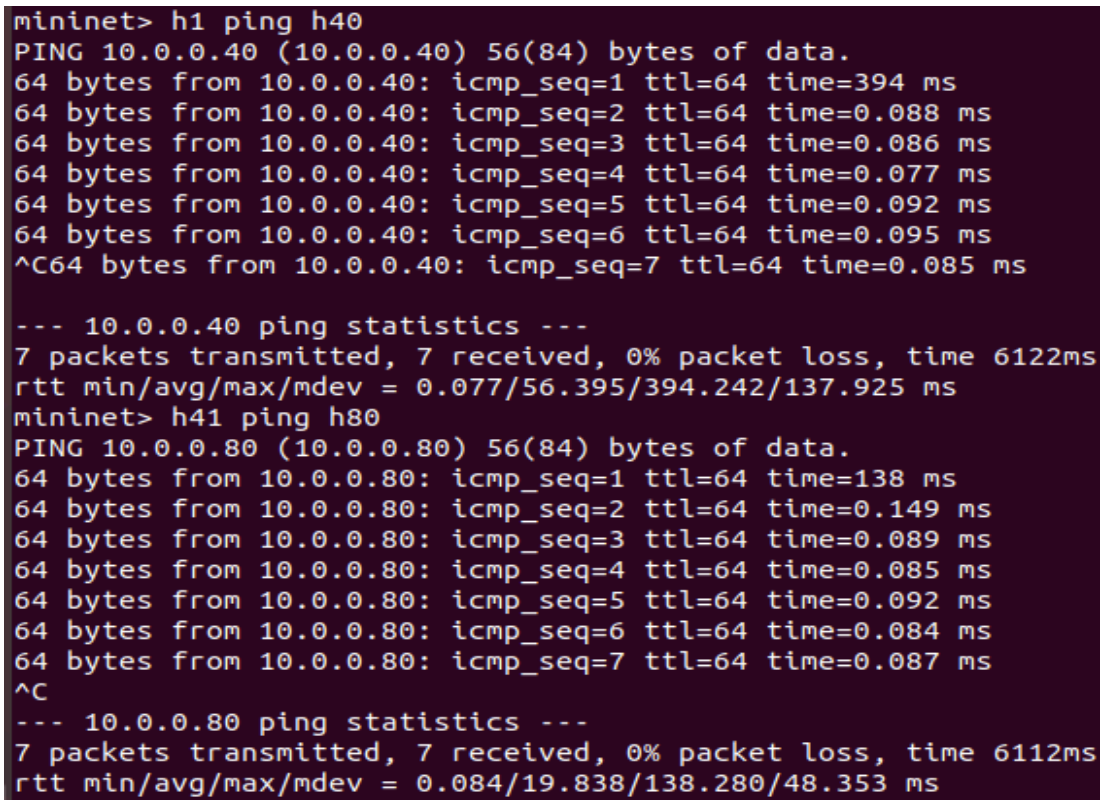
Şekil 67:h40 ana makinesinde Pingall komutunun sonucu



Şekil 68:h241 ana makinesinde Pingall komutunun sonucu

B. Ping Komutuna Göre Sonuçlar

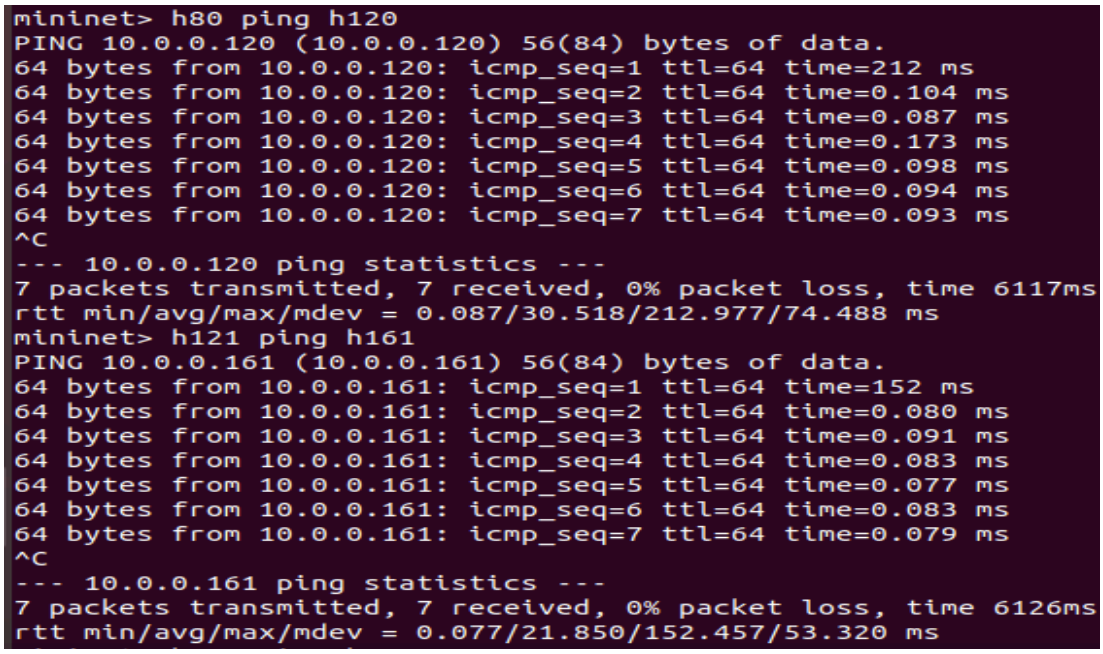
Şekil 63'te pingall komutunda linklerde sorun gösteren bağlantılar tekrar test edilmiştir. Şekil 69'da gösterildiği gibi birinci ve kırkıncı ana makine arasındaki bağlantı test edilmiştir. Aynı zamanda kırk birinci ve sekseninci ana makine arasındaki bağlantı da test edilmiştir.



Şekil 69: Ana makineler arasındaki bağlantı testi

Bu bölümde farklı ana makineler arasında bağlantı test edilmiştir. Şekil 70'da sekseninci ve yüz yirminci ana makineler arasında bağlantı test edilmektedir. Aynı

zamanda yüz yirmi birinci ana makine ve altmış birinci ana makine arasında bağlantı da test edilmiştir.



Şekil 70:h80 ile h120 ve h121 ile h161 arasındaki bağlantı testi

Ve son olarak da 162'inci ana makine ve iki yüzüncü ana makine arasındaki bağlantı test edilmiştir. Şekil 71'de görüldüğü üzeri bağlantıda bir sorun yoktur. Tüm test sonuçlarına dayanarak bağlantılarda bir sorun olmadığı görülmektedir.

```
mininet> h162 ping h200
PING 10.0.0.200 (10.0.0.200) 56(84) bytes of data.
64 bytes from 10.0.0.200: icmp_seq=1 ttl=64 time=232 ms
64 bytes from 10.0.0.200: icmp_seq=2 ttl=64 time=0.093 ms
64 bytes from 10.0.0.200: icmp_seq=3 ttl=64 time=0.099 ms
64 bytes from 10.0.0.200: icmp_seq=4 ttl=64 time=0.097 ms
64 bytes from 10.0.0.200: icmp_seq=5 ttl=64 time=0.096 ms
64 bytes from 10.0.0.200: icmp_seq=6 ttl=64 time=0.094 ms
64 bytes from 10.0.0.200: icmp_seq=7 ttl=64 time=0.092 ms
^C
--- 10.0.0.200 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6096ms
rtt min/avg/max/mdev = 0.092/33.281/232.396/81.288 ms
mininet> h201 ping h243
PING 10.0.0.243 (10.0.0.243) 56(84) bytes of data.
64 bytes from 10.0.0.243: icmp_seq=1 ttl=64 time=0.915 ms
64 bytes from 10.0.0.243: icmp_seq=2 ttl=64 time=0.085 ms
64 bytes from 10.0.0.243: icmp_seq=3 ttl=64 time=0.087 ms
64 bytes from 10.0.0.243: icmp_seq=4 ttl=64 time=0.084 ms
64 bytes from 10.0.0.243: icmp_seq=5 ttl=64 time=0.091 ms
64 bytes from 10.0.0.243: icmp_seq=6 ttl=64 time=0.081 ms
64 bytes from 10.0.0.243: icmp_seq=7 ttl=64 time=0.090 ms
^C
--- 10.0.0.243 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6125ms
rtt min/avg/max/mdev = 0.081/0.204/0.915/0.290 ms
```

Şekil 71:h162 ile h200 ve h201 ile h243 arasındaki bağlantı testi

C. Iperf Komutu Kullanılarak Bant Genişliği Ölçümü

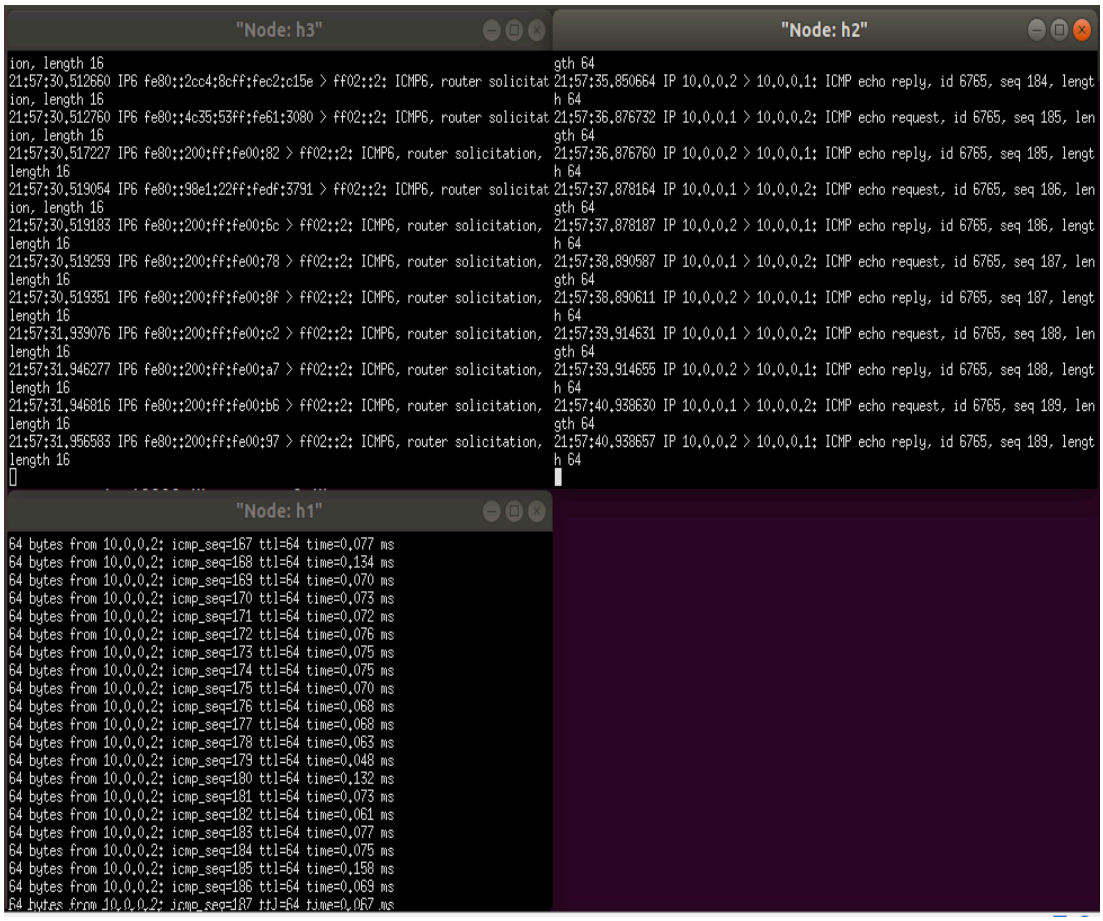
Bu aşamada da şekil 72'de gösterildiği gibi genel olarak bant genişliği ölçülmüştür.

```
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h243
*** Results: ['9.92 Gbits/sec', '9.92 Gbits/sec']
```

Şekil 72:İperf komutunun sonucu

D. Tcpdump Komutu Kullanılarak Paketleri Dinlemeye Almak

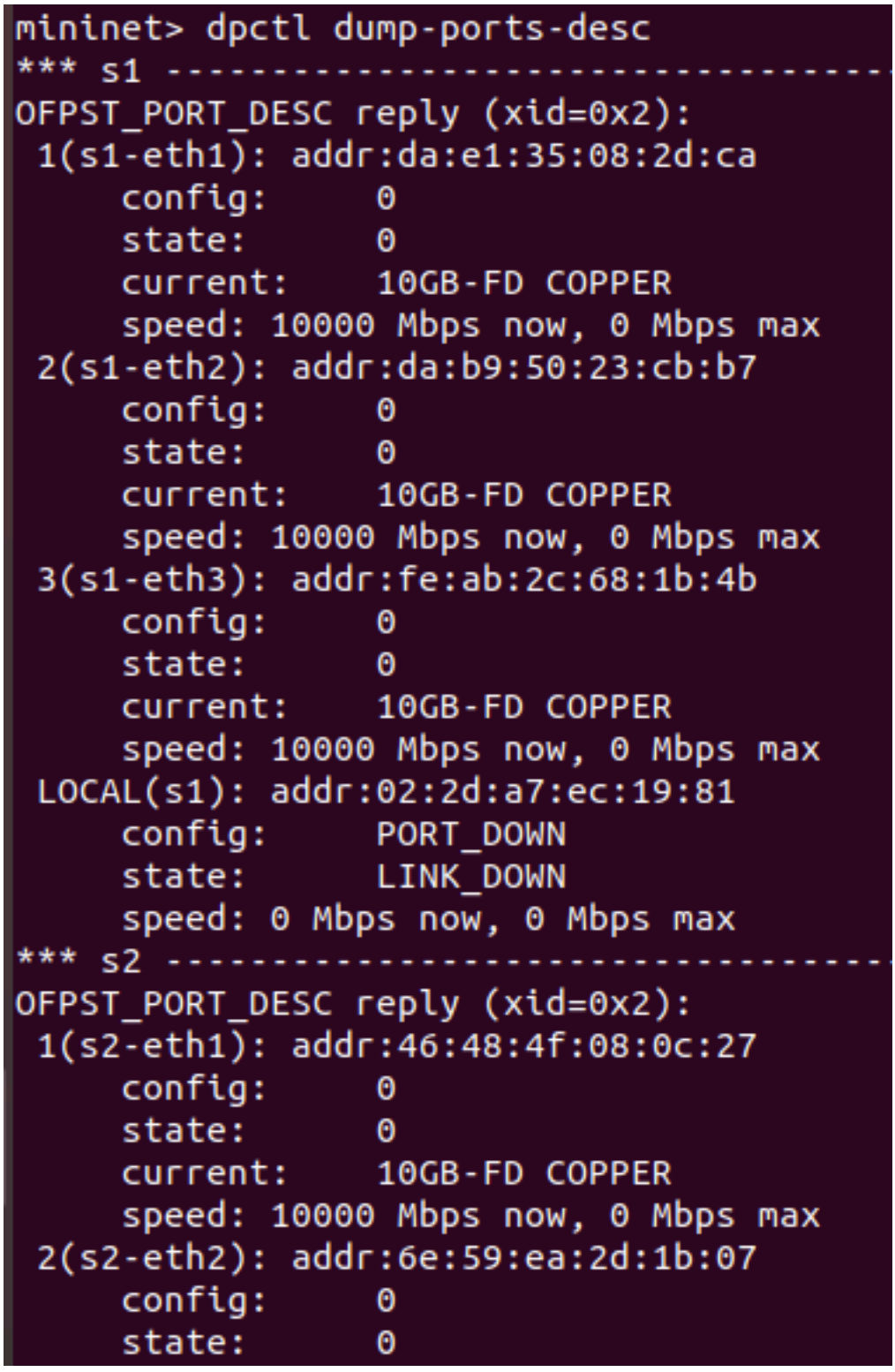
Bu aşamada birinci, ikinci ve üçüncü ana makineler arasındaki bağlantı dinlemeye alınmıştır. Bunun için ilk önce pox kontrolörünü başlatmak gerekmektedir. Daha sonra sudo komutu kullanarak bir yazılım tanımlı ağ oluşturulmuştur. Sonra h1'den h2'ye ping atılmaktadır. Aynı zamanda üçüncü ana makinenin de bu trafiği alıp almadığı kontrol edilmiştir. Daha sonra ise şekil 73'te de görüldüğü üzere sonuç vermiştir (Configuring POX controller as learning switch with Mininet:SDN laboratory, 2017).



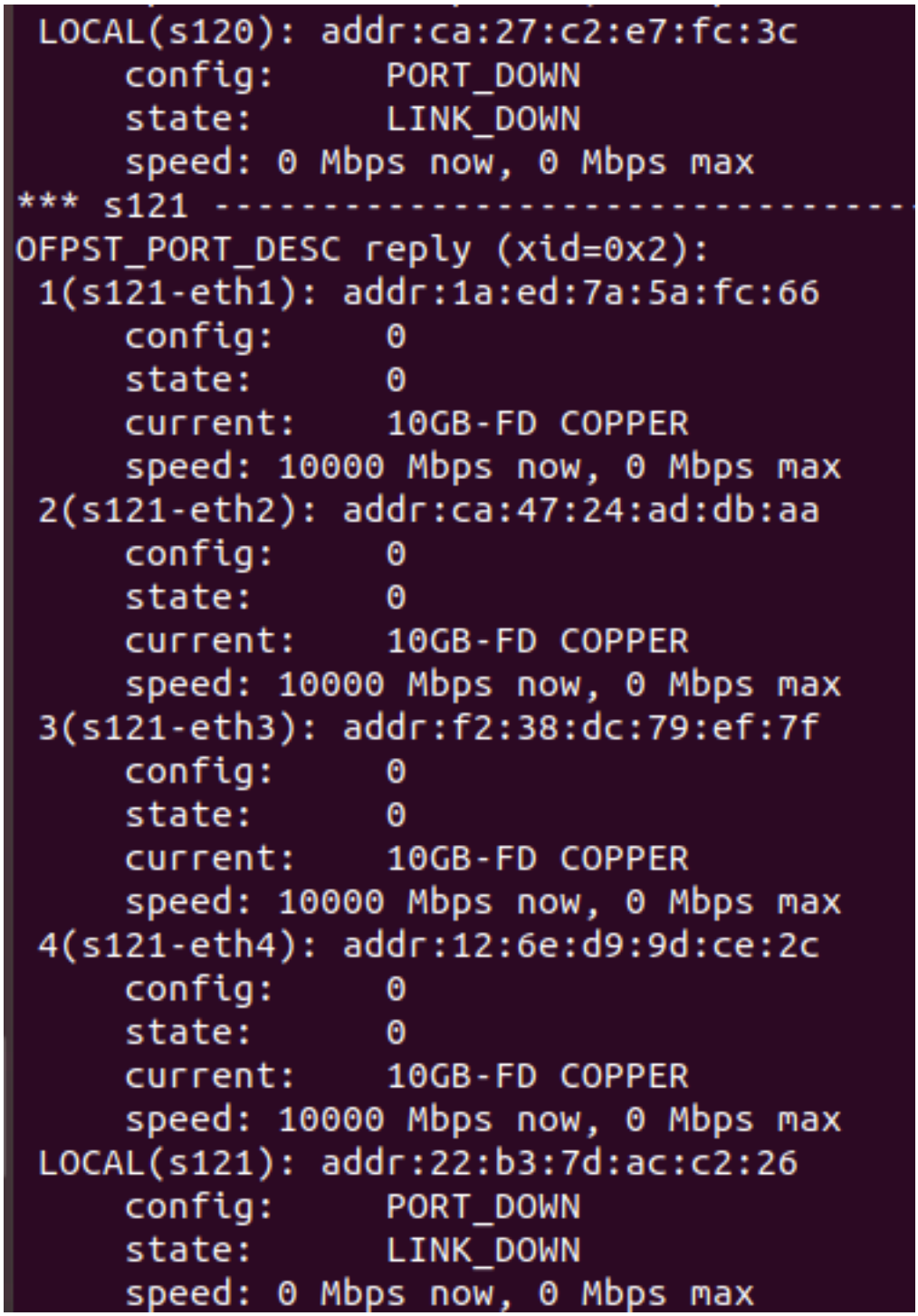
Şekil 73:Tcpdump komutunun sonucu

E. Dpctl Komutuna Göre Sonuçlar

Anahtardaki akışların listesini kontrol etmek için ‘dpctl dump-flows’ komutu kullanılmakta olup, sonucu şekil 74’te gösterilmiştir.



Şekil 74:Dpctl komutunun sonucu



Şekil 75:Dpctl komutuyle anahtardaki akış kontrolü

VI. BULGULARIN ANALİZİ

Bu bölümde elde edilen ping verileri tablo iki vasıtasıyla analiz edilmektedir.

Çizelge 2: Elde edilen sonuçların analizi

Ana makineler	Gönderilen Paketler	Alınan Paketler	Paket Kaybı	Minimum Zaman
h1-h40	7	7	0	0.077ms
h41-h81	7	7	0	0.084ms
h1-h68	7	7	0	0.077ms
h80-h120	7	7	0	0.087ms
h121-h161	7	7	0	0.077ms
h162-h200	7	7	0	0.092ms
h201-h243	7	7	0	0.081ms

Çizelge ikide görüldüğü gibi tüm ağda gönderilen ve alınan paket sayısı eşittir. Aynı zamanda paket kaybı da sıfır olup ve zaman dilimleri fark göstermektedir.

VII. SONUÇ VE ÖNERİLER

Test üç tür ağ üzerinde yapılmıştır. Birinci tür ağ tek kontrolü ağ ve ikinci tür ağ ise üç kontrolü ağ ve son ağ ise komut yöntem ile oluşturulmuştur.

Ağlardan elde edilen sonuçlara göre, h1 ve h40 ana makineleri arasında gönderilen ve alınan paket sayısı yedi ve minimum zaman dilim 0.077 mikro saniyedir. Dolayısıyla h41 ve h81 arasında gönderilen ve alınan paketlerde yedi ve minimum zaman dilimi 0.084 mikro saniyedir. Aynı zamanda h1 ve h68 arasında gönderilen paketlerin sayısı da yedi ve minimum zaman dilimi 0.077 mikro saniyedir. h80 ve h120 arasında gönderilen ve alınan paket sayısı ise yedi ve minimum zaman dilimi 0.087 mikro saniyedir. h121 ve h161 arasında gönderilen ve alınan paket sayısı yedi ve minimum zaman dilimi 0.077 mikro saniyedir. h162 ve h200 arasında gönderilen paket sayısı da yedi ve minimum zaman dilimi ise 0.092 mikro saniyedir. Son olarak, h201 ve h243 arasında gönderilen ve alınan paket sayısı gene yedi olup ve minimum zaman dilimi ise 0.0 81 mikro saniyedir.

Bu projenin uygulaması MacOs üzerinde VirtualBox ve Ubuntu 18.04 LTS işletim sistemi kullanılarak yapılmıştır. Eğer başka işletim sistemi kullanılacaksa elde edilecek sonuçlar değişebilir. Bir yazılım tanımlı ağ tasarlamak amaçlandığında, komut veya grafiksel arayüz yöntemiyle mininet kullanılabilir. Ağ oluşturulması ve özelliklerinin kontrol edilmesi mininet aracılıyla nispeten kolaydır. Ayrıca programlama ve ağ bilgisine sahip olanlar için kullanımı basittir.

VIII. KAYNAKÇA

KİTAPLAR

CHING-HAO, C. & LİN, D. Y.-D., (2015). **Internet Of Things for Industry and Human Applications**, Ukraine, Cilt 2.

GENG, H., (2015). **Data Center Handbook**. Palo Alto, CA, USA: John Wiley & Sons,Inc.,1.Baskı.

GÖRANSSON, P. & BLACK, C., (2014).**Software Defined Networks: A Comprehensive Approach**. Waltham(massachusetts): Elsevier Inc., 1.Baskı.

GÖRANSSON, P. & BLACK, C., (2014).**Software Defined Networks: A Comprehensive Approach.**Waltham(Massachusetts): Elsevier Inc.,1.Baskı.

GÖRANSSON, P. & BLACK, C., (2014). **Software Defined Networks: A Comprehensive Approach.**Waltham(Massachusetts): Elsevier Inc., 1.Baskı.

GÖRANSSON, P. & BLACK, C., (2014).**Software Defined Networks : A Comprehensive Approach.** Waltham(Massachusetts): Elsevier Inc., 1.Baskı.

GÖRANSSON, P. & BLACK, C., (2014).**Software Defined Networks: A Comprehensive Approach.** Waltham(Massachusetts): Elsevier Inc., 1.Baskı

HANRAHAN, H., (2007). **Network Convergence: Services, Applications, Transport, and Operations Support**, John Wiley & Sons, Ltd, 1.Baskı.

JAVVİN, (2004-2005). **Network Protocols Handbook**, Saratoga(California), Javvin Technologies Inc, 2.Baskı.

MİNASİ, M., ANDERSON, C., SMİTH, B. M. & TOOMBS, D., (2001).**Mastering Windows 2000 Server**, Alameda,CA: SYBEX,Inc., 3.Baskı.

NADEAU, T. D. & GRAY, K., (2013). **SDN: Software Defined Networks:An Authoritative Review of Network Programmability Technologies,** Sebastopol(CA): O'Reilly Media Inc,1.Baskı.

NADEAU, T. D. & GRAY, K., (2013). **SDN: Software Defined Networks:An Authoritative Review of Network Programmability Technologies,** Sebastopol(CA): O'Reilly Media Inc,1.Baskı.

NADEAU, T. D. & GRAY, K., (2013).**SDN: Software Defined Networks:An Authoritative Review of Network Programmability Technologies**, Sebastopol(CA): O'Reilly Media Inc,1.Baskı**.**

NADEAU, T. D. & GRAY, K., (2013).**SDN:Software Defined Networks:An Authoritative Review of Network Programmability Technologies**, Sebastopol(CA): OREILLY Media Inc,1.Baskı.

NADEAU, T. D. & GRAY, K., (2013).**SDN: Software Defined Networks:An Authoritative Review of Network Programmability Technologies**. Sebastopol(CA): O'Reilly Media, Inc., 1.Baskı.

OREBAUGH, A. VE DİĞERLERİ, (2007)."**Wireshark & Ethereal Network, Protocol Analyzer Toolkit**",Rockland, Syngress Publishing,Inc,2.Baskı.

PETERSEN, R., **Ubuntu 18.04 LTS Desktop:Application and Administration**, Alameda: Surfing Turtle Press,13.Baskı.

STALLİNGS, W., (2016,2013,2010). **Computer Organization and Architecture: Designing for Performance**, Hoboken(New Jersey), Pearson Education, Inc.,10.Baskı.

WARD, B., (2002). **The Book of VMware—The Complete Guide to VMware Workstation**. Canada: No Starch Press Inc,1.Baskı.

MAKALELER

CASADO, M. VE DİĞERLERİ, (2006). "SANE: A Protection Architecture for Enterprise Networks", **Proceedings of the 15th USENIX Security Symposium (Security '05)**, ss. 137-151.

CASADO, M., KOPONEN, T., MOON, D. & SHENKER, S., (2008). "Rethinking Packet Forwarding Hardware", Calgary, Alberta, Canada, **Sigcomm hotnets**, ss.2.

CASTILLO, A. C., (2018). "Simulation of Software Defined Network with Open

Network Operating System and Mininet", **International Journal of Computer Science & Information Technology** ,Ekim , cilt 10, sayı 5, ss. 8.

FERNANDEZ, M. P., (2013). "Comparing OpenFlow Controller Paradigms Scalability: Reactive and Proactive",**IEEE**, ss. 1009-1016.

GODA, K., M. I., KİTSUREGAWA, M. & S. M. I., 2012. "The History of Storage Systems", **Proceedings of the IEEE**, 13 Mayıs, Cilt 100, ss. 1437.

GUDE, N. VE DİĞERLERİ, (2008). "NOX: Towards an Operating System for Networks", **ACM SIGCOMM Computer Communication Review**, Temmuz, cilt 38, sayı 3, ss.1-6.

HOLIK, F. & KARAMAZOV, S., (2016). "Modeling Tool of TRILL Protocol" **Journal** of **Telecommunication, Electronic and Computer Engineering**, Cilt 9,Sayı (1-3), ss. 72.

HORİUCHİ, S., AKASHİ, K., SATO, M. & KOTANİ, T., (2017). "Network Resource Management Technology" **NTT Technical Review,** Ekim , Cilt 15, Sayı 10, ss. 2-3.

LANDRECHT, T. VE DİĞERLERİ, tarih yok." BUS 478: Seminar on Business Strategy" **SYNOPSIS,** ss. 1.

LARA, A., KOLASANİ, A. & RAMAMURTH, B., (First 2014). "Network Innovation using OpenFlow: A Survey" **IEEE COMMUNICATIONS SURVEYS & TUTORIALS,** Cilt 16, Sayı 1, ss. 493-512.

MANZANARES, J. F. G., PACHÓN DE LA CRUZ, A. & MADRİD MOLİNA, J. M., (2018). "Performance of QoS Policies in Software-Defined Networks. Guadalajara", **IEEE**.

MEYER, D. VE DİĞERLERİ, (2007). "Locator/ID Separation Protocol Locator/ID Separation Protocol (LISP) Tutorial", **IETF Vancouver**.

PELOSİ, S., (2006). "Network Management Systems for Overseas Solution" **FUJITSU Science and Technology Journal**, Cilt 42, Sayı 4, ss. 476-482.

ELEKTRONİK KAYNAKLAR

ADMİN,"Datagram nedir?", Bilgisayar Terimleri, http://bilgisayarterimleri.org/bilgisayar-aglari/datagram-nedir/, (Erişim Tarihi: 1.10. 2019)

ANON., "Routing Definition", LINFO, http://www.linfo.org/routing.html, (Erişim Tarihi : 9 .10. 2019)

ANON., "Apache CloudStack: About", Apache CloudStack, https://cloudstack.apache.org/about.html, (Erişim Tarihi : 5 Ekim 2019).

ANON., "THE LINUX FOUNDATION PROJECTS", Open Daylight, https://www.opendaylight.org , (Erişim Tarihi : 30.09.2019)

ANON.,"Policy and Charging Rules Function (PCRF)", STL25, https://www.stl.tech/french/policy-and-charging-rules-function-pcrf.html , (Erişim Tarihi : 5 Ekim 2019)

ANON.,"Generic Routing Encapsulation (GRE)" Juniper Networks, https://www.juniper.net/documentation/en_US/junos/topics/topic-map/switches-interface-gre.html,(Erişim Tarihi : 2 Ekim 2019)

ANON,"Mininet Walkthrough",Mininet Team,http://mininet.org/walkthrough/ (Erişim Tarihi : 10.11. 2019).

ANON., "Open Networking Foundation", ONF, https://www.opennetworking.org/ , (Erişim Tarihi : 8.10.2019)

ANON., "opennetworking" ,ONF, https://www.opennetworking.org/about/onf-overview, (Erişim Tarihi : 30.09.2019)

ANON., "Rapid NETCONF controller integration testing", Lighty, https://lighty.io/netconf-performance-test/,(Erişim Tarihi : 5.10.2019)

ANON., "RECAP", Recap Project, https://recap-project.eu/news/multi-tenant-colocation-data-centers/, (Erişim Tarihi : 28.09.2019)

ANON, "VirtualBox nedir,nasıl yüklenir?", technous.net, https://technous.net/virtualbox-nedir-nasil-yuklenir/ , (Erişim Tarihi : 7 Kasım 2019)

ANON., "WHAT IS OPENSTACK?",Software, https://www.openstack.org/software/,(Erişim Tarihi : 5.10. 2019)

BAHADUR, N., KİNİ, S. & MEDVED, J., 2018. "Routing Information Base Info Model",Tools IETF , https://tools.ietf.org/id/draft-ietf-i2rs-rib-info-model-17.html, (Erişim Tarihi : 30 Eylül 2019)

CONTİNİ, A.,"Software-defined-networking-fundamentals part 1 intro to networking planes",opendaylight,

https://www.opendaylight.org/blog/2016/11/16/software-defined-networking-fundamentals-part-1-intro-to-networking-planes , (Erişim Tarihi : 30.09. 2019)

DARREN, "rib, fib, lfib, lib etc.", Mellowd,https://mellowd.co.uk/ccie/rib-fib-lfib- lib-etc/,(Erişim Tarihi : 2.10.2019)

DEMİCOLİ, C., "Beginner's Guide to Understanding BGP" ,cdemi, https://blog.cdemi.io/beginners-guide-to-understanding-bgp/ (Erişim Tarihi : 1.10. 2019)

DİERKS, T. & RESCORLA, E., tools ietf, "The Transport Layer Security (TLS) Protocol",tools ietf,https://tools.ietf.org/html/rfc5246, (Erişim Tarihi : 11 Ekim 2019.)

ERGUN, O., "What Is OAM – Operation, Administration, Maintenance ?", orhanergun, https://orhanergun.net/2015/05/what-is-oam-operation-administration-maintenance/, (Erişim Tarihi : 2.10. 2019)

FAİRHURST, G.,"Difference between Unicast, Broadcast and Multicast in Computer Network"erg abdn,https://erg.abdn.ac.uk/users/gorry/course/intro-pages/uni-bmcast.html, (Erişim Tarihi : 2 Ekim 2019)

FOSKETT, S. & RAT, P., " Virtual Machine Mobility: Of What, and to Where and in What State?", fosketts, https://blog.fosketts.net/2012/01/16/virtual-machine-mobility-state/, (Erişim Tarihi : 28.10.2019)

IZARD, R., "How to Add an OpenFlow Experimenter Extension", project floodlight, https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/ pages/ 22380593/How+to+Add+an+OpenFlow+Experimenter+Extension #How toAddanOpenFlowExperimenterExtensionDefiningthe OpenFlow Protocol in Loxigen, (Erişim Tarihi : 9.10.2019)

KAHYA, O., "Ağ katmanı: introduction", osmankahya, https://osmankahya.net/bilgisayar-aglari/39-ag-katmani-introduction.html, (Erişim Tarihi : 2 Ekim 2019)

KHAN, M. I., "chmod command linux", GeeksforGeeks, https://www.geeksforgeeks.org/chmod-command-linux/, (Erişim Tarihi : 12.05.2019)

KNOW-HOW, "MAC address (media access control)", jonos,

ionos.co.uk/digitalguide/server/know-how/mac-address/
(Erişim Tarihi : 2.10. 2019)

LİNNERZ, S., "overvıew of junıper networks contraıl",xantaro, https://www.xantaro.net/en/tech-blogs/contrail/ , (Erişim Tarihi : 5 Ekim 2019)

PARLAKYİGİT, M., "hypervisor nedir ve hypervisor türleri ", parlakyigit, https://www.parlakyigit.net/hypervisor-nedir- ve-hypervisor-trleri/, (Erişim Tarihi : 24.09.2019)

PEPELNJAK, I., "Nicira NVP Control Plane",ipspace, https://blog.ipspace.net/2013/08/nicira-nvp-control-plane.html, (Erişim Tarihi : 5 Ekim 2019)

PEPELNJAK, I.,"What Is EVPN?", ipspace, https://blog.ipspace.net/2018/05/what-is-evpn.html,(Erişim Tarihi : 2 Ekim 2019)

PROJECT, S. C. S., "Ethane: A Security Management Architecture", stanford, http://yuba.stanford.edu/ethane/ (Erişim Tarihi : 7.10.2019).

RAO, S., "SDN Series Part Five: Floodlight, an OpenFlow Controller", the new stack, https://thenewstack.io/sdn-series-part-v-floodlight/, (Erişim Tarihi : 5 Ekim 2019).

RİVENES, L., "What is Network Throughput?", datapath, https://datapath.io/resources/blog/what-is-network-throughput/ (Erişim Tarihi : 1.10.2019)

SALİSBURY, B.,"The Control Plane, Data Plane and Forwarding Plane in Networks", networkstatic, http://networkstatic.net/the-control-plane-data-plane-and-forwarding-plane-in-networks/, (Erişim Tarihi : 1 Ekim 2019)

SEYİR DEFTERİ, "İnternet'in Tarihçesi",itübidb: bilgi işlem daire başkanlığı", https://bidb.itu.edu.tr/seyir-defteri/blog/ 2013/ 09/ 07/ internet'in-tarihçesi, (Erişim Tarihi : 2.10.2019)

SEYİR DEFTERİ, "mpls",itübidb:bilgi işlem daire başkanlığı, http://bidb.itu.edu.tr/seyir-defteri/blog/2013/09/06/mpls-(multi-protocol-label-switching---%C3%A7oklu-protokol-etiket-anahtalama) (Erişim Tarihi : 24.09. 2019)

TAŞDEMİR, C.,"Gömülü Sistemler, Elektronik, Teknoloji" , coskuntasdemir, http://coskuntasdemir.net/gomulu-yazilimlar/donanim-soyutlama-katmani-

hardware-abstraction-layer-nedir.html,(Erişim Tarihi : 30 Eylül 2019).

TECHLİBRARY, "Understanding Management Interfaces", Juniper Networks, https://www.juniper.net/documentation/en_US/junos/topics/concept/interface-management-understanding.html,(Erişim Tarihi : 30 Eylül 2019)

TUTAR, M. M., "BGP Protokolü Nedir?" , ipuçları,https://kod5.org/bgp-protokolu-nedir/, (Erişim Tarihi : 30.09.2019)

WHİTE, R., HARES, S. & VİGOUREUX, M., "Interface to the Routing System (i2rs)",ietf Data Tracker, https://datatracker.ietf.org/wg/i2rs/about/, Erişim Tarihi : 30.09.2019)

YAZAR, S., 2013. "Ağ Anahtarlarında OpenFlow Protokolü ve Pox Denetleyicisi Kullanımı", Bilgisayar Mühendisliği Ana Bilim Dali,Trakya Üniversitesi.

ZHAO, L., HARİS, N. & SAEED, G.,Department of information Technology, https://www.it.uu.se/edu/course/homepage/sakdat/ht07 / pm/programme/ VPN.pdf,(Erişim Tarihi : 30 Eylül 2019)

TEZLER

AL-ANİ, L. A., (2015). "Integratıng IP Protocol Into Optıcal Networks By Using Software-Defined Network (Sdn)", Electrical Engineering and Computer Science,University of Ottawa.

HARSH, S., (2016). "Evaluation of Multiple Controller based Software Defined Networks Architecture over Single Controller Software Defined Architecture", Department of Engineering Technology, University of Houston.

ONGARO, F., (2013-2014). "Enhancing Quality of Service in Software-Defined Networks", Department of Computer Science and Engineering, University Of Bologna.

ONGARO, F., (2013-2014). "Enhancıng Qualıty Of Servıce In Software-Defined Networks", Department of Computer Science and Engineering, University Of Bologna.

ONGARO, F., (2013-2014). "Enhancıng Qualıty Of Servıce In Software-Defined Networks", Department of Computer Science and Engineering, University Of Bologna.

YAZAR, S., 2013. "Ağ Anahtarlarında OpenFlow Protokolü ve Pox Denetleyicisi Kullanımı", Bilgisayar Mühendisliği Ana Bilim Dali,Trakya Üniversitesi.

DİĞER KAYNAKLAR

ANON, (2008). "Erişim Denetim Listeleri Oluşturma", Mili Eğitim Bakanlığı, Ankara: MEGEP.

ANON, (2012). "OpenFlow Ports",The Open Networking Foundation.

ANON,(2016). "Charter: Architecture Working Group (ArchWG)", Open Networking Foundation.

ANON, (2012). "OpenFlow ports",The Open Networking Foundation.

BHOWMİK, S. VE DİĞERLERİ, (2015). "Distributed Control Plane for Software-defined Networks: A Case Study Using Event-based Middleware", Oslo,Norway, ACM, ss. 3-4.

CONGDON, P., (2003). "Link Layer Discovery Protocol", St Louis: IEEE.

FEDYK, D., (2012). "Introduction to Shortest Path Bridging IEEE 802.1aq", NetNod.

KAPLAN, Y., (2008). "mpls (Multi Protocol Label Switching)"

KELLY, B., tarih yok. "Rackspace Case Study", United States: ForcePoint.

KESDEN, G., (2014). "Software Defined Networking (SDN)", Genova: Talk @ IEIIT Consiglio Nazionale delle Ricerche (CNR).

KESDEN, G., (2014). "Software Defined Networking (SDN)", Genova: Talk @ IEIIT Consiglio Nazionale delle Ricerche (CNR).

MCKEOWN, N., (2010). "OpenFlow Switch Specification".

SHERWOOD, R. VE DİĞERLERİ, (2009). "FlowVisor: A Network Virtualization Layer",OPENFLOW-TR-2009-1.

TONG, A. & WADE, K., (2017). "NFV and SDN Guide for Carriers and Service Providers", Hanover: Ciena.

VMWARE,(2006). "Virtualization Overview".

CERTBROS,(2018). "OSPF Explained | Step by Step"(video), 2018.

FUTUROLOGY, (2017)."History of Computing",Video,2017.

GORMAN F.,(2015)."Mininet setup from start to finish on Ubuntu 14.04 virtual machine", (video), 2015.

HOWTO.(2017). "Configuring POX controller as learning switch with

Mininet:SDN laboratory" ,Video, 2017.

HOWTO.(2017)."Configuring POX controller as learning switch with Mininet:SDN laboratory", Video, 2017.

PRATAMA, Y.A,(2017). "Mininet Part 2", Video, 2017.

ÖZGEÇMİŞ

Ad-Soyad **:** Habiba Amed

Doğum Tarihi ve Yeri **:** 20.11.1985 Faryab

E-posta **:** habiba_amed@yahoo.com

ÖĞRENİM DURUMU:

- **Lisans** **:** 2012 , Kabul Üniversitesi, ICT Fakültesi, ICT Bölümü
- **YüksekLisans** : 2020 , İstanbul Aydın Üniversitesi , Lisansüstü Eğitim Enstitüsü , Bilgisayar Mühendisliği YüksekLisans

MESLEKİ DENEYİM VE ÖDÜLLER:

Value added services/ Etisalat Afghanistan : 01.01.2012-29.03.2012

NSS Engineer/Afghan Telekom Şirketi : 15.04.2013-30.10.2013

Öğretim Üyesi/Kabul Politeknik Üniversitesi : 08.02.2014 Bu Zamana Kadar

En İyi Çalışan Takdir Belgesi : 30.12.2015 Kabul Politeknik Üniversitesi

TEZDEN TÜRETİLEN YAYINLAR, SUNUMLAR VE PATENTLER:

Amed, H. & Cevik, T., (2020). "Performance Evaluation of Software Defined Networks", **(IJCSIT) International Journal of Computer Science and Information Technologies**, 13 Mayıs, Mayıs-Haziran, Cilt 11 Sayı 3, ss. 31-36.

Amed, H. & Cevik, T., ”A Survey on Software Defined Networks”, Yayınlanmamış.