Karaelmas Fen ve Mühendislik Dergisi
Dergi web sayfası: http://fbd.beun.edu.tr

Derleme
Geliş tarihi / Received : 05.01.2017
Kabul tarihi / Accepted : 02.02.2017

Yazılım Tanımlı Ağlar – YTA

Software Defined Networks – SDN

Murtaza Cicioğlu[1*], Ali Çalhan[2]

[1]Düzce Üniversitesi, Elektrik-Elektronik ve Bilgisayar Mühendisliği, Düzce, Türkiye
[2]Düzce Üniversitesi, Teknoloji Fakültesi Bilgisayar Mühendisliği, Düzce, Türkiye

Öz

Bilgi ve iletişim teknolojilerindeki (bulut bilişim, sosyal ve mobil ağlar, veri merkezleri vb.) yeni gelişmeler geleceğin interneti için her yerde ve her an erişilebilirlik, yüksek bant genişliği, dinamik yönetim ortamları gibi yeni gereksinimleri de beraberinde getirmektedir. Manuel yapılandırma özelliğine sahip ağ cihazlarından oluşan geleneksel ağ yaklaşımı hantal ve hata eğilimlidir. Bununla birlikte fiziksel ağ altyapısının tamamen etkili ve verimli şekilde kullanılmasına da olanak tanımamaktadır. YTA (Yazılım Tanımlı Ağlar) bu anlamda geleceğin interneti için en umut verici çözümlerden biri olarak görülmektedir. YTA'nın en önemli iki ayırt edici karakteristik özelliği; geleneksel ağ mimarisinde tümleşik olan kontrol ve veri düzleminin birbirinden ayrılmasını ve ağ uygulaması geliştirmek için programlanabilirlik sağlamasıdır. Bunun bir sonucu olarak, YTA yenilikçi ağ tasarımlarını karşılayabilen, daha etkili yapılandırmaya sahip, daha iyi performans ve daha çok esneklik sağlayan yeni bir yaklaşım olarak karşımıza çıkmaktadır. Bu çalışmada son zamanlarda aktif ve popüler bir çalışma konusu olan YTA ile ilgili yapılan çalışmalar ve son gelişmeler kapsamlı bir şekilde incelenmiştir. İlk olarak geleneksel ağ yaklaşımı ile YTA yaklaşımı karşılaştırılmış ve YTA mimarisi ele alınmıştır. Daha sonra YTA yaklaşımının katman yapısı (ağ altyapısı, güney ara yüzü, ağ hypervisor, ağ işletim sistemi, kuzey ara yüzü, dil tabanlı sanallaştırma, programlama dilleri ve ağ uygulamaları) detaylı bir şekilde incelenmiştir. Son olarak gelecek için açık araştırma konuları önerilmiştir.

Anahtar Kelimeler: Ağ işletim sistemi, Ağ sanallaştırma, Programlanabilen ağ, Yazılım tanımlı ağlar (YTA), Yazılım tanımlı ortamlar, YTA katman yapısı

Abstract

New developments in information and communication technologies (cloud computing, mobile and social networks, data centers etc.) bring new problems for internet of the future as ubiquitous accessibility, high bandwidth, and dynamic management environments. The traditional network approach consists network devices which have manually configuration feature is cumbersome and error-prone. It also does not permit to use physical network infrastructure fully effective and efficiently. In this sense, SDN (Software Defined Networks) is seen as one of the most promising solutions for the internet of the future. The two most important distinguishing characteristic of SDN are; providing programmability for developing network applications and separating control and data planes which are integrated in the traditional network architecture. As a result, SDN emerges as a new approach which can meet innovative network design, has more efficient configuration, provides better performance and more flexibility. In this study, studies and latest developments about SDN, recently active and popular topic, were studied in a comprehensive manner. Firstly, traditional network approach and SDN approach compared and SDN architecture was discussed. Then layer structure of the SDN approach (network infrastructure, southbound interface, network hypervisor, network operation system, northbound interface, language-based virtualization, programming languages and network applications) is discussed in detail. Finally, open research topics have been proposed for the future works.

Keywords: Network operation system, Network virtualization, Programmable network, Software defined networking (SDN), Software-defined environments, SDN layer structure

*Sorumlu yazarın e-posta adresi: murtazacicioglu@gmail.com

Murtaza Cicioğlu orcid.org/0000-0002-5657-7402
Ali Çalhan orcid.org/0000-0002-5798-3103

1. Giriş

İnternet cihazların ve insanların birbiri ile iletişim içinde olduğu, dünyanın herhangi bir yerinden ulaşılabilen dijital bir toplum oluşumunu sağlamıştır. Her geçen gün hızla yaygınlaşan ve kullanımı artan internet, hızını kesmeden yaygınlaşmaya ve gelişmeye devam etmektedir. Mevcut geleneksel ağ mimarisi bu artan ve gelişmekte olan ihtiyaçları karşılamakta yetersiz kalmaktadır. Bunun yanı sıra geleneksel ağ mimarisi yönetimi oldukça zor ve karmaşıktır (Kreutz vd. 2015, Akyildiz vd. 2014).

Geleneksel ağ mimarisi, yönlendiriciler (router), anahtarlayıcılar (switch) ve birçok aracı internet aygıtları (middlebox) ile bu ağ cihazlarına tanımlı birçok karmaşık protokollerden oluşmaktadır (Nunes vd. 2014). Her yeni protokol bir takım sorunları çözerken, bununla birlikte daha karmaşık ve yönetilmesi daha zor ağ yapılarının oluşmasına sebep olmaktadır (Benson vd. 2009). Bunun yanı sıra, ağı yönetmek ve ağ üzerinde çeşitli uygulamalar yapmak için farklı üreticilerin ürettiği standart olmayan ağ cihazları, işletim sistemleri ve ağ denetim yazılımları kullanılmaktadır. Geleneksel ağ mimarisinde performans, güvenlik, yönlendirme gibi işlemler için mevcut altyapılara ara ağ cihazı olarak eklenen cihazların sayısı, ağ iskelet yapısı için kullanılan cihaz sayısına eşit seviyeye ulaşmıştır. Sisteme sonradan dahil edilen her bir ağ cihazı, yönetilmesi zor ve karmaşık yapıları artırmaktadır. Farklı üreticilerin standart olmayan ara ağ cihazları güvenlik açıklarına neden olmaktadır. Bu nedenle bahsedilen ağ problemlerinin üstesinden gelebilmek amacıyla yazılım tanımlı ağ kavramı ortaya atılmıştır.

Yazılım Tanımlı Ağlar -YTA (Software Defined Networks - SDN) son zamanlarda akademik dünyada ve sanayi topluluklarında büyük ilgi görmektedir. Günümüz ağ altyapısının sınırlılıklarını ortadan kaldırmaya çalışan YTA, geleneksel ağ mimarisinin altyapısında tümleşik olan veri (data plane) ve kontrol düzlemlerinin (control plane) ayrılmasını öneren (Feamster vd. 2014), ağ ve yönetim sistemlerinin basitleştirilmesini (Kreutz vd. 2015) ve otonom bir yapıya kavuşmasını sağlayan yeni bir ağ yaklaşımı olarak karşımıza çıkmaktadır. Böylece merkezden yönetilebilen bir kontrol düzlemi dinamik, yüksek performanslı, verimli ve geliştirilebilir bir iletişim altyapısı inşa edilebilmesini sağlamaktadır (Galinac Grbac vd. 2015). Bununla birlikte yeni güvenlik sorunlarını da beraberinde getirmektedir (Kaur vd. 2015).

Yazılım Tanımlı Ağlar, merkezi bir kontrol panel mimarisi ile standartlaştırılmış bir ara yüz aracılığıyla, tüm ağ altyapılarının birleştirilmesi ve yönetilebilmesini sağlamaktadır. Böylece mevcut ağ altyapıları üzerinde yapılacak değişikliklerin çok daha hızlı bir şekilde gerçekleştirilmesi sağlanmaktadır. Bunun yanı sıra, ağ altyapısı üzerinde oluşturulacak yeni teknolojilerin uygulanması daha kolay hale gelmektedir. Bu durum servis performansını ve kaynak kullanımını arttırmaktadır (Galinac Grbac vd. 2015). Ayrıca YTA, kontrol edilebilir, uçtan uca gerçek zamanlı takip edilebilir ve programlanabilir bir ağ altyapısı sunmaktadır. Çizelge 1'de yazılım tanımlı ağ ile geleneksel ağ yaklaşımları karşılaştırılmıştır.

YTA kavramı 1990'ların öncesine dayanmakla birlikte, 2006 yılında OpenFlow kavramıyla birlikte küresel bazda ilgi görmeye başlamıştır. YTA teknolojisi Open Network Foundation tarafından desteklenmiş ve standardize edilmiştir (Chen vd. 2015).

2. Gereç ve Yöntem/Yöntemler

2.1. Literatür Taraması

Literatürde, YTA konusunda çeşitli çalışmalar yapılmıştır. Çizelge 2'de, literatürde yapılan çalışmalar verilmiştir. Kim ve Feamster (2013) YTA tabanlı ağ yönetiminin gelişimi üzerine çalışma yapmışlardır. Bu çalışmada ağ yönetimini çeşitli yönlerden geliştirmek ve sorunlarını tespit etmek amacıyla mevcut ağ mimarisinin yapılandırılması ile yönetim mekanizmalarını incelemişlerdir. Ağ yönetimi üzerinde üç probleme odaklanmışlardır. Bunlar; sık sık değişen ağ şartları ve durumuna izin verilmesi, yüksek seviyeli bir dilde ağ yapılandırmasının sağlanması, ağ durum tespiti ve sorun çözümü için görevler üzerinden daha iyi görsellik ve kontrol sağlanmasıdır. Çalışmalarında ağ operatörlerine, geniş yelpazede yüksek seviyeli bir dil aracılığıyla ağ ilkelerine belirlemeye izin veren ve performans sorunlarının kaynağını kolaylıkla tespit eden teknolojiler tanımlamışlardır. Bunun yanı sıra YTA mimarisinin ortak ağ yönetim görevlerini nasıl arttırıldığını gösteren yerleşke ve ev ağlarında çeşitli prototip kurulumlar önermişlerdir.

Silva vd. (2014), tamamen mobil ve güvenilir sisteme yönelik yazılım tanımlı e-sağlık ağı üzerine çalışmışlardır. Yazılım Tanımlı Ağ mimarisine dayalı CAMA (Bağlam Duyarlı Mobil Yaklaşımı - Context-Aware Mobile Approach) yaklaşımında kalite odaklı mobilite kontrol becerisini etkinleştirmek, mobilite tahmin uygulaması önerilmiştir. Ayrıca bağlanma noktasına (Point of Attachment - PoA) karar veren ve hem uygulama kalite gerekliliklerini karşılamak ve hem de mevcut kablosuz kalite koşullarını

Çizelge 1. Yazılım tanımlı ağ ile geleneksel ağ yaklaşımlarının karşılaştırılması. (Software defined network and traditional network approaches compare) (Xia vd. 2015).

	Yazılım Tanımlı Ağlar	Geleneksel Ağlar
Özellikler	Veri ve kontrol düzlemlerini birbirinden ayırır ve programlanabilirlik	Her bir problem için yeni bir protokol, zor ve karmaşık ağ kontrolü
Yapılandırma	Merkezi doğrulama ile birlikte otomatik yapılandırma	Hata eğilimli manuel yapılandırma
Performans	Çapraz katman yapısı ile birlikte dinamik toplu kontrol	Sınırlı bilgi ve nispeten statik yapılandırma
Yenilik	Yenilikler için kolay yazılım uygulamaları, izolasyon ile yeterli test ortamı ve yazılım güncellemeleri kullanılarak hızlı dağıtım	Yenilikler için zor donanım uygulamaları, sınırlı test ortamı, uzun standardizasyon süreçleri

geliştirmek, bunun yanında e-sağlık biyo-geribildirim sistemlerinin güvenirliliği ve dayanıklılığını arttırmak için bir ağ altyapısı önermişlerdir.

Liu ve Li (2015) Geniş Alan Ağları'nda (Wide Area Networks) Yazılım Tanımlı Ağı ölçekleme üzerine çalışmışlardır. Yazılım Tanımlı Geniş Alan Ağları'nın ölçeklenebilirlik sorunlarını gidermek için kullanılacak mekanizmaları incelemişlerdir. Büyük çaplı ağ alt yapısı olan Domain Name Server (DNS) sistemlerinin ölçeklenebilir hale getirilmesi için temel unsurlar incelenmiştir. Bununla birlikte, literatürde önerilen mevcut teknolojileri üç kategoride ele almışlardır. Bunlar; veri düzlemini çoğaltma ve ölçeklendirme ile kontrol düzlemini ölçeklendirmedir. Yazılım Tanımlı Geniş Alan Ağları'nın ölçeklendirmesine yönelik olası çalışmalar önermişlerdir.

Grbac vd. (2015), YTA yaklaşımının yazılım teknolojilerinden talepleri üzerine çalışma yapmışlardır. Yazılım teknolojilerindeki yeni gelişmeler gerektiren bazı kritik kalite konuları ile YTA yaklaşımının mimari yapısı incelenmiştir. Ağ gelişiminin adresleme problemlerini, yazılım teknolojilerinin bu problemleri çözebilme durumları ile en temelde ağ ve yazılım teknolojilerinin güçlü entegrasyon ihtiyacına vurgu yapılmıştır. Chen vd. (2015) Yazılım Tanımlı Mobil Ağları üzerine çalışmış ve SDMN (Software Defined Mobile Networks) kavramını öne sürmüşlerdir. Radyo erişim ağları için yazılım tanımlı tasarım üzerine odaklanılarak, SDMN mimarisinin gereksinim ve ihtiyaçları ortaya konulmuştur. YTA tasarımında SDMN kavramı olarak sunulan mimari radyo erişim ağlarının temel sorunları temelinde analiz edilmiş, SDMN ve standardizasyon faaliyetleri için geçerli çözümler sunmuşlardır.

Costanzo vd. (2015) Yazılım Tanımlı Kablosuz Ağ (Software Defined Wireless Network –SDWN), Kablosuz Kişisel Alan Ağları (Wireless Personal Area Networks - WPANs) için evrimleşebilen mimari üzerine bir çalışma yapmışlardır. Kablosuz Kişisel Alan Ağları (W-PANs) gibi daha az altyapısı olan kablosuz ağ ortamlarına YTA mimarisinin avantajlarını, Kablosuz Kişisel Alan Ağları'na YTA mimarisini uygulayabilmek için yerine getirilmesi gereken gereklilikleri incelemişlerdir. Bu inceleme sonucunda Kablosuz Kişisel Alan Ağları için tam bir YTA çözümü sunan Yazılım Tanımlı Kablosuz Ağ (Software Defined Wireless Network - SDWN) adında bir prototip uygulaması geliştirmişlerdir. Jagadeesan ve Krishnamachari (2015) kablosuz ağların YTA yaklaşımı üzerine bir çalışma yapmışlardır. Benzer çalışmada YTA mimarisi kablosuz ağ bağlamında incelenmiş ve ana tasarım modelleri hakkında genel bilgiler verilmiştir.

Jain ve Raul (2013) bulut bilişim (Cloud Computing) için YTA ve ağ sanallaştırması üzerine bir çalışma yapmışlardır. Sanallaştırma nedenleri ile gelişmiş veya çeşitli standartlarda gelişen ağ teknolojileri hakkında genel hatlarıyla bilgiler sunulmuştur. Ağ programlanabilirlik bağlamında anahtar rol oynayan YTA yaklaşımı temel alınarak, açık uygulama dağıtım katmanı (Open Application Delivery Network - OpenADN) olarak adlandırılan sanallaştırılmış yeni bir oturum katmanı önermişlerdir.

Hu vd. (2012) yazılım tanımlı ağlarda denetleyici yerleşimi üzerine bir çalışma yapmışlardır. Ağ sorunları, kontrol ve yönlendirme düzlemleri arasında kolayca bağlantı kopmasına sebep olduğundan, bilgilendirilmiş yerleşim kararları almak adına YTA'ın güvenirliliğini maksimize etmek için kullanılabilecek çeşitli algoritmalar geliştirmişlerdir. Gerçek topolojiler kullanılarak değerlendirilen bu algoritmaların benzetim sonuçları, denetleyici yerleşim stratejilerinin YTA performansı için çok önemli olduğunu ve bir Greedy

algoritmanın optimale yakın yerleşim sonuçları sağladığını göstermektedir. Huang vd. (2014) çalışmalarında çoklu YTA sağlayıcılara yayılan YTA ağlarını kurmak için yeni bir yol önermişlerdir. Yaklaşımlarının anahtar teması Network Hypervisors servisidir. Network Hypervisors servisi, karmaşık YTA ağ servislerini oluşturma görevini büyük ölçüde basitleştiren API'ler ve yüksek seviye soyutlama sunmaktadır. Bunun yanı sıra Network Hypervisors servisi çeşitli YTA sağlayıcılarını, tek bir ara yüz aracılığıyla birlikte haberleştirme yeteneğine sahiptir. Böylece bir arada uygulamalar, YTA sağlayıcılar arasındaki farklılıkları görmek ve uğraşmak zorunda olmadan uçtan uca akışlar kurabilmektedir.

Kaur vd. (2015) YTA mimarisi için OpenFlow tabanlı programlanabilen güvenlik duvarı uygulaması önermişlerdir. YTA tabanlı ağların güvenliği için güçlü ve esnek güvenlik duvarı uygulamaları oluşturmayı amaçlamışlardır. Birçok güvenlik duvarı fonksiyonları özel bir donanıma gerek kalmadan yazılım kullanarak oluşturulabildiğini ifade etmişlerdir. Çalışmalarında açık kaynaklı Pyhton dilinde oluşturulmuş POX denetleyicisi kullanmışlardır. Uygulamalarında sanallaştırma çözümünü VMPlayer kullanarak, ağ topolojilerini oluşturmak için ise Mininet emülatörünü kullanmışlardır.

Wu vd. (2013) merkezileştirilmiş sunucu ve dağıtılmış aracılar içeren YTA yaklaşımı tabanlı IaaS (Bir Servis Olarak Openstack Altyapısı - Openstack İnfrastructure as a Service) bulutunda yeni bir programlanabilir sanal ağ örneklemesi önermişlerdir. Prototip uygulama, bu mimarinin makul olduğunu performans ve esneklik arasında tekli merkezileştirilmiş ve dağıtılmış çözümlere göre daha iyi uzlaşımlar sağlayabildiğini göstermektedir. Akyıldız vd. (2014) SDN-Openflow ağlarında trafik mühendisliği için bir yol haritası üzerinde çalışma yapmışlardır. Bu çalışmada YTA yaklaşımında trafik mühendisliğinin son durumu incelemiş ve akış yönetimi, hata toleransı, topoloji güncellemesi ve trafik analizi gibi ana konulara odaklanmışlardır. Bununla birlikte mevcut ve temsili trafik mühendisliği araçları açıklanmış, YTA trafik mühendisliği çözümleri detaylı olarak ele alınmıştır.

Soltani (2014) YTA'da akış başlatma sorununu gidermek için hibrit bir mekanizma önermiştir. Önerilen mekanizma da akış başlatma süreci içerisinde anahtarlar için etkili yönlendirici kararları almasını sağlayan önsel bilgi oluşturmak için denetleyicilerde bulunan birleşik ağ haritası kullanmıştır. Selvi (2015) hiyerarşik YTA denetleyicilerinde işbirlikçi yük dengeleme isimli bir yük dengeleme yöntemi önermiştir. Yazıcı (2015) kablosuz ağlar için yazılıma dayalı ağ yaklaşımı isimli bir çalışma yürütmüştür. Çalışmasında merkezi ağ servislerinde farklı başarım ve karmaşıklık seviyelerine ve 5G sistemlerde servis farklılaştırılmasına imkân sunan, hiyerarşik ağ kontrol donanımına sahip bütün bir YTA ağ mimarisi yaklaşımı önermiştir. Hareketlilik, hücreler arası transfer ve rotalama yönetimini tümleşik bir biçimde ele alan, bunun yanı sıra uygulama geliştiricileri ve servis sağlayıcıları için, üyelerin hareketliliğinin ağ üzerindeki akışlarına olan etkisinin ücretlendirmeye göre farklılaştırılmasına olanak sağlayan Servis Olarak Bağlantı Yönetimi (SBOY) yaklaşımı temel alınmıştır.

Görkemli vd. (2015) YTA ile servis kalitesi (QoS) yönetimi ve önceliklendirme üzerinde çalışmışlardır. YTA teknolojisinin, yüksek kaliteli ve kesintisiz (Qos) ses ve görüntü hizmetini internet protokolü üzerinden nasıl gönderileceğini, kayıp oranını %5'ten, gecikmenin ise %50'den fazla azaltarak QoS'in YTA paradigması kullanılarak daha esnek bir şekilde yönetilebileceğini önermişlerdir. Bu çalışmada YTA mimarisi için özel geliştirilmiş yazılımsal yönetici gerçeklemesi önerilmiştir. Ali vd. (2015) YTA kullanarak ağların güvenliği üzerine çalışma yapmışlardır.

Özçevik vd. (2015) YTA için yonga üstü iletişim ağının (NOC) temelli Banyan OpenFlow anahtarı tasarımı önermişlerdir. Çalışmalarında Banyan ağ yapısı kullanarak, eşleşme azaltılarak gerçekleştirilen akış yönlendirmesi sayesinde akışın servis kalitesindeki ek gecikmeyi azaltan ve OpenFlow anahtarının akılsızlık özelliğini akışın yönlendirileceği bir sonraki tablo belirteci ile koruyan yeni OpenFlow anahtarı tasarlamışlardır. Güner vd. (2015) YTA' da denetleyici yerleşimi üzerine çalışmışlardır. Çalışmalarında yazılım tanımlı mobil ağlarda denetleyici yerleştirme sorunlarını incelemişlerdir. Akyıldız ve Saygun (2015) YTA – ağ fonksiyon sanallaştırması (NFV) - bulut bilişim (Cloud Computing) teknolojilerinin servis sıralaması kapsamında incelemişlerdir. Çalışmalarında YTA, ağ fonksiyon sanallaştırması ve bulut bilişim teknolojileri ve birbirleri ile ilişkileri ifade edilmiş, daha sonra bu teknolojileri kullanan servis sıralama uygulamasına odaklanmışlardır.

2.2. Yazılım Tanımlı Ağ Yaklaşımı

Birçok araştırmacı ağ konusunda güvenirlik, hız, ölçeklendirme vb. birçok konuda yıllardır iyileştirme çalışmaları yapmaktadırlar. Ancak bu çalışmalar, sonucu sadece proje aşamasında kalmış, karmaşık ve yönetilmesi neredeyse imkânsız durumda olan günümüz ağ altyapısına uyarlamada başarılı olamamıştır. Yönlendirici ve anahtar gibi ağ cihazları

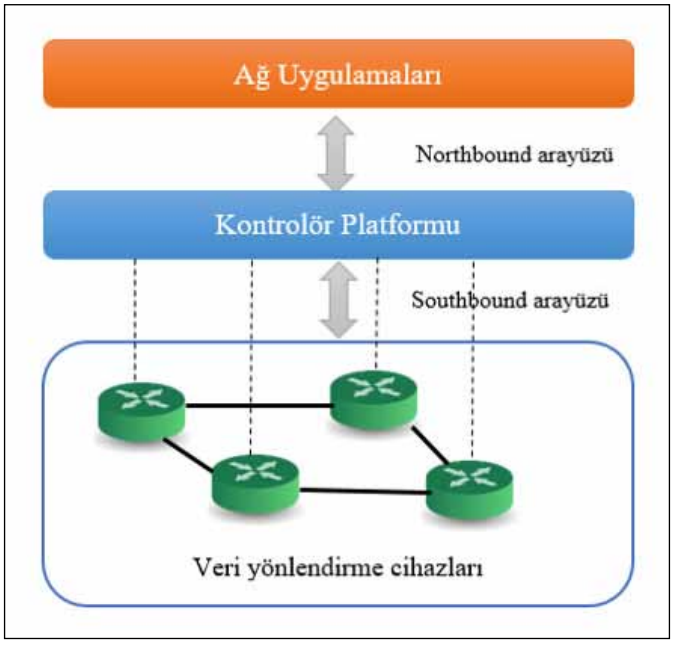
Çizelge 2. Literatürdeki YTA çalışmaları (Studies in the literatüre about SDN).

SDN konulu çeşitli araştırmalar	Öneriler	Kaynaklar
SDN yaklaşımının yazılım teknolojilerinden talepleri	-	Galinac Grbac vd. (2015)
Yazılım tanımlı mobil ağlar: SDMN (Software Defined Mobile Networks)	Software Defined Mobile Networks - SDMN (Survey)	Chen vd. (2015)
SDN ile ağ yönetiminin gelişimi	-	Kim ve Feamster (2013)
Tamamen mobil ve güvenilir sisteme yönelik yazılım tanımlı e-sağlık ağı	Software Defined eHealth Networking	Silva vd. (2014)
Geniş alan ağlarında yazılım tanımlı ağı ölçekleme	SDN in WAN	Liu ve Li (2015)
Yazılım tanımlı kablosuz ağ (Software Defined Wireless Network –SDWN), kablosuz kişisel alan ağları (Wireless Personal Area Networks - W-PANs) için evrimleşebilen mimari	Software Defined Wireless Network - SDWN	Costanzo vd. (2015)
Kablosuz ağlar üzerine yazılım tanımlı ağ (SDN) paradigmaları	Survey	Jagadeesan ve Krishnamachari (2015)
Bulut bilişim (Cloud Computing) için yazılım tanımlı ağlar (SDN) ve ağ sanallaştırması	SDN for Cloud Computing	Jain ve Paul (2013)
Yazılım tanımlı ağlarda denetleyici yerleşimi	On the placement of controllers in SDN	Hu vd. (2012)
Network Hypervisors: SDN mimarisini arttırma	Network Hypervisors	Huang vd. (2014)
Merkezileştirilmiş sunucu ve dağıtılmış aracılar içeren SDN yaklaşımı tabanlı IaaS (OpenStack infrastructure as a service) bulutunda yeni bir programlanabilir sanal ağ örneklemesi	Programmable virtual network	Wu vd. (2013)
SDN-Openflow ağlarında trafik mühendisliği için bir yol haritası	-	Akyildiz vd. (2014)
Yazılım tanımlı ağlarda akış başlatma sorununu gidermek için hibrit bir mekanizma	Flow Initiation	Soltani (2014)
Hiyerarşik yazılım tanımlı ağ kontrolörlerinde işbirlikçi yük dengeleme isimli bir yük dengeleme yöntemi	Controller Load Balancing Schemes	Selvi (2015)
Kablosuz ağlar için yazılıma dayalı ağ (SDN) yaklaşımı	CMaaS (Servis Olarak Bağlantı Yönetimi)	Yazıcı (2015)
SDN ile Qos yönetimi ve önceliklendirme	QoS Control and Prioritization with SDN	Gorkemli vd. (2015)
Yazılım tanımlı ağ (SDN) kullanarak ağların güvenliği	SDN to security (survey)	Ali vd. (2015)
YTA için NOC temelli Banyan OpenFlow anahtarı tasarımı	NOC based Banyan OpenFlow Switch	Ozcevik vd. (2015)
YTA'da denetleyici yerleşimi		Guner vd. (2015)
SDN-NFV-Bulut Bilişim teknolojilerinin servis sıralaması kapsamında incelenmesi		Akyildiz ve Saygun (2015)

sadece üretildikleri firma tarafından oluşturulan yazılımlarla yönetilebilmekte ve bu yönetim sınırlı olmaktadır. Bu sınırı ortadan kaldırabilmek amacıyla Nick McKeown ve arkadaşları tarafından geliştirilen OpenFlow standardı, bir yazılım aracılığıyla veri akışının tanımlanabilmesine izin veren bir ağ yaklaşımı oluşturmuştur (K. Greene 2009). Bu paradigma ile birlikte ağ trafiğini yönetmek, veri akış tablolarına ulaşmak, anahtar ve yönlendirici gibi ağ cihazlarına görevler atamak gibi birçok iş ve işlemleri bir yazılım aracılığıyla (OpenFlow) yapılmasını sağlamıştır. Bununla birlikte farklı üretici firma tarafından üretilen ağ cihazlarının yönetilebilir olması sağlanmıştır. Şekil 1'de YTA mimarisinin basitleştirilmiş görünümü verilmiştir.

YTA yaklaşımı ağ mimarisinde veri düzleminin programlanabilen kontrol düzlemi tarafından yönetilmesini sağlayan ve bu iki düzlemi birbirinden ayıran güncel bir teknolojidir. Günümüz ağ yapısında birleştirilmiş durumda olan kontrol platformu ve yönlendirme işlevleri, YTA mimarisinde birbirinden ayrılmış ve kontrol platformu YTA denetleyicisi adı verilen, sunucu üzerinde çalışan bir uygulamaya devredilmiştir. Merkezi konumda olan denetleyici, tüm ağ yapısı ile ilgili merkezi verilere sahip olduğu için veri düzlemindeki ağ cihazlarını yönetebilmektedir (Baktır vd., Schaller ve Hood 2017, Tomovic vd. 2017, Liao vd. 2017).

Bu mimaride denetleyici ve anahtarlayıcılar arasında iletişim sağlayabilmek için protokole ihtiyaç duyulmaktadır. Günümüzde en yaygın kullanılan kontrol protokolü OpenFlow protokolüdür (Farhady vd. 2015). Southbound arayüzü ve OpenFlow protokolü kullanılarak veri düzleminde var olan ağ cihazları akış bazında yönetilebilmektedir. Denetleyici yönlendirme kurallarını OpenFlow protokolü ile southbound ara yüzünden ağ cihazlarına yüklemekte ve böylece veri katmanında bulunan cihazlar sadece yönlendirme işlemi yapmaktadırlar. Denetleyicinin üst katmanı olan yönetim katmanıyla olan iletişimini de northbound ara yüzü ile gerçekleştirmektedir. Burada farklı yapıdaki uygulamalar aracılığıyla dinamik olarak ağ üzerinde çeşitli politikalar, kurallar tanımlanabilmektedir (Jain ve Paul 2013).



Şekil 1. YTA mimarisinin basitleştirilmiş görünümü.

YTA yaklaşımı, günümüz ağ mimarisinin dinamik ihtiyaçlara cevap verebilecek yeteneğe sahip olmasını sağlamaktadır. Her geçen gün artan cihazlar ve uygulamalar sonucunda ağ ve hizmet kalitesi gereksinimleri de dinamik olarak artmaya ve değişmeye devam etmektedir. Bu bağlamda YTA yaklaşımı akış bazlı yapılandırmaya uygun olan OpenFlow protokolü aracılığıyla bu dinamizm içinde ağ ve hizmet kalitesi gereksinimlerinin karşılanabilmesini sağlamaktadır (Baktır vd., Lin vd. 2017).

YTA mimarisinde merkezde yer alan kontrol düzlemi sayesinde denetleyici ağ üzerinde var olan tüm anahtarlayıcıların akış tablolarını tek bir merkezden yönetebilmektedir (Hu vd. 2014). Merkezde yer alan denetleyici ağ cihazlarından istatistiki tüm bilgileri toplayabilir ve böylece ağ hakkında merkezi tüm bilgilere sahip olabilmektedir. Bununla birlikte Northbound ara yüzünde çalışan uygulamalar üzerinden anahtarlayıcılara OpenFlow protokolü yardımıyla komutlar da gönderebilmektedir. Sonuç olarak kontrol düzlemini yönlendirme düzleminden ayıran bu YTA yaklaşımı teknolojik gelişmeye sınırlı olan günümüz ağ mimarisine esneklik ve teknolojik gelişim imkânları sunmaktadır.

YTA yaklaşımının yük dengeleme, güvenlik vb. uygulamaların koordinasyon kolaylığı, donanımdan bağımsız gelişim imkanı, ağ davranışlarının anlaşılması ve öngörülmesinin çok daha kolay hale gelmesi, gelişmiş yapılandırma, gelişmiş performans ve ağ mimarisinde yeniliklere açık olması gibi birçok faydaları mevcuttur. Xia vd. (2015) YTA'nın benimsediği kontrol, sadece bir geçişte paket yönlendirme değil, aynı zamanda veri bağı seviyesinde bağlantı ayarlayarak, katman yapısının engellerini kıran bir yapı içermektedir. Dahası ağın anlık durumunu elde edebilme yeteneği ile YTA hem anlık ağ durumuna hem de kullanıcı tanımlı politikalara göre bir ağın gerçek zamanlı merkezi kontrolüne izin vermesi de önemli faydalar arasındadır. YTA'nın veri merkezleri, omurga ağlar, internet değişim noktaları (IXP), bulut bilişim, bilişsel radyo uygulamaları, mobil, kurum ve ev ağları vb. birçok uygulama alanları vardır.

2.3. Yazılım Tanımlı Ağ Mimarisi

Xia vd. (2015) YTA mimarisinin üç katmandan oluştuğunu ve bu katmanların; veri katmanı, kontrol katmanı ve yönetim katmanı olduğunu ifade etmişlerdir. Kreutz vd. (2015) ise katman yapısının fonksiyonlarını daha ayrıntılı ele alıp sekiz temel katman yapısı oluşturmuşlardır. Bunlar; ağ altyapısı, güney ara yüzü (southbound interface), ağ hypervisor, ağ işletim sistemi, kuzey ara yüzü (northbound interface), dil tabanlı sanallaştırma, programlama dilleri ve ağ uygulamalarıdır. Farklı katmanlardan oluşan bu mimaride her katmanının kendine has fonksiyonel görevleri mevcuttur. Bu katmanlardan güney ara yüzü (southbound API), ağ işletim sistemi (NOSs), kuzey ara yüzü (northbound API) ve ağ uygulamaları YTA mimarisine özgü katmanlardır (Kreutz vd. 2015). Şekil 2'de yazılım tanımlı ağların düzlem, katman yapısı ve sistem tasarım mimarisi görülmektedir.

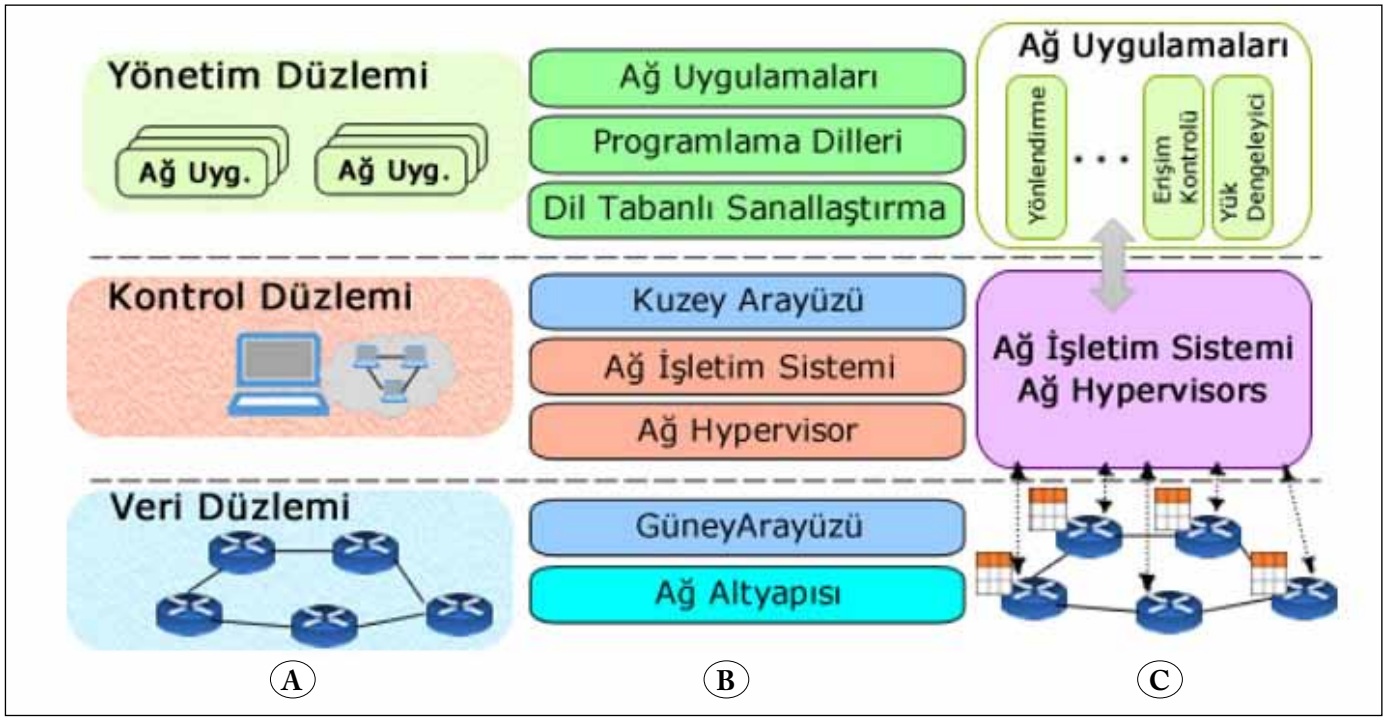
2.3.1. Katman: Ağ Altyapısı

Anahtar, yönlendirici ve middlebox gibi ağ cihazlarından oluşan YTA referans modelinin en alt katmanı olan ağ altyapısı, günümüz ağ altyapısı ile benzerlik göstermektedir. Temel farklılık ise; geleneksel ağ cihazları içine gömülen kontrol veya yazılım aracılığıyla otonom kararlar alabilirken, YTA yaklaşımında bu cihazlar sadece basit yönlendirme işlemleri yapmaktadırlar. Akıllı ağ, veri katmanı cihazlarından alınıp, mantıksal merkezileştirilmiş kontrol sistemlerine (ağ işletim sistemi ve uygulamalar) devredilmiştir. Farklı veri ve kontrol katman cihazları arasında uyumlu çalışabilmeye, düzgün bir şekilde iletişim kurabilmeye ve konfigürasyon yapabilme gibi önemli görevlerin standart bir ara yüz ile gerçekleştirilebilmesi bu yaklaşımın en önemli özelliğidir (Kreutz vd. 2015, Xia vd. 2015).

2.3.2. Katman: Southbound Ara Yüzü

Yönlendirme cihazları (SDN switches) ile kontrol katmanı elemanları arasında köprü görevi gören ve iletişim protokolü olarak tanımlanan Southbound ara yüzü (Kim ve Feamster 2013), fonksiyonel olarak kontrol ve veri katmanının tamamen ayrılmasını sağlayan katmandır (Kreutz vd. 2015, Hakiri vd. 2014). Southbound ara yüzü Şekil 2.a'da gösterildiği gibi veri düzlemi ile kontrol düzlemi arasında yer alan bir protokol önermektedir. YTA'nın katman mimarisi olan Şekil 2.b'de ağ altyapısı ve ağ hypervisor arasında yer alan katmandır.

OpenFlow ve Network Configuration Protocol (NetConf), YTA için Southbound ara yüz standardının en yaygın olanlarıdır. OpenFlow veri ve kontrol cihazları için bir iletişim kanalı sağlamakla birlikte, yönlendirme cihazları için ortak özellikler sağlamaktadır. OpenFlow ile birlikte ForCES (Doria vd. 2010), Open vSwitch Database (OVSDB) (Pfaff ve Davie 2013), POF (Song 2013), OpFlex (Smith vd. 2014), OpenState (Bianchi vd. 2014)



Şekil 2. Yazılım tanımlı ağlar **A)** düzlem yapısı, **B)** katman yapısı ve **C)** sistem tasarım mimarisi.

uygulamaları YTA için Southbound ara yüz uygulamaları arasında yer almaktadır.

2.3.3. Katman: Ağ Hypervisors

Günümüz modern bilgisayarlarda sanallaştırma kavramı etkin bir şekilde kullanılmaktadır. Özellikle son zamanlarda teknolojideki hızlı gelişmeler ile sanallaştırma kavramı bilgi işlem platformlarında ana konu haline gelmiştir. Sanal sunucu sayılarının fiziksel sunucu sayılarını aşması bunun bir göstergesi olarak görülmektedir (Koponen vd. 2014).

Sanallaştırma farklı sanallaştırma cihazları aracılığıyla sağlanmaktadır. Bu aracı cihazlar, yazılım veya donanım tabanlı olabilmektedir. Genel olarak sanal makineler ile donanımlar ve işletim sistemleri arasındaki iletişimi sağlayan ara katman "hypervisor" olarak adlandırılmaktadır. Hypervisors, farklı sanal makinelerin benzer donanım kaynaklarını paylaşmasına izin vermektedir (Blenk vd. 2016). Bu mimari de yer alan ağ hypervisors katmanı ise ağ donanımı için soyutlama katmanı sağlamakla birlikte, fiziksel ağdan bağımsız tamamen sanal ağlar oluşturulmasına izin vermektedir. Bu sayede etkin sanal ağ segmentlerinin dinamik ve bağımsız yönetilebilirliği sağlanmaktadır. Ağ hypervisors katmanı programlanabilen ve sanal optik ağ oluşmasını sağlamaktadır. Ağı oluşturan fiziksel elemanlar (anahtar, yönlendirici, güvenlik duvarı) sunucu sanallaştırma tarafından oluşturulan trafik taleplerini karşılamak için sanallaştırılabilmektedir. Ağ bileşenleri bir kez sanallaştırıldığında, ağ mühendislerinin programlı konfigürasyon kontrolünü yapabilmesi ve bu kaynakların yönetebilmesi gerekmektedir. Her sanal sunucu içinde bazı beceriler hypervisors tarafından sağlanmakta iken, daha sağlam programlama kaynakları tek bir havuz olarak ağdaki tüm sunucu ve servislerin kullanılır duruma gelmesi için gereklidir. İşte bu aşamada ağ hypervisors katmanı devreye girmektedir.

2.3.4. Katman: Ağ İşletim Sistemleri (Denetleyici)

YTA mimarisinde ağ yöneticisi tarafından belirlenen ilkelere dayalı ağ yapılandırmasını oluşturmak için kontrol mantığının en önemli öğesi olan ağ işletim sistemi (NOS) önemli bir unsurdur (Kreutz vd. 2015). Bu katmandaki işletim sisteminin rolü bilgisayar işletim sistemine çok benzemekle birlikte, ağ işletim sistemlerinin sağlamlığı ve güvenirliği ağ için oldukça önem arz etmektedir. Bu nedenle ağ kontrol yazılım tasarımı en aktif araştırma konularından biridir (Gong vd. 2015). Literatürde NOX (Gude vd. 2008), ONIX (Koponen vd. 2010), Maestro (Cai 2012), NOSIX (Yu vd. 2014) gibi birkaç farklı ağ işletim sistemi tasarımları bulunmaktadır. YTA kontrol yazılımının araştırmaları; a) denetleyici uygulama tasarımı ve performans ayarı, b) denetleyici ölçeklenebilirliği, c) ağ durum tutarlılığı, d) denetleyicinin servis kalitesi (QoS) gibi konulara odaklanmışlardır. McNettle (Voellmy vd. 2012) ve Beacon (Erickson 2013) gibi bazı araştırmaların tasarımları yaygın olarak kabul görmüştür. Araştırmalar ağ kaynaklarının yönetimi ve tüm yapılandırmaları kolaylaştırmak için cihazdan bağımsız ve ölçeklenebilen kontrol katmanı üzerinde birleşmektedirler.

Ağ işletim sistemleri sunucu üzerinde çalışan ve ağ üzerinde işletilen birçok ağ işlevlerinin yönetilmesini sağlayan bir yazılımdır. Ancak ağlar cihaza özel komut setleri, çoğunluklu bir şirkete ait ağ işletim sistemleri ile düşük seviyede yapılandırılmakta ve yönetilmektedir. Bu birtakım sınırlılıklar ve sorunlar doğurmaktadır. YTA ağ yönetimini kolaylaştırmak ve bir ağ işletim sistemi tarafından sunulan mantıksal merkezi kontrol aracılığıyla ağın sorunlarını çözmeyi ve yükünü hafifletmeyi temin etmektedir (Gude vd. 2008). Ağ işletim sisteminin en önemli faydası soyutlama, temel servisler ve genel uygulama geliştiricileri sağlamasıdır. Ağ durumu ve topoloji bilgileri, cihaz tanıtma ve ağ yapılandırma dağılımları gibi genel işlevler ağ işletim sisteminin bir servis hizmeti olarak sağlanabilmektedir. Örneğin ağ yöneticisi ağ politikalarını belirlerken artık yönlendirme elemanları arasında veri dağılımının düşük seviye detayları hakkında ilgilenme ihtiyacı duymayacaktır. Bu tür sistemler ağ uygulamaları ve yeni ağ protokolleri oluşturma karmaşıklığını azaltarak daha hızlı bir biçimde yenilik yapabilen bir ortam oluşturabilmektedir (Kreutz vd. 2015).

2.3.5. Katman: Northbound Arayüzü

Kontrol düzlemi ile yönetim düzlemi arasındaki veri alış verişini sağlayan uygulama odaklı ağ katmanı Northbound ara yüzüdür (Kreutz vd. 2015, Hakiri vd. 2014). Yönetim düzlemindeki uygulamalar aracılığıyla denetleyiciye komutlar göndermek için bu ara yüz aracılık görevi görmektedir. Northbound ara yüzü Southbound uygulamaları gibi bir donanım değil çoğunlukla yazılımsal bir ekosistemdir.

Northbound ve Southbound ara yüzleri YTA mimarisinin iki önemli soyutlama katmanıdır. Southbound ara yüzü için OpenFlow protokolü yaygın bir biçimde kabul görülmekte, ancak Northbound için böyle standart bir protokol henüz bulunmamaktadır (Hakiri vd. 2014). Northbound ara yüzü için bir standart tanımlamak için çok erken olmakla

birlikte, hala geliştirme çalışmaları devam etmektedir. YTA mimarisinin gelişimine bağlı olarak ortak bir Northbound ara yüzü beklenmektedir. Spesifik uygulamalara bağlı olmayan ağ uygulamalarına izin veren soyutlama, YTA yaklaşımının tüm potansiyelinin keşfedilmesi için önemlidir (Kreutz vd. 2015).

Northbound ara yüzü operasyonel görevlerin ve ağ politikaların nasıl ifade edileceğini ve aynı zamanda denetleyicinin anlayabileceği bir biçime nasıl dönüştürüleceğini tanımlamaktadır. Denetleyicinin üstünde yer alan bu ara yüz ağ politikalarının belirlendiği katmandır. Açık ve standart Northbound ara yüzleri farklı kontrol platformları arasında uygulamaların taşınabilirliği ve birlikte çalışabilirliği için önemlidir (Kim ve Feamster 2013). Northbound arayüzü Şekil 2.a'da gösterildiği gibi kontrol düzlemi ile yönetim düzlemi arasında yer almaktadır. YTA'nın katman mimarisi olan Şekil 2.b'de ise ağ işletim sistemi ile dil tabanlı sanallaştırma arasında yer alan katmandır. Floodlight, Trema, NOX, Onix, and OpenDaylight gibi denetleyiciler kendilerine özel Northbound uygulamaları önermektedirler. Ancak bunların her biri kendilerine özgü tanımlamalar içermektedir (Kreutz vd. 2015).

2.3.6. Katman: Dil Tabanlı Sanallaştırma

YTA yaklaşımının da dahil olduğu sanallaştırma tekniklerinin çoğu türlerine göre programlanabilen ağ, kontrol ve yönetim fonksiyonlarına, belli başlı teknoloji kaynaklarına ve ağ servisleri için yüksek seviyeli soyutlamaya izin vermektedir (Mambretti vd. 2014). Sanallaştırma çözümlerinin iki önemli karakteristiği koruma gibi arzu edilen özellikleri garanti ederken, modülerlik yeteneği ile farlı düzeylerde soyutlamalara izin vermesidir. Sanallaştırma teknikleri tek bir fiziksel altyapının farklı görünümlerini izin verebilmektedir. Örnek olarak bir sanal "büyük switch" birkaç yönlendirme cihazlarını bir arada temsil edebilecektir. Bu temelde yönlendirme kurallarını yüklemek amacıyla bilmek durumunda kaldıkları switch sırası hakkında düşünmek zorunda kalmadıkları için uygulama geliştiricilerin işini kolaylaştırmaktadır. Daha ziyade geliştiricilerin ağı basit olarak bir "büyük switch" olarak görmesini sağlamaktadır. Bu tür bir soyutlama gelişmiş güvenlik ile ilgili hizmetler gibi, karmaşık ağ uygulamalarının dağılımını ve gelişimi önemli ölçüde kolaylaştırır (Kreutz vd. 2015). Pyretic ağ topolojilerinde yüksek seviyeli soyutlama sağlayan ilginç bir programa dili örneğidir (Reich vd. 2013).

Dil tabanlı sanallaştırmanın bir diğer formu statik dilimlemedir. Burada ağ, uygulama katmanında tanımlı bir derleyici tarafından dilimlenmektedir. Statik dilimleme yaklaşımının bir örneği de Splendid izolasyonudur (Gutz vd. 2012).

2.3.7. Katman: Programlama Dilleri

Programlama dili, programcının bir algoritmayı ifade etmek için bilgisayara yapılması gereken işleri anlatan standartlaştırılmış kurallar bütünüdür. Günümüze kadar birçok farklı programlama dilleri geliştirilmiş ve geliştirilmeye de devam etmektedir. Programlama dilleri Assembly gibi düşük seviyeli makine dilinden, Java, C++ gibi yüksek seviyeli programlama dillerine evrimleşmiştir. Benzer biçimde ağların programlanabilmesi de düşük seviyeli makine dillerinden yüksek seviyeli programlama dillerine doğru gelişim göstermektedir (Koponen vd. 2014, Hinrichs vd. 2009, Foster vd. 2011, Monsanto vd. 2012, Voellmy ve Wang 2012, Voellmy ve Hudak 2011, Monsanto vd. 2013). OpenFlow (McKeown vd. 2008) ve NOX (Song 2013) gibi Assembly benzeri makine dilleri, esasen yönlendirme cihazlarının davranışlarını taklit ederek, ağ operatörünü ağ sorunlarını çözmek yerine daha çok düşük seviyeli detaylarla uğraştırmaktadır. Düşük seviyeli makine dillerinin kullanımı yazılımın tekrar kullanımını, modüller sistem oluşturmayı ve kapsamlı kod oluşturmayı daha zor hale getirmektedir. Bunun yanı sıra gelişim süreçlerinde daha çok hata oluşmasına sebep olmaktadır (Monsanto vd. 2013, Nelson vd. 2014, Katta vd. 2012).

Yüksek seviyeli programlama dilleri tarafından sağlanan soyutlamalar düşük seviyeli komut setlerinin oluşturduğu birçok sorunları çözmekte yardımcı olabilmektedir (Hinrichs vd. 2009, Foster vd. 2011, Monsanto vd. 2012, Voellmy vd. 2012, Voellmy ve Hudak 2011, Monsanto vd. 2013). Kreutz vd. (2015) YTA yaklaşımında yüksek seviyeli programlama dillerinin;

Programlanabilir yönlendirme cihazlarının görevlerini basitleştirmek amacıyla yüksek seviyeli soyutlamalar oluşturmak, Ağ yazılımı programcıları için daha verimli ve problem odaklı ortamları sağlamak, gelişim ve yeniliğe hızlandırmak, Ağ kontrol düzleminde yazılımın yeniden kullanılabilirliğini ve modüler sistem oluşmasını sağlamak, Ağ sanallaştırma gelişimini desteklemek için kullanılabileceğini ve tasarlanabileceğini ifade etmişlerdir.

2.3.8. Katman: Ağ Uygulamaları

Bu katman ağın beyni olarak görülebilir. Ağ uygulamaları yönlendirme cihazlarının davranışlarını belirlemekte ve veri düzlemine kurulacak komutların tercüme edilmesini sağlayan kontrol mantığını uygulamaktadır. Basit bir örnek

olarak yönlendirme işlemini alabiliriz. Buradaki temel mantık A noktasından B noktasına gidecek olan paketlerin gideceği yolu tanımlamaktır. Bu amacı gerçekleştirmek için yönlendirme uygulaması topoloji girişine göre kullanılacak yola karar verip, A noktasından B noktasına seçilen yoldaki tüm yönlendirme cihazlarına yönlendirme kurallarını sırasıyla yüklemek için denetleyiciye talimat vermelidir. YTA herhangi bir geleneksel ağ ortamına uygulanabilmektedir. Bu bağlamda çeşitli ağ ortamlarına uygulanabilen bu mimari çok çeşitli ağ uygulamalarına sebep olmaktadır. Mevcut ağ uygulamaları yönlendirme, yük dengeleme ve güvenlik politikaları gibi geleneksel işlevleri gerçekleştirmekte, ancak aynı zamanda güç tüketiminin azaltılması gibi yeni yaklaşımları da keşfetmektedir. Veri düzlemi için güvenirlik işlevleri ve yük devretme, uçtan ucu QoS uygulamaları, ağ sanallaştırması, kablosuz ağlar için mobilite yönetim vb. birçok ağ uygulamaları vardır. Gerçek kullanım durum dağıtımları ile birleştirilmiş çeşitli ağ uygulamalarının YTA'nın yaygın olarak benimsenmesi için önemli güçlerden biri olması beklenmektedir (Kreutz vd. 20153).

3. Tartışma ve Sonuç

Geleneksel ağları yönetmek oldukça karmaşık ve zordur. Bunun en önemli sebeplerinden biri kontrol ve veri düzlemlerinin dikey entegrasyonu ve üretilen firmaya özel donanımların olmasıdır. Her bir farklı üreticiye ait ağ donanımlarının kendine özgü yapılandırma ve yönetim ara yüzleri, ürün güncellemeleri veya yükseltmeleri için uzun dönemler beklenmesi diğer sebepler arasında yer almaktadır. Tüm bu durumlar ağ altyapısı sahipleri için satıcı odaklı problemlere yol açmış, herhangi bir değişim ve yenilik için ciddi kısıtlamalara sebep olmuştur. YTA uzun süredir devam eden bu sorunlar için bir çözüm olmuştur. Açık güney ara yüzü (Southbound) üzerinden dinamik olarak programlanabilen yönlendirme cihazlarının geliştirilmesi, kontrol ve veri düzlemlerinin birbirinden ayrılması, "ağın beyni" olarak ifade edilen mantıksal merkezileştirme aracılığıyla ağ yapısına global bir görünüm kazandırılması YTA'nın bu sorunlara ürettiği çözümlerden bazılarıdır.

YTA ilgili yapılan çalışmaların özellikle son yıllarda arttığı görülmektedir. YTA geleneksel ağ yapısı için geniş çözüm yelpazesi sunmasına rağmen henüz gelişmesi ve geliştirilmesi gereken yeni bir yaklaşımdır. Bu bağlamda yakın gelecekte YTA ile bulut bilişim, mobil ağlar, 4.5G veya 5G üzerine çalışmalar, bilişsel radyo, kablosuz ağlar, e-sağlık, veri merkezleri için çözümler gibi araştırmalara şahit olmaya devam edeceğiz. YTA ağ dünyası için gelecek anlamına gelmekle birlikte, araştırılması gereken başlıca konular henüz standart bir protokolü olmayan Northbound katmanı için protokoller, YTA yaklaşımı için güvenlik fonksiyonları, ağ işletim sistemleri, merkezi denetleyici ve anahtarlayıcı tasarımları, ağ cihazlarının performansı ve ölçeklenebilirlik gelişimleri üzerine çeşitli çalışmalar yapılabilir. Literatürde yapılan araştırmaların bu konuda hala çok eksik olduğu görülmektedir. Bu çalışmada geleneksel ağlar ile YTA yaklaşımı karşılaştırılmıştır. Bu yeni yaklaşımın katman yapısı aşağıdan yukarıya doğru; 1) ağ altyapısı, 2) güney ara yüzü, 3) ağ hypervisor, 4) ağ işletim sistemi, 5) kuzey arayüzü, 6) dil tabanlı sanallaştırma, 7) programlama dilleri, 8) ağ uygulamaları olmak üzere sekiz katman olarak ele alınmış ve her katman yapısı detaylı bir şekilde açıklanmıştır. Sonuç olarak YTA ile ilgili yapılan çalışmalar incelenmiş ve YTA yaklaşımı, düzlem ve katman yapısı hakkında kapsamlı bir çalışma sunulmuştur.

4. Kaynaklar

Akyildiz, I. F., Lee, A., Wang, P., Luo, M., Chou, W. 2014. A roadmap for traffic engineering in SDN-OpenFlow networks. *Comput. Netw.*, 71: 1-30.

Akyildiz, H. A., Saygun, E. 2015. SDN-NFV-cloud introduction in the context of service chaining. *In Signal Processing and Communications Applications Conference (SIU),* 2015 23th (pp. 2605-2608). IEEE.

Ali, S. T., Sivaraman, V., Radford, A., Jha, S. 2015. A survey of securing networks using software defined networking. *Reliability, IEEE Transactions on*, 64(3), (pp. 1086-1097).

Baktır, AC., Özgövde, BA., Ersoy, C. Servis Merkezli Yazılım Tanımlı Ağ Yaklaşımları.

Benson, T., Akella, A., Maltz, DA. 2009. Unraveling the Complexity of Network Management. *NSDI* (pp. 335-348).

Bianchi, G., Bonola, M., Capone, A., Cascone, C. 2014. OpenState: programming platform-independent stateful openflow applications inside the switch. *Comput. Commun. Rev.*, 44(2): 44-51.

Blenk, A., Basta, A., Reisslein, M., Kellerer, W. 2016. Survey on network virtualization hypervisors for software defined networking. *IEEE Communications Surveys & Tutorials*, *18*(1): 655-685.

Cai, Z. 2012. Maestro: achieving scalability and coordination in centralizaed network control plane. *Doktora Tezi*, Rice University.

Chen, T., Matinmikko, M., Chen, X., Zhou, X., Ahokangas, P. 2015. Software defined mobile networks: concept, survey, and research directions. *Commun. Magazine, IEEE*, 53(11): 126-133.

Costanzo, S., Galluccio, L., Morabito, G., Palazzo, S. 2015. Software Defined Wireless Network (SDWN): An evolvable architecture for W-PANs. *In Res. and Techn. for Soc. and Industry Leveraging a better tomorrow (RTSI), 2015 IEEE 1st International Forum on* (pp. 23-28). IEEE.

Doria, A., Salim, J. H., Haas, R., Khosravi, H., Wang, W., Dong, L., Halpern, J. 2010. Forwarding and control element separation (ForCES) protocol specification. *Internet Requests for Comments, RFC Editor, RFC*, 5810.

Erickson, D. 2013. The beacon openflow controller. *In Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking* (pp. 13-18). ACM.

Farhady, H., Lee, H., Nakao, A. 2015. Software-defined networking: A survey. *Comput. Netw.*, 81: 79-95.

Feamster, N., Rexford, J., Zegura, E. 2014. The road to SDN: an intellectual history of programmable networks. *Comput. Commun. Rev.*, 44(2): 87-98.

Foster, N., Harrison, R., Freedman, M. J., Monsanto, C., Rexford, J., Story, A., Walker, D. 2011. Frenetic: A network programming language. *ACM SIGPLAN Notices* , 46(9): 279-291.

Galinac Grbac, T., Caba, C. M., Soler, J. 2015. Software Defined Networking demands on software technologies. *In Inf. and Commun. Techn., Electr. and Microelectr. (MIPRO), 2015 38th Int. Conv. on* (pp. 457-462). IEEE.

Gorkemli, B., Tatlicioglu, S., Komurcuoglu, M., Karaman, M. A., Karakaya, O., Ulas, A. 2015. QoS control and prioritization with SDN. *In Signal Processing and Communications Applications Conference (SIU),* 2015 23th(pp. 2613-2616). IEEE.

Gong, Y., Huang, W., Wang, W., Lei, Y. 2015. A survey on software defined networking and its applications. *Frontiers of Comput. Science*, 9(6), (pp. 827-845).

Greene, K. 10 Şubat 2016. "10 Breakthrough Technologies: TR10: Software-defined Networking MIT Technology Rev. 2009. http://www2.technologyreview.com/news/412194/tr10-software-defined-networking/

Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N., Shenker, S. 2008. NOX: towards an operating system for networks. *Comput. Commun. Rev.*, 38(3): 105-110.

Guner, S., Selvi, H., Gur, G., Alagoz, F. 2015. Controller placement in software-defined mobile networks. *In Signal Processing and Communications Applications Conference (SIU),* 2015 23th (pp. 2619-2622). IEEE.

Hakiri, A., Gokhale, A., Berthou, P., Schmidt, D. C., Gayraud, T. 2014. Software-defined networking: Challenges and research opportunities for future internet. Comput. Netw., 75: 453-471.

Hinrichs, T. L., Gude, N. S., Casado, M., Mitchell, J. C., Shenker, S. 2009. Practical declarative network management. In Proceedings of the 1st ACM workshop on Research on enterprise networking (pp. 1-10). ACM.

Hu, Y. N., Wang, W. D., Gong, X. Y., Que, X. R., Cheng, S. D. 2012. On the placement of controllers in software-defined networks. J. *China Univ. Posts Telecom.*, 19: 92-171.

Hu, F., Hao, Q., Bao, K. 2014. A survey on software-defined network and openFlow: from concept to implementation. *Commun. Surveys & Tutorials, IEEE*, 16(4): 2181-2206.

Huang, S., Griffioen, J., Calvert, K. L. 2014. Network hypervisors: enhancing SDN infrastructure. *Comput. Commun.*, 46: 87-96.

Jagadeesan, N. A., Krishnamachari, B. 2015. Software-defined networking paradigms in wireless networks: a survey. *CSUR*, 47(2): 27.

Jain, R., Paul, S. 2013. Network virtualization and software defined networking for cloud computing: a survey. *Commun. Magazine, IEEE*, 51(11): 24-31.

Kaur, K., Kumar, K., Singh, J., Ghumman, N. S. 2015. Programmable firewall using Software Defined Networking. *Computing for Sustainable Global Development (INDIACom), 2015 2nd International Conference on* (pp. 2125-2129). IEEE.

Katta, N. P., Rexford, J., Walker, D. 2012. Logic programming for software-defined networks. In Workshop on Cross-Model Design and Validation (XLDI) (Vol. 412).

Kim, H., Feamster, N. 2013. Improving network management with software defined networking. *Commun. Magazine, IEEE*, 51(2): 114-119.

Koponen, T., Amidon, K., Balland, P., Casado, M., Chanda, A., Fulton, B., Lambeth, A. 2014. Network virtualization in multi-tenant datacenters. In *11th USENIX Symposium on Networked Systems Design and Implementation* (NSDI 14) pp. 203-216.

Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., Shenker, S. 2010. Onix: A Distributed Control Platform for Large-scale Production Networks. *In OSDI,* 10: 1-6.

Kreutz, D., Ramos, F. M., Esteves Verissimo, P., Esteve Rothenberg, C., Azodolmolky, S., Uhlig, S. 2015. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1): 14-76.

Liao, J., Sun, H., Wang, J., Qi, Q., Li, K., Li, T. (2017). Density cluster based approach for controller placement problem in large-scale software defined networkings. *Comput. Networks*, 112: 24-35.

Lin, Y. D., Lai, Y. C., Teng, H. Y., Liao, C. C., Kao, Y. C. (2017). Scalable multicasting with multiple shared trees in software defined networking. *J. Network Comput. Appl.*, 78: 125-133.

Liu, S., Li, B. 2015. On scaling software-Defined Networking in wide-area networks. *Tsinghua Science and Techn.*, 20(3): 221-232.

Mambretti, J., Chen, J., Yeh, F. 2014. Software-Defined Network Exchanges (SDXs) and Infrastructure (SDI): Emerging innovations in SDN and SDI interdomain multi-layer services and capabilities. *In Science and Technology Conference (Modern Networking Technologies)(MoNeTeC), 2014 International* (pp. 1-6). IEEE.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Turner, J. 2008. OpenFlow: enabling innovation in campus networks. *Comput. Commun. Rev.*, 38(2): 69-74.

Monsanto, C., Foster, N., Harrison, R., Walker, D. 2012. A compiler and run-time system for network programming languages. In ACM SIGPLAN Not., Vol. 47(1): 217-230.

Monsanto, C., Reich, J., Foster, N., Rexford, J., Walker, D. 2013. Composing software defined networks. In Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13) pp. 1-13.

Nelson, T., Ferguson, A. D., Scheer, M. J., Krishnamurthi, S. 2014. Tierless programming and reasoning for software-defined networks. In 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14) pp. 519-531.

Nunes, B. A., Mendonca, M., Nguyen, X. N., Obraczka, K., Turletti, T. 2014. A survey of software-defined networking: Past, present, and future of programmable networks. *Commun. Surveys & Tutorials, IEEE*, 16(3): 1617-1634.

Ozcevik, Y., Erel, M., Canberk, B. 2015. NOC based Banyan OpenFlow switch for Software Defined Networks. *In Signal Processing and Communications Applications Conference (SIU),* 2015 23th pp. 1433-1436.

Pfaff, B., Davie, B. 2013. The Open vSwitch Database Management Protocol.

Reich, J., Monsanto, C., Foster, N., Rexford, J., Walker, D. 2013. Modular sdn programming with pyretic. Technical Reprot of USENIX.

Schaller, S., Hood, D. 2017. Software Defined Networking Architecture Standardization. *Comput. Stand. Interfaces*.

Selvi, H. 2015. Controller load balancing schemes in software defined networks / Yazılım tanımlı ağ kontrolörlerinde yük dengeleme. *Yüksek Lisans Tezi*, Boğaziçi Üniversitesi / Fen Bilimleri Enstitüsü, Ankara.

Silva, F., Castillo-Lema, J., Neto, A., Silva, F., Rosa, P. 2014. Software defined eHealth networking towards a truly mobile and reliable system. *In e-Health Networking, Applications and Services (Healthcom), 2014 IEEE 16th International Conference on* (pp. 560-564).

Smith, M., Dvorkin, M., Laribi, Y., Pandey, V., Garg, P., Weidenbacher, N. 2014. OpFlex control protocol. *IETF, Apr*.

Soltani, A. 2014. Flow initiation in software defined networking / Yazılım tanımlı ağlarda akış başlatma. *Yüksek Lisans Tezi*, Orta Doğu Teknik Üniversitesi / Enformatik Enstitüsü, Ankara.

Song, H. 2013. Protocol-oblivious forwarding: Unleash the power of SDN through a future-proof forwarding plane. In Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking (pp. 127-132). ACM.

Tomovic, S., Yoshigoe, K., Maljevic, I., Radusinovic, I. 2017. Software-Defined Fog Network Architecture for IoT. *Wireless Personal Commun.*, *92*(1): 181-196.

Voellmy, A., Hudak, P. 2011. Nettle: Taking the sting out of programming network routers. In Practical Aspects of Declarative Languages (pp. 235-249). Springer Berlin Heidelberg.

Voellmy, A., Wang, J. 2012. Scalable software defined network controllers. *In Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for comput. Commun.* 289-290.

Voellmy, A., Kim, H., Feamster, N. 2012. Procera: a language for high-level reactive network control. In Proceedings of the first workshop on Hot topics in software defined networks. ACM., pp. 43-48.

Xia, W., Wen, Y., Foh, C., H., Niyato, D. Xie, H. 2015. A Survey on Software-Defined Networking. *IEEE Commun. Surveys Tutorials, 17,1, first quarter.*

Wu, YJ., Liang, JX., Zhang, H., Lin, ZW., Yan, M. A., Zhong, TIAN. 2013. Programmable virtual network instantiation in IaaS cloud based on SDN. *J. China Univ. Posts Telecommun.*, 20: 121-125.

Yazıcı, V. 2015. A software-defined networking approach for wireless systems / Kablosuz ağlar için bir yazılıma dayalı ağ yaklaşımı. *Doktora Tezi*, Özyeğin Üniversitesi / Fen Bilimleri Enstitüsü, İstanbul.

Yu, M., Wundsam, A., Raju, M. 2014. NOSIX: A lightweight portability layer for the SDN OS. *ACM SIGCOMM Comput. Commun. Rev.*, 44(2): 28-35.