BİLGİSAYAR AĞLARI PROJE ROPORU

Yazılım Tanımlı Ağ (Software-Defined Network) Programlama

19010310021 Gaye Nur YANKIN

18670310076 Yaman ASHMAR

19010310048 Ceyda ÖZTÜRK

20010310061 Meryem SOY

18010311010 Kerem AY

18010310018 Doğan ARSLAN

GRUP_13

GİRİŞ:

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 birtakım sorunları çözerken, bununla birlikte daha karmaşık ve yönetilmesi daha zor ağ yapılarının oluşmasına sebep olmaktadır (Benson vd. 2009).

Yazılım Tanımlı Ağ (SDN) Nedir?

Yazılım tanımlı ağ (SDN), ağ kontrol düzleminin sevk düzleminden fiziksel olarak ayrılması ve bir kontrol düzleminin çeşitli aygıtları denetlemesini sağlar. Yazılım tanımlı ağ (SDN), günümüz uygulamalarının yüksek bant genişliği ve dinamik yapısı için ideal hale getiren, dinamik, yönetilebilir, uygun maliyetli ve uyarlanabilir yeni bir mimaridir. Bu mimari, ağ denetiminin doğrudan programlanabilmesini ve altyapının, uygulamalar ve ağ hizmetleri için soyutlanmasını sağlayan ağ kontrol ve yönlendirme işlevlerini ayrıştırır. OpenFlow® protokolü, SDN çözümleri oluşturmak için temel bir unsurdur.

Yazılım Tanımlı Ağ (SDN) Mimarisi

Doğrudan programlanabilir: Ağ kontrolü direkt olarak programlanabilir çünkü yönlendirme işlevlerinden ayrılmıştır.

Çevik: Yönlendirme kontrolü sunarak, yöneticilerin değişen ihtiyaçları karşılamak için ağ genelinde trafik akışını dinamik olarak ayarlamalarını sağlar.

Merkezi olarak yönetilen: Ağ zekası, ağlara genel bakışını koruyan yazılım tabanlı SDN denetleyicilerinde (mantıksal olarak) merkezileştirilir. Bu, uygulamaları ve ilke motorlarını tek bir mantıksal anahtar olarak görür.

Programlı olarak yapılandırılmış: SDN, ağ yöneticilerinin ağ kaynaklarını dinamik, otomatik SDN programları yoluyla yapılandırmalarını, yönetmelerini, güvenliğini ve optimize etmelerini sağlar; bu programlar, tescilli yazılımlara bağımlı olmadığı için kendileri yazabilirler.

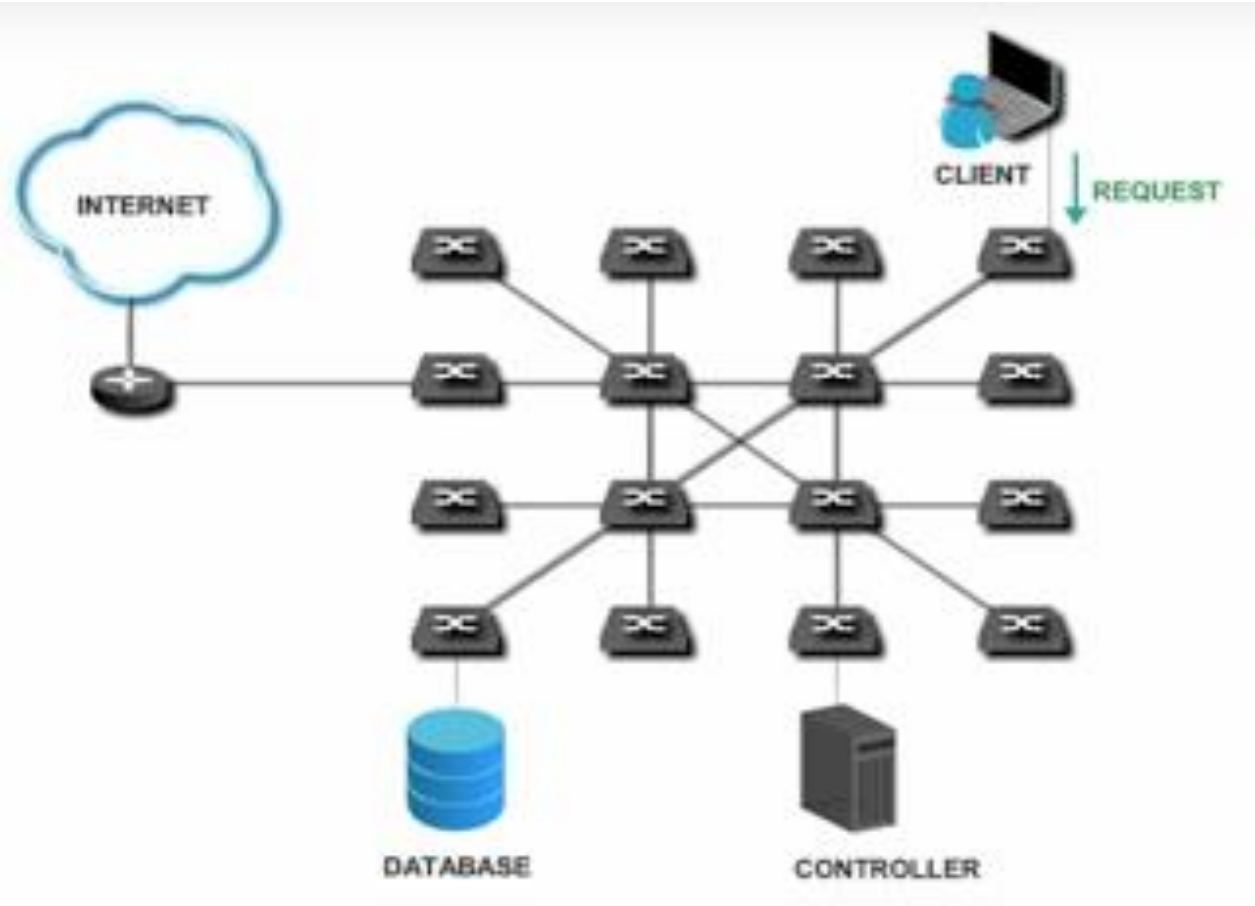
Açık standartlar- Tedarikçi Tabanlı- Tarafsız: SDN, açık standartlar aracılığıyla uygulandığında, ağ tasarımını ve operasyonunu basitleştirir, çünkü talimatlar birden çok satıcıya özgü cihaz ve protokol yerine SDN denetleyicileri tarafından sağlanmaktadır.

Yazılım Tanımlı Ağ Mimarisi:

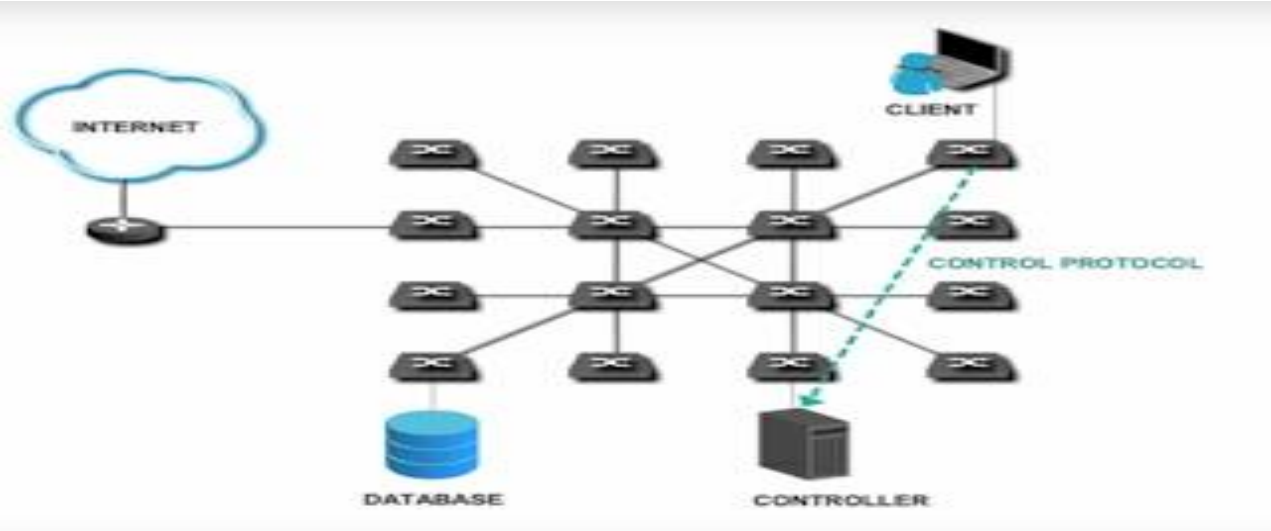
(2015) YTA mimarisinin üç katmandan oluştuğunu ve bu katmanların; veri katmanı, kontrol katmanı ve yönetim katmanı olduğunu ifade etmişlerdir. Kreutz vd. (2015) ise katman yapısının fonksiyonlarını daha ayrıntılı ele alıp sekiz temel katman yapısı oluşturmuşlardır. Bunlar; ağ altyapısı, güney ara yüzü (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.

SDN Nasıl Çalışır ?

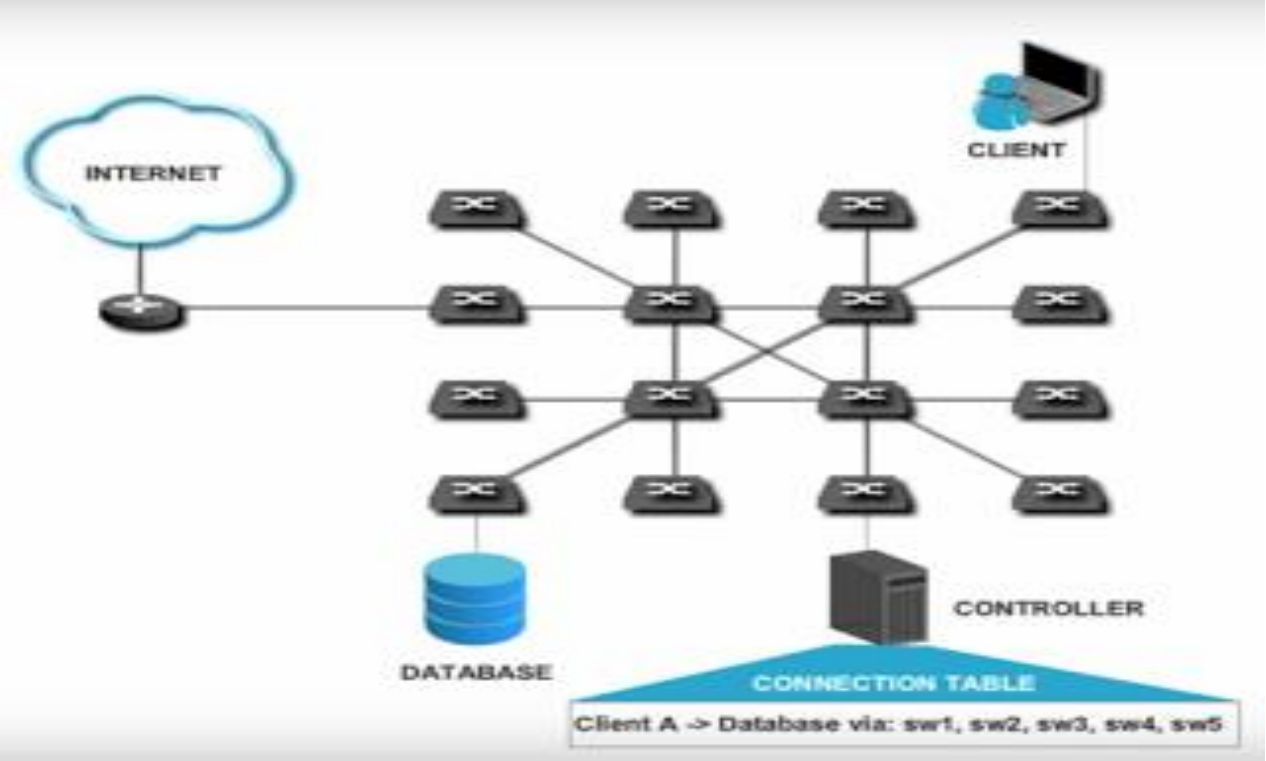
Ağ yapılarında haberleşmek yada bir bilgi almak için istemci kendisine en yakın cihaz aracılığıyla istekte bulunur.



Aynı istek olup olmadığını kontrol ederek istek alan cihaz direk olarak merkezi kontrolcü ile haberleşir, gelen trafik isteği karşısında nasıl davranmaları gerektiğini kontrolcüye sorarlar.

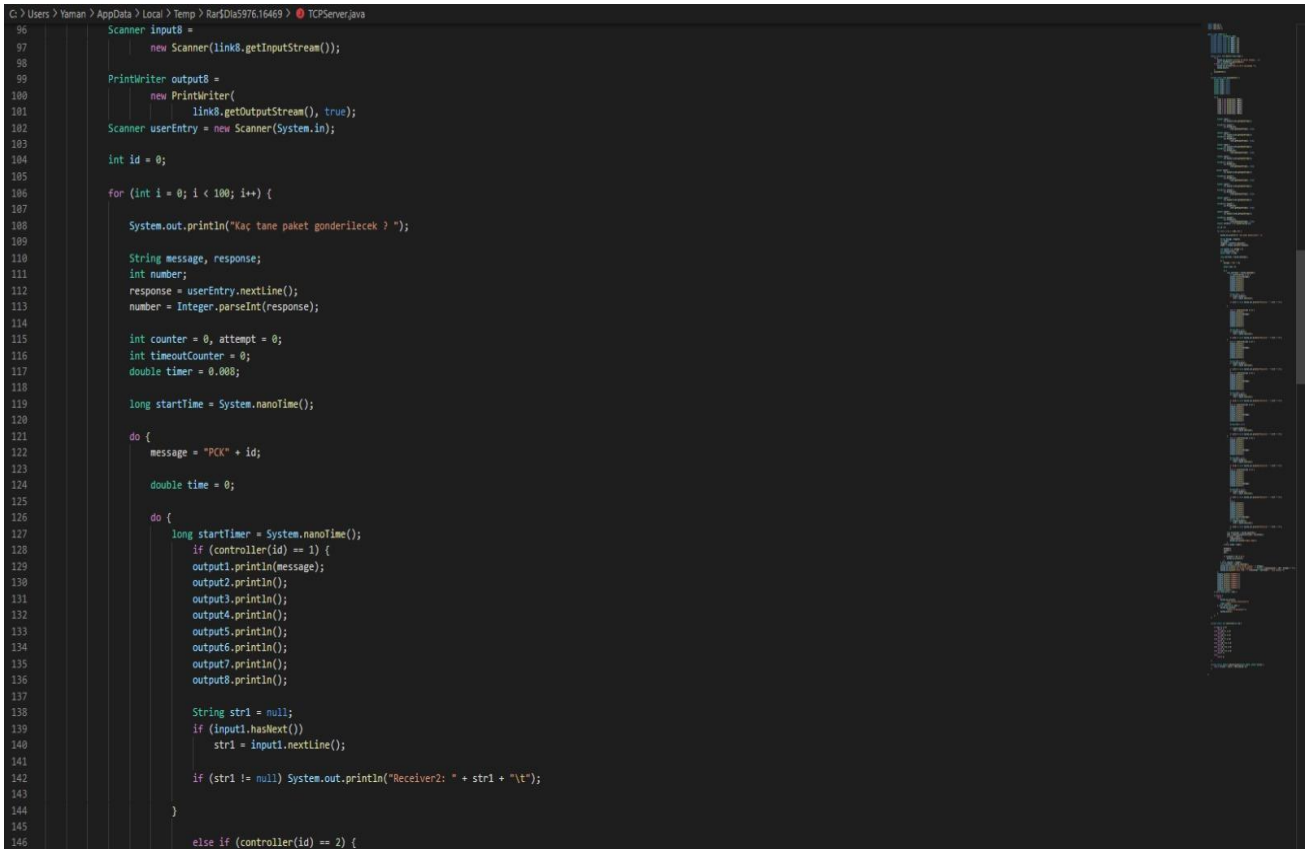


Kontolcü tarafından forwarding table'da bir kural olup olmadığı kontrol edilir. Eğer mevcut bir akış kuralı var ise o davranış gönderilir. Bir kural yok ise tanımlanan kurallar çerçevesinde akış tablosuna yeni bir kural eklenerek yapılacak davranış kendisine sorgu yapan cihaza cevap olarak gönderilir.



Proje Kodları:

```
private static String handleClient()
{
   Socket ServerLink = null;

   try
   {
      ServerLink = ServerSocket.accept();

      Scanner ServerInput =
         new Scanner(ServerLink.getInputStream());

      PrintWriter senderOutput = new PrintWriter(ServerLink.getOutputStream(), true);

        String message = ServerInput.nextLine();


        Scanner receiverInput = new Scanner(ClientLink.getInputStream());

        PrintWriter ClientOutput = new PrintWriter(ClientLink.getOutputStream(), true);


      while (!message.equals("***KAPAT***")){

          if(!message.isEmpty())
              System.out.print("Sunucu mesaj "+message+"\t");

          ClientOutput.println(message);

          String str=receiverInput.nextLine();

          if(!str.isEmpty() && !message.isEmpty())
              System.out.println("Alıcıdan mesaj: "+str);

          senderOutput.println(str);

          message = ServerInput.nextLine();

     }
```




```
45      private static String handleClient()
46      {
47          Socket ServerLink = null;
48
49          try
50          {
51              ServerLink = ServerSocket.accept();
52
53              Scanner ServerInput =
54                  new Scanner(ServerLink.getInputStream());
55
56              PrintWriter senderOutput = new PrintWriter(ServerLink.getOutputStream(),  autoFlush: true);
57
58                String message = ServerInput.nextLine();
59
60
61                Scanner receiverInput = new Scanner(ClientLink.getInputStream());
62
63                PrintWriter ClientOutput = new PrintWriter(ClientLink.getOutputStream(),  autoFlush: true);
64
65
66              while (!message.equals("***KAPAT***")){
67
68                  if(!message.isEmpty())
69                      System.out.print("Sunucu mesajı: "+message+"\t");
70
71                  ClientOutput.println(message);
72
73                  String str=receiverInput.nextLine();
74
75                  if(!str.isEmpty() && !message.isEmpty())
```

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.

Kaynakça:

https://github.com/evrimguler/SocketProgrammingJAVA/find/master

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.