T.C.

BARTIN ÜNİVERSİTESİ

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

AKILLI SİSTEMLER MÜHENDİSLİĞİ ANABİLİM DALI

YÜKSEK LİSANS TEZİ

YAZILIM TANIMLI ÇOKLU AĞLARDA MAKİNE ÖĞRENMESİ VE BLOK ZİNCİRİ İLE GELİŞTİRİLMİŞ SERVİS KALİTESİ DESTEKLİ YÖNLENDİRME MİMARİSİ

ZEYNEP ÖNDER

DANIŞMAN

DR. ÖĞR. ÜYESİ EVRİM GÜLER

BARTIN-2023

T.C.

BARTIN ÜNİVERSİTESİ

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

AKILLI SİSTEMLER MÜHENDİSLİĞİ ANABİLİM DALI

YAZILIM TANIMLI ÇOKLU AĞLARDA MAKİNE ÖĞRENMESİ VE BLOK ZİNCİRİ İLE GELİŞTİRİLMİŞ SERVİS KALİTESİ DESTEKLİ YÖNLENDİRME MİMARİSİ

YÜKSEK LİSANS TEZİ

Zeynep ÖNDER

BARTIN-2023

BEYANNAME

Bartın Üniversitesi Lisansüstü Eğitim Enstitüsü tez yazım kılavuzuna göre Dr. Öğr. Üyesi Evrim GÜLER danışmanlığında hazırlamış olduğum "YAZILIM TANIMLI ÇOKLU AĞLARDA MAKİNE ÖĞRENMESİ VE BLOK ZİNCİRİ İLE GELİŞTİRİLMİŞ SERVİS KALİTESİ DESTEKLİ YÖNLENDİRME MİMARİSİ" başlıklı yüksek lisans tezimin bilimsel etik değerlere ve kurallara uygun, özgün bir çalışma olduğunu, aksinin tespit edilmesi halinde her türlü yasal yaptırımı kabul edeceğimi beyan ederim.

28.08.2023

Zeynep ÖNDER

ÖNSÖZ

Tez çalışmamı hazırlarken bilgi ve tecrübeleri ile bana sürekli destek olan danışman hocam Sayın Dr. Öğr. Üyesi Evrim GÜLER'e teşekkür eder, saygılarımı sunarım.

Yüksek lisans öğrenimim boyunca her zaman yanımda olan başta eşim Emre ÖNDER'e, çocuklarım Ahmet Uluğ ÖNDER ve Amine Ulum ÖNDER'e, Bartın Belediyesi Bilgi İşlem Müdürlüğü'nde birlikte çalıştığım arkadaşlarım başta olmak üzere mesai arkadaşlarıma teşekkür ederim.

Zeynep ÖNDER

ÖZET

Yüksek Lisans Tezi

YAZILIM TANIMLI ÇOKLU AĞLARDA MAKİNE ÖĞRENMESİ VE BLOK ZİNCİRİ İLE GELİŞTİRİLMİŞ SERVİS KALİTESİ DESTEKLİ YÖNLENDİRME MİMARİSİ

Zeynep ÖNDER

Bartın Üniversitesi
Lisansüstü Eğitim Enstitüsü
Akıllı Sistemler Mühendisliği Anabilim Dalı

Tez Danışmanı: Dr. Öğr. Üyesi Evrim GÜLER
Bartın-2023, sayfa: 40

Geleceğin ağlarının ana görevi, mümkün olduğunca entellektüelleştirme, etkinleştirme ve özelleştirme için akıllı ağ oluşturma mimarileri oluşturmaktır. Yazılım Tanımlı Ağlar (YTA) oluşturma teknolojisi, geleneksel ağ mimarisinde kontrol düzlemi ile veri düzlemi arasındaki sıkı bağlantıyı kırarak ağ kaynaklarının kontrol edilebilirliğini, güvenliğini ve ekonomisini gerçeğe dönüştürür. Yapay zekanın yöntemlerinden biri olan makine öğrenimi, YTA mimarisiyle birleştiğinde ağ kaynak yönetimi, uçtan uca yol planlama, trafik programlama, arıza teşhisi veya ağ güvenliği gibi alanlarda etkin olmaktadır. Günümüz ağ mimarisindeki cihaz sayılarının hızla artışı, güvenlik, gizlilik, hizmet sağlama ve ağ yönetimi gibi operasyonların merkezi bir kontrol sistemi ile yapılmasını zorlaştırmaktadır. Bu nedenle, merkezi olmayan ağ yönetiminin güvenli, akıllı ve verimli olabilmesi için blokzinciri ve makine öğreniminin birlikte kullanımı akademik çalışmalarda ve endüstriyel uygulamalarda ilgi görmektedir. YTA, geleneksel ağlara göre yazılım ile yönetildiğinden kontrolü daha kolaydır. Bu ağlarda, blokzincir işlem hacmini artırmak ve ortalama blok oluşturma süresini azaltmak gibi blokzincir teknolojisi ile ilgili literatürde farklı makine öğrenme teknikleri kullanılmıştır. Bu çalışmanın amacı, pekiştirmeli öğrenme modellerinin yardımıyla blokzinciri ile servis kalitesi destekli uçtan

uca çoklu ağlarda yol bulma mimarisinin oluşturulmasıdır.

Anahtar Kelimeler: blokzincir, yazılım tanımlı ağlar, çoklu alan ağları, pekiştirmeli öğrenme, makine öğrenmesi, servis kalitesi destekli yönlendirme

ABSTRACT

M. Sc. Thesis

MACHINE LEARNING ENHANCED QUALITY OF SERVICE BASED ROUTING WITH BLOCKCHAIN IN MULTI-DOMAIN SOFTWARE DEFINED NETWORKS

Zeynep ÖNDER

Bartın University
Graduate School
Department of Intelligent Systems Engineering

Thesis Advisor: Asst. Prof. Dr. Evrim GÜLER
Bartın-2023, pp: 40

The main task of future networks is to create intelligent networking architectures for as much intellectualization, activation and customization as possible. Software Defined Networks (SDN) creation technology breaks the tight connection between the control plane and the data plane in traditional network architecture, making the controllability, security and economy of network resources a reality. Machine learning, one of the methods of artificial intelligence, when combined with SDN architecture, is effective in areas such as network resource management, end-to-end route planning, traffic programming, fault diagnosis or network security. The rapid increase in the number of devices in today's network architecture makes it difficult to carry out operations such as security, privacy, service provision and network management with a central control system. Therefore, the combined use of blockchain and machine learning for decentralized network management to be secure, smart and efficient is of interest in academic studies and industrial applications. SDN is easier to control as it is managed by software compared to traditional networks. In these networks, different machine learning techniques have been used in the literature on blockchain technology, such as increasing the blockchain transaction volume and reducing the average block generation time. The aim of this study is to create a wayfinding architecture in end-to-end multi-networks supported by blockchain with the

help of Reinforcement Learning (RL) models.

Keywords: blockchain, software defined networks, multi-domain networks, quality of service based routing, Reinforcement Learning, machine learning

İÇİNDEKİLER

KABUL VE ONAY ..... Hata! Yer işareti tanımlanmamış.
BEYANNAME ..... **iii**
ÖNSÖZ **4**
ÖZET **5**
ABSTRACT **7**
İÇİNDEKİLER **9**
ŞEKİLLER DİZİNİ **11**
KISALTMALAR DİZİNİ **12**
1. GİRİŞ **1**
1.1 Yazılım Tanımlı Ağlar **1**
1.2 Hizmet Kalitesi **2**
1.3 Çoklu Alan Ağları **3**
1.4 Pekiştirmeli Öğrenme **4**
1.4.1 Pekiştirmeli Öğrenme'nin Bileşenleri: **4**
1.4.2 Pekiştirmeli Öğrenme'nin Temel Kavramları: **5**
1.5 Blokzincir **7**
2. LİTERATÜR ÖZETİ **10**
3. MATERYAL VE METOT **12**
3.1 MODÜLLER **12**
3.1.1 Hizmet Kalitesi Yönetim Modülü **12**
3.1.2 Pekiştirmeli Öğrenme Modülü **12**
3.1.3 Blokzincir Modülü **14**
3.1.4 Ağ Sanallaştırma Modülü **18**
3.1.5 Yönlendirme Optimizasyon Modülü **19**
3.2 UYGULAMALAR **20**
3.2.1 Hizmet Kalitesi Destekli Yönlendirme Uygulaması **20**
3.2.2 Ağ İzleme ve Analitik Uygulaması **21**
3.2.3 Kaynak Tahsisi ve Yönetim Uygulaması **22**
3.2.4 Güvenlik ve Kimlik Doğrulama Uygulaması **23**
3.2.5 Hizmet Düzeyi Anlaşması Yönetimi Uygulaması **24**
3.3 Pekiştirmeli Öğrenme Tabanlı Yönlendirme Mimarisi Çalışma Alt Yapısı **25**

4. BULGULAR **26**
5. TARTIŞMA **36**
6. SONUÇ VE ÖNERİLER **38**
KAYNAKLAR **39**
ÖZGEÇMİŞ ..... Hata! Yer işareti tanımlanmamış.

ŞEKİLLER DİZİNİ

Şekil No | **Sayfa No**

Şekil 1. Genel Yazılım Tanımlı Ağ Mimarisi 1
Şekil 2. Blokzincir Entegre Edilmiş Yazılım Tanımlı Çoklu Ağlar Yapısı 9
Şekil 3. Pekiştirmeli Öğrenme Mimarisi 13
Şekil 4. Hizmet İsteği veri yapısı 16
Şekil 5. Blokzincir özellikli hizmet kalitesi uyumlu ağlar arası yönlendirmedeki bir işlemin veri yapısı 17
Şekil 6. Yazılım tanımlı çoklu ağlarda makine öğrenmesi ve blokzincir ile geliştirilmiş servis kalitesi destekli yönlendirme mimarisinde ağ denetleyicisin genel yapısı 21
Şekil 7. Pekiştirmeli Öğrenme Tabanlı Yönlendirme Mimarisi Çalışma Alt Yapısı 25
Şekil 8. Blokzincir İşlem Modülünün Tanımlanması 26
Şekil 9. Blok Yapısı 27
Şekil 10. Blok zincir Modülü 28
Şekil 11. Blokzincir Yöneticisi 29
Şekil 12. Konsensüs Modülü 30
Şekil 13. Hizmet Kalitesi Destekli Servis Talebi Modülü 31
Şekil 14. Servis Talebi Oluşturma Modülü 32
Şekil 15. Pekiştirmeli Öğrenme Tabanlı Uçtan Uca En Kısa Yol Bulma Modülü 35

KISALTMALAR DİZİNİ

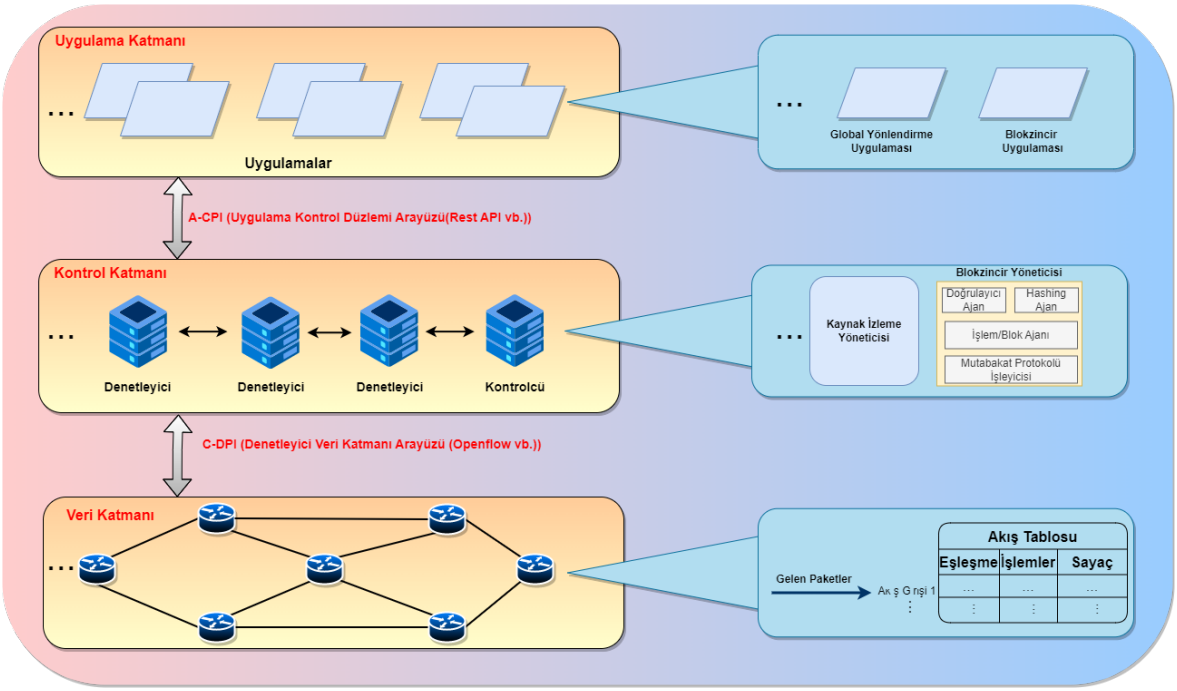
KISALTMALAR

A	: Eylem
A-CPI	: Uygulama Kontrol Düzlem Arayüzü
As NO	: ISS Kimlik Numarası
C-CPI	: Denetleyici Veri Katmanı Arayüzü
Egress	: Çıkış Düğümü
Ingress	: Giriş Düğümü
ISS	: Internet Servis Sağlayıcı
ID	: Kimlik Numarası
P	: Durum
PoS	:Hisse Kanıt Algoritması
PoW	:İş Kanıt Algoritması
R	: Ödül
RL	: Pekiştirmeli Öğrenme
SDMN	: Yazılım Tanımlı Çoklu Ağlar
SLA	: Hizmet Düzey Anlaşması
OSPF	:En Kısa Yol Öncelikli Yönlendirme Protokolü
QoS	: Ağ İletişimi Hizmet Kalitesi
YTA	:Yazılım Tanımlı Ağ

1. GİRİŞ

1.1 Yazılım Tanımlı Ağlar

Yazılım Tanımlı Ağ (YTA) teknolojisi son yıllarda ortaya çıkan yeni bir ağ mimarisidir. Openflow protokolü kullanılarak oluşturulan bu mimari, farklı uygulama ve servislere yönelik özel algoritmalar kullanmaya olanak sağlıyor. YTA, kontrol mekanizmalarını yönlendirme cihazlarından ayırmayı amaçlamaktadır. Yazılım tanımlı ağ oluşturma yaklaşımı, ağdaki trafik kontrol sistemini, trafik iletim sisteminden ayırarak bilgisayar ağlarını kolaylaştırmak için geliştirilen bir yaklaşımdır. YTA, geleneksel ağ cihazlarında barındırılan kontrol düzlemini veri düzleminden ayırmayı ve ağ servisleri için programlanabilirlik sağlayarak ağ servislerinin programlanabilir olmasına yardımcı olmaktadır. Şekil 1, Yazılım Tanımlı Ağ mimarisinin 3 katmanlı (Veri katmanı, Kontrol katmanı ve Uygulama katmanı) genel mimarisini göstermektedir.



Şekil 1. Genel Yazılım Tanımlı Ağ Mimarisi

Yazılım Tanımlı Ağ, şekilde gösterildiği gibi veri, kontrol ve uygulama düzlemlerinden oluşur. Alt düzlemin büyük bölümünü oluşturan veri düzlemi, sanal ve fiziksel anahtarlar ve yönlendiriciler, erişim noktaları vb. dahil olmak üzere çeşitli ağ bileşenlerini içerir. Denetleyici-Veri Düzlemi Arabirimleri (C-DPI) aracılığıyla, YTA denetleyicileri bu

aygıtlarla etkileşime girebilir ve bunları kontrol edebilir. OpenFlow iletişim protokolü (McKeown vd., 2008), veri düzlemi cihazları ve denetleyiciler arasındaki etkileşimi desteklemek için yaygın olarak kullanılan C-DPI standardıdır ve paket yönlendirme, çok önemli ve temel bir veri düzlemi işlevidir.

1.2 Hizmet Kalitesi

Hizmet Kalitesi (Quality of Service - QoS) provizyonu için geleneksel ağ yapısına göre en iyi seçenek Yazılım Tanımlı Ağ (YTA)'dır. YTA denetleyicisi, QoS seviyesini sağlamak üzere ağ cihazlarının otomatik ve esnek bir şekilde programlanmasını sağlar (Helebrandt ve Kotuliak, 2014).

Hizmet kalitesi, sınırlı ağ kapasitesinin yüksek öncelikli uygulamaların ve trafiğin, güvenilir performans kriterleri altında çalışmasını sağlayan bir ağ ön koşuludur. Video yayını, bulut bilgi işlem ve gerçek zamanlı iletişim sistemleri gibi sistemler, bant genişliğine bağımlı yoğun uygulamaların artan yaygınlığı ile güvenilir ve tutarlı ağ performansına olan ihtiyacın artmasına neden oldu. QoS, sistem tarafından sağlanan hizmet kalitesinin seviyesini belirlemede çok önemli bir yere sahiptir. QoS, sistemin bant genişliği, paket kaybı ve gecikme süresi gibi çeşitli parametrelerin değerlendirilmesinin sonucunda elde edilir (Tomovic vd., 2014).

Şu anda, ağ ortamlarındaki kullanıcıların katlanarak artması nedeniyle, istenen hizmet kalitesi seviyesine ulaşmak imkânsız hale geldi. Ağlarda tıkanıklık oluşması, paket kaybına ve gecikme sürelerinin artmasına neden olur. Ayrıca, ağ yapısının heterojenliğinin bir sonucu olarak kaynakların daha etkin tahsisi sorunu ortaya çıkmıştır. Uygulamalar arasındaki ağ trafiği tercihlerindeki çeşitlilik, her uygulama için hizmet kalitesi özelliklerine duyulan ihtiyacı ortaya çıkarmıştır. Bu tür talepleri, verimli ve güvenilir bir şekilde sunmak zorlu bir konu olarak ortaya çıkmaktadır.

İstenilen hizmet kalitesi seviyelerine ulaşmanın ve sürdürmenin karmaşıklığı, ağ koşullarının dinamik doğasıyla daha da artar. Ağlar, trafik modellerindeki, kullanıcı gereksinimlerindeki ve ağ topolojilerindeki değişiklikler dahil olmak üzere dinamik koşullara tabidir. Sürekli değişen koşullar karşısında tutarlı bir hizmet kalitesini sürdürmek için trafik yönetimi, kaynak tahsisi ve yönlendirme optimizasyonu gibi gelişmiş teknikler

kullanmak gerekir. Hizmet kalitesini korurken hem güvenlik hem de gizliliğe ulaşmak, incelikli ve detaylı bir yaklaşım gerektirir. Şifreleme ve kimlik doğrulama gibi güçlü güvenlik protokollerinin uygulanması, ağ sistemleri üzerinde önemli bir yük oluşturarak potansiyel olarak ağ performansını etkileyebilir. Modern ağ ortamlarının karmaşıklıklarını ve çeşitli ihtiyaçlarını etkili bir şekilde ele almak ve hizmet kalitesini iyileştirmek için yenilikçi yaklaşımlar, ileri teknolojiler ve etkili yönetim stratejileri kullanmak gereklidir.

1.3 Çoklu Alan Ağları

Dijital ağlarda gelişmiş bağlantı hizmetleri, çok alanlı iletişimlerin sağlanması için bir yetenek gerektirir. Çoklu alan ağları, birbirine bağlı birden çok alandan oluşan bir ağ topolojisi türüdür. Bu bölgeler genellikle yönlendiriciler aracılığıyla bağlanır ve ölçeklenebilir ve verimli bir ağ tasarımı oluşturmak için kullanılır.

Çoklu Alan Ağlarının Özellikleri:

Ölçeklenebilirlik: Çoklu alan ağları, ağın büyümesine ve daha etkili bir şekilde birçok yönlendirici ile ağı yönetmeye olanak tanır. Ağı daha küçük bölgelere bölmek, her bölgenin bağımsız olarak yönetilmesini sağlar ve genel ağın karmaşıklığını azaltır.

Azaltılmış En Kısa Yol Algoritması Hesaplama: Tek bir büyük ağda, en kısa yol hesaplamaları, özellikle ağın boyutu arttıkça hesaplama gücü açısından yoğun olabilir. Çoklu alan ağları, en kısa yol hesaplamasını daha küçük bölgelere böler ve yol hesaplamaları için gereken zaman ve kaynakları azaltır. Algoritma olarak pekiştirmeli öğrenme (Reinforcement Learning) algoritmasının daha hızlı ve verimli olabildiğini bu çalışmamızda önermekteyiz.

Daha Hızlı Yakınsama: Ağda bir değişiklik meydana geldiğinde, örneğin bir bağlantı hatası veya yeni bir ağ eklendiğinde, çoklu alan ağları daha hızlı bir şekilde yakınsar. Sadece etkilenen alan algoritma hesaplamasını yeniden yapar ve değişiklikler diğer bölgelere iletilir.

Azaltılmış Bağlantı Durumu İlanı Yükü: Yönlendirme algoritmalarında, yönlendiriciler yönlendirme bilgilerini paylaşmak için bağlantı durum ilanları değiş tokuş ederler. Tek bir bölgede, bu ağ bant genişliğini tüketen yüksek bir bağlantı durum ilanı hacmi oluşturabilir. Çoklu alan ağları, bağlantı durum ilanları bireysel bölgelerin içinde tutarak bu yükü azaltmaktadır.

Daha İyi Ağ Tasarımı ve Kontrol: Çoklu alan ağları, ağ yöneticilerinin güvenlik politikaları, trafik yönlendirme ve adresleme düzenleri gibi belirli gereksinimlere göre farklı alanların tasarlanması ve kontrol etmesine olanak tanır.

Çok bölgeli bir ağın en yaygın örneği, büyük ağlarda yaygın olarak kullanılan Önce En Kısa Yolu Öncelikli (OSPF – Open Shortest Path First) yönlendirme protokolüdür. Fakat yönlendirme algoritması olarak pekiştirmeli öğrenme kullanımının daha işlevsel olduğunu çalışmamızda önermekteyiz.

1.4 Pekiştirmeli Öğrenme

Pekiştirmeli öğrenme (Reinforcement Learning - RL) algoritmaları, dinamik bir ortamdan modelsiz en uygun yönergeleri türetmeyi amaçlayan yapay zeka alanındaki bir makine öğrenme paradigmasıdır. Bir ajanın çevre ile etkileşim kurarak kararlar almasını öğrendiği bir yöntemdir. Ajan, çevredeki eylemlerle belirli bir hedefi gerçekleştirmeyi veya zamanla birikimli ödülü maksimize etmeyi amaçlar. Ajan, eylemleri sonucunda ödül veya ceza alarak deneme yanılma yoluyla öğrenir (Sutton ve Barto, 2018). Altta yatan geçiş olasılıkları bilinmediğinden, RL ajan çevre ile sürekli etkileşime girerek, karar verme dizisinin beklenen azalan ödülü en üst düzeye çıkaran eylemleri seçmesine izin verir. RL'nin dört ana öğesi vardır: Durum, Eylem, Çevre ve Ödül. RL temsilcileri, uygun politikaya göre eylemleri seçer ve anında ödül alır. Temsilci her adımda yeni durumu gördükçe, çevre ödüle göre güncellenir ve bir sonraki eylem seçilir. Temsilcinin hedefi, uzun vadede maksimum indirimli ödüle ulaşmaktır (Chen ve Wang, 2020).

1.4.1 Pekiştirmeli Öğrenme'nin Bileşenleri:

1. Ajan (Agent): Ajan, pekiştirmeli öğrenme sisteminin merkezinde yer alan aktif öğrenen varlıktır. Ajan, çevreyle etkileşim kurar ve hedefine ulaşmak veya maksimum ödülü elde etmek için aksiyonlar alır. Ajan, belirli bir görevi yerine getirmek veya belirli bir performans ölçütünü optimize etmek için öğrenir.

2. Çevre (Environment): Çevre, ajanın etkileşimde bulunduğu ve öğrenme sürecinin gerçekleştiği dış dünyayı veya bir problem alanını temsil eder. Çevre, ajanın her aksiyonundan sonra yanıt verir ve ajanın algoritmasının geri bildirim alması için gereklidir.

3. Durum (State): Durum, ajanın çevreyi algıladığı ve kararlar aldığı zaman anlık

bilgiyi temsil eder. Durum, çevreyle ilgili tüm önemli bilgileri içerir ve ajanın nasıl hareket etmesi gerektiği konusunda karar vermesine yardımcı olur. Durumlar, çeşitli özelliklerin veya gözlemlerin birleşimi olabilir ve ajanın çevre hakkındaki anlayışını temsil eder.

4. Eylem (Action): Eylemler, ajanın çevrede gerçekleştirdiği işlemleri veya kararları ifade eder. Ajan, belirli bir durumda mevcut olan eylem kümesinden bir eylem seçer. Eylem kümesi, probleme bağlı olarak farklılık gösterebilir ve ajanın yapabileceği olası hareketlerin tamamını içerir.

5. Ödül (Reward): Ödül, ajanın bir eylem gerçekleştirdikten sonra çevreden aldığı anlık geri bildirimdir. Ajanın amacı, uzun vadede maksimum toplam ödülü elde etmektir. Ödüller, ajanın başarılı veya başarısız olduğunu belirleyen motivasyon sağlar.

6. Politika (Policy): Politika, ajanın belirli bir durumda hangi eylemi seçeceğini tanımlayan stratejidir. Politika, durumları eylemlere eşleyen bir fonksiyon veya algoritma olarak ifade edilir. Ajan, politika aracılığıyla çevreye etki eder ve ödül kazanır.

7. Değer Fonksiyonu (Value Function): Değer fonksiyonu, ajanın bir durumda ne kadar iyi olduğunu veya bir eylem sonrasında ne kadar ödül beklediğini tahmin eden bir fonksiyondur. Değer fonksiyonu, ajanın politikasını güncelleme ve en iyi eylemleri seçme konusunda rehberlik eder.

Pekiştirmeli öğrenme, bu bileşenlerin etkileşimi yoluyla ajanların belirli görevleri öğrenmesini ve çevrelerle etkileşimde bulunarak en iyi stratejileri geliştirmesini sağlayan bir öğrenme yaklaşımıdır. Ajan, çevreden aldığı geri bildirimlere göre politikasını güncelleyerek ve değer fonksiyonunu tahmin ederek sürekli olarak öğrenir ve gelişir. Bu sayede, karmaşık ve gerçek dünya problemlerini çözmek için kullanılarak çeşitli uygulama alanlarında büyük başarılar elde edebilir.

1.4.2 Pekiştirmeli Öğrenme'nin Temel Kavramları:

1. Hedef (Goal) veya Görev (Task): Pekiştirmeli öğrenme de, ajanın ulaşmak istediği bir hedefi veya tamamlamak istediği bir görevi bulunur. Hedef, genellikle belirli bir durumda veya durum dizisinde elde edilmesi amaçlanan ödül miktarını maksimize etmek veya belirli bir davranışı öğrenmektedir.

2. Durum (State): Durum, pekiştirmeli öğrenme ortamında ajanın bulunduğu zamandaki durumu veya çevre ile etkileşim halindeki anlık durumu ifade eder. Durum, ajanın çevre hakkında bilgi sahibi olduğu gözlemlenebilir değişkenlerin bir araya

gelmesiyle oluşur. Ajan, durumu algılayarak ve anlayarak kararlarını verir.

3. Eylem (Action): Eylem, ajanın durumu temel alarak çevrede gerçekleştirdiği işlemleri veya hareketleri ifade eder. Ajan, belirli bir durumda mevcut olan eylem kümesinden bir eylem seçer. Eylem kümesi, problem ve çevre bağlamına göre farklılık gösterebilir.

4. Politika (Policy): Politika, ajanın ortamın durumunu göz önünde bulundurarak hangi eylemi seçeceğini belirleyen stratejidir. Politika, durumları eylemlere eşleyen bir fonksiyon veya olasılık dağılımı olarak ifade edilir. Ajan, politika sayesinde çevre ile etkileşime girer ve ödüller kazanır.

5. Ödül (Reward): Ödül, ajanın bir eylem gerçekleştirdikten sonra çevreden aldığı anlık geri bildirimdir. Ödül, ajanın ne kadar iyi bir performans gösterdiğini veya hedefine ne kadar yaklaştığını belirleyen motivasyon sağlar. Ajanın amacı, toplamda maksimum ödülü elde etmek için stratejisini ve politikasını geliştirmektir.

6. Değer Fonksiyonu (Value Function): Değer fonksiyonu, ajanın belirli bir durumda ne kadar "iyi" veya "değerli" olduğunu tahmin eden bir fonksiyondur. Değer fonksiyonu, ajanın uzun vadeli hedefleri için değerlendirmeler yapmasına yardımcı olur. Değer fonksiyonu, ajanın politikasını güncelleme ve en iyi eylemleri seçme konusunda kılavuzluk eder.

7. Model (Model): Model, ajanın çevreyle etkileşimini önceden tahmin etmek için kullanılan matematiksel bir yapıdır. Model tabanlı pekiştirmeli öğrenme yaklaşımlarında, ajan çevreyi simüle edebilir veya çevre davranışını tahmin edebilen bir modeli kullanarak kararlar alabilir ve öğrenebilir.

8. Geri Bildirim (Feedback) ve Gecikmiş Ödüller (Delayed Rewards): Ajan, her eylem sonrasında çevreden geri bildirim alır, bu geri bildirim ödül veya ceza şeklinde olabilir. Gecikmiş ödüllerde ise ödüller zamanla veya birkaç adım sonra elde edilebilir. Ajanın hedefine ulaşmak için gecikmiş ödülleri doğru bir şekilde değerlendirmesi ve kullanması önemlidir.

Pekiştirmeli öğrenme, ajanların hedeflerini gerçekleştirmek için etkili stratejiler geliştirmek ve çevre ile etkileşime girerek ödüller kazanmak için kullanılan bir öğrenme paradigmasıdır. Bu temel kavramlar, ajanların belirli görevleri başarmak için nasıl öğrenme yaptığını ve çevreleriyle nasıl etkileşime girdiğini anlamak için önemlidir. Pekiştirmeli öğrenme algoritmaları, ajanın çevre ile etkileşim yoluyla en iyi kararları almasını ve hedefini elde etmesini sağlamak için kullanılan yöntemlerdir. Bu algoritmalar,

ajanın politikasını ve değer fonksiyonunu güncelleyerek ve çevreden aldığı geri bildirimi kullanarak öğrenme sürecini yönlendirebilmektedir.

1.5 Blokzincir

Blokzincir, dağıtık ve merkezi olmayan bir veritabanı teknolojisidir ve verilerin güvenli ve şeffaf bir şekilde saklanmasını ve paylaşılmasını sağlar. Temelde, şifreleme yöntemiyle birbirine bağlanmış bloklardan oluşan bir kayıtlar listesidir. Her blok, bir dizi veri, bloğun benzersiz tanımlayıcısı olan bir hash ve önceki bloğun hash'i ile birbirine bağlanır, böylece bloklar bir zincir oluşturur. Blokzincir teknolojisi başlangıçta kripto para birimi Bitcoin'in temel teknolojisi olarak tanıtılmış olsa da, kripto paraların ötesinde çeşitli endüstrilerde kullanım potansiyeline sahiptir (Nofer vd., 2017).

Blokzincir, kamuya açık ve değiştirilemez bir defter olarak işlev görür, yani veriler bir kez blok zincirine kaydedildiğinde, çoğunluk katılımcılarının onayı olmadan değiştirilemez veya silinemez. Bu özellik veri bütünlüğünü sağlar ve işlemleri doğrulamak ve yönetmek için merkezi bir otoriteye ihtiyaç duymayı ortadan kaldırır. Blokzincir tasarımı, dışarıdan gelebilecek müdahaleye karşı dayanıklı, güvenli ve sansürü önleyici bir yapıya sahiptir. Bu nedenle, güven ve şeffaflığın kritik olduğu birçok uygulama için uygundur (Gupta vd., 2020).

Blokzincir, birçok düğümün (bilgisayarın) ağda katılımcı olduğu merkezi olmayan bir ağ modelinde çalışır. Her düğüm, tam blok zincirinin bir kopyasına sahiptir ve verileri ağda tutmak ve doğrulamak için işbirliği yapar. Blokzincir mimarisi merkezi olmadığından, tek bir kontrol noktasına bağımlılıktan doğabilecek riski azaltır ve tek taraflı manipülasyonlara karşı dirençli bir tavır ortaya koyabilmektedir. Tipik bir genel blokzincirde (örneğin Bitcoin) herhangi bir birey düğüm olarak ağa katılabilir ve onaylama sürecine katılabilir.

Konsensüs mekanizması, tüm düğümlerde oluşturulabilecek işlemlerin geçerliliği ve bloklara ekleme sırası konusunda anlaşmaya varmalarını sağlamaktadır. İş Kanıtı (Proof of Work - PoW) ve Hisse Kanıtı (Proof of Stake - PoS) gibi çeşitli konsensüs algoritmaları, düğümler arasında anlaşmayı sağlamak için kullanılır.

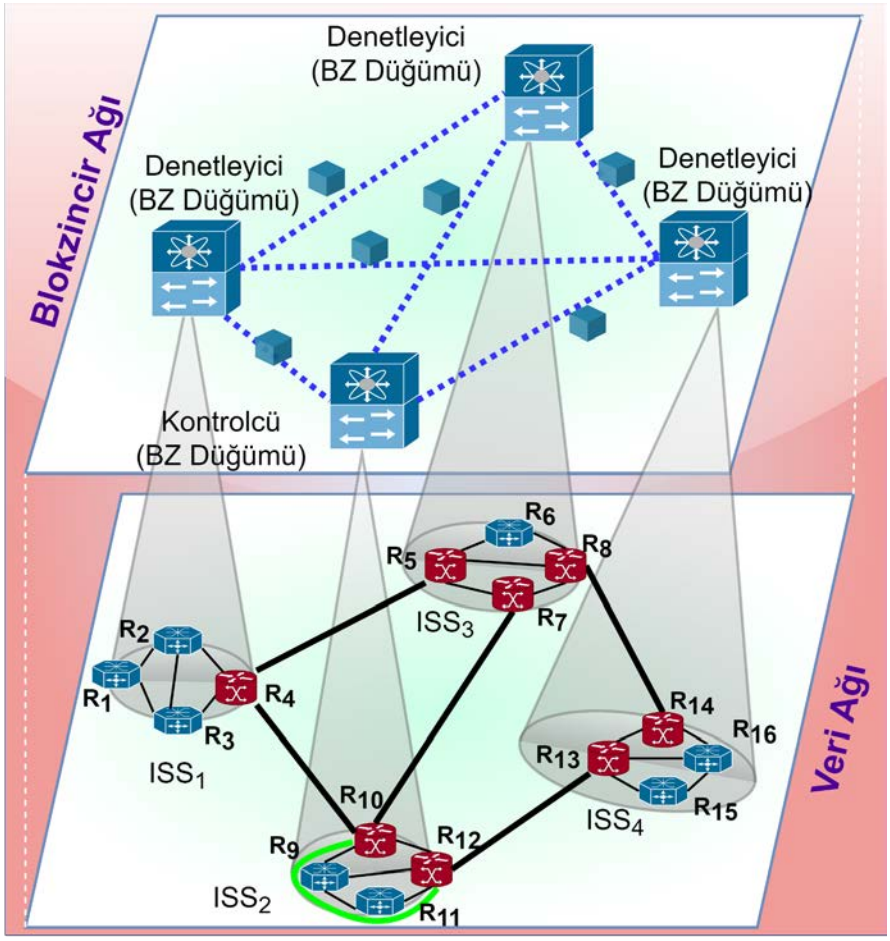
Blok Yapısı: Blokzincir'deki her bir blok, üç temel bileşenden oluşur:

(i) *Veri (Data):* Bu bölüm, blok üzerinde kaydedilen bilgileri içerir. Kripto paraların durumunda, veri genellikle gönderen, alıcı ve miktar gibi işlem ayrıntılarını içerir.

(ii) *Hash:* Hash, bloğun benzersiz tanımlayıcısıdır ve blok içeriğini temsil eden bir şifreleme fonksiyonu kullanılarak hesaplanır. Blok içeriğinde herhangi bir değişiklik, tamamen farklı bir hash değeri üreteceğinden, blok içeriğini değiştirmek neredeyse imkansız hale gelir.

(iii) *Önceki Bloğun Hash'i:* Her blok, bir önceki bloğun hash değeriyle bağlantılıdır. Bu, blokların birbirine bağlı bir zincir oluşturmasını sağlar ve herhangi bir blokta yapılan bir değişiklik, tüm sonraki blokları etkiler, bu nedenle değişiklik farkedilebilir hale gelir.

Özetle, blokzincir, dağıtık ve güvenli bir teknolojidir ve ağdaki düğümler arasında oluşturulan işlemler üzerinde çalışır. Blokzincirdeki her bir blok, veri, benzersiz bir hash ve bir önceki bloğun hash'ini içeren değiştirilemez bir zincirden oluşur. Bu yenilikçi teknoloji, veri yönetimi ve işlemler için güvenli ve şeffaf bir sistem sağlayarak çeşitli endüstrilere devrim niteliğinde bir potansiyel sunar.



Şekil 2. Blokzincir Entegre Edilmiş Yazılım Tanımlı Çoklu Ağlar Yapısı

Blokzincir teknolojisinin entegrasyonu, yönlendirme mekanizmalarını desteklediği için Yazılım Tanımlı Çoklu Ağlar hizmet kalitesini artırma potansiyeline sahiptir (Karakus vd., 2021). Blokzincir teknolojisi, güvenli ve dağıtılmış bir defteri tutmak için merkezi olmayan ve şeffaf bir sistem sağlar. Blokzincir teknolojisinin kullanımı, ağ trafiğinin ve yönlendirme belirlemelerinin güvenli belgelenmesi ve yayılması için geçerli bir çözüm sunar. Hizmet kalitesi yönetiminin bir blokzincir platformunda uygulanması, tüm ağ katılımcılarının güncel ve gerçek bilgiler elde etmesini sağlayan şeffaf bir ekosistemi teşvik eder. Ayrıca, blokzincir teknolojisinin merkezi olmayan mimarisi, ağ içindeki güvenlik açıklarını azaltır ve hizmet kalitesini yükseltir. Pekiştirmeli öğrenme ve blokzincirin entegrasyonu, Yazılım Tanımlı Çoklu Ağlar'da hizmet etkin yönlendirme kalitesini artırmayı amaçlayan araştırma ve geliştirme için yeni beklentiler sunuyor. Şekil 2, blokzincir entegrasyonu tamamlanmış yazılım tanımlı çoklu ağlar mimarisini temsil etmektedir. Burada ağ denetleyicileri blokzincir katılımcıları olarak rol almaktadır.

2. LİTERATÜR ÖZETİ

Hizmet kalitesi destekli yönlendirmeyi geliştirmek için Yazılım Tanımlı Çoklu Ağlarda ölçeklenebilirlik sorunlarını ele alıp bunların SDN'ye özgü olmadığını savunuldu. Farklı ortamlarda ortaya çıkan ortak endişeleri keşfedip SDN tasarım alanındaki ölçeklenebilirlik kısımlarını tartışıyor ve SDN ölçeklenebilirliği üzerine bazı yeni araştırmalar yapıldı. Yaygın olarak kullanılan performans ölçümlerinin ötesinde önemli ancak gözden kaçan ölçeklenebilirlik fırsatlarını ve zorluklarını da listelediler (Software Defined Multiple Networks- SDMN) (Farhady vd., 2015). İnternet trafik mühendisliğini yönlendirme optimizasyonu açısından incelendiği 1990'ların sonlarında trafik mühendisliği kavramının ortaya çıkışından kalma, literatürdeki yönlendirme algoritmalarının bir araştırması olup algoritmalar çok boyutlu olarak sınıflandırılmıştır (Zhang vd., 2018). Pekiştirmeli öğrenme (Reinforcement Learning – RL) ,bir ajanın dinamik bir ortamla deneme yanılma etkileşimleri yoluyla bir davranışı öğrenirken karşılaştığı zorluktur. (Yeganeh vd., 2013) Yaptıkları çalışma psikolojidekine benzer, ancak ayrıntı ve kullanım açısından önemli ölçüde farklılık gösterir. Ticari veya keşif ve madencilik de dahil olmak üzere pekiştirmeli öğrenmenin temellerini tartışırken Markov karar teorisi aracılığıyla alan için temel atıyor, pekiştirmeli öğrenmeyi geciktiriyor, genellemeyi kullanarak öğrenmeyi hızlandırmak için ampirik modeller inşa ediyor (Kaelbling vd., 1996). Yapay zekadaki en aktif araştırma alanlarından biri olan pekiştirmeli öğrenme, karmaşık ve belirsiz bir ortamla etkileşim kurarken aldığı toplam ödülü en üst düzeye çıkarmaya çalıştığı, öğrenmeye yönelik hesaplamalı bir yaklaşımdır (Sutton ve Barto, 2018).

OpenFlow, entegre bir akış tablosuna ve trafik girişlerini eklemek ve kaldırmak için standartlaştırılmış bir arayüze sahip bir Ethernet anahtarına dayalıdır. Kullanımdaki amaç ağ sağlayıcılarını, sistem omurgalarına ve kablo dolaplarına dağıtım için anahtarlama ürünlerine OpenFlow eklemeye teşvik etmektir. OpenFlow'un pragmatik bir uzlaşma olduğunu, araştırmacıların yüksek hat hızları ve bağlantı noktası yoğunlukları ile heterojen anahtarlar üzerinde deneyler yapmasını, öte yandan, satıcıların anahtarlarının iç işleyişini ifşa etmesi gerekmez (McKeown vd., 2008). Otonom araçlardaki yolcular, kimliğe bürünme, hizmet reddi, zamana dayalı saldırılar ve solucan delikleri gibi birçok güvenlik, gizlilik ve güvenlik sorunuyla karşı karşıya kalmış ve bunları kullanmaktan korkan birçok soru ve endişeye yol açmıştır. Kimlik doğrulama, hesap verebilirlik ve hizmet

kullanılabilirliğini kullanarak otonom araçlara yönelik tehdit sınıflandırmasını analiz ettikten sonra uygulama sorunlarının blockchain ile sorunların üstesinden nasıl gelinebileceği anlaşılmıştır. (Gupta vd., 2020)

Geleneksel yönlendirme teknikleri, önceden belirlenmiş algoritmik kurallara bağlıdır ve uyarlanabilirlik ve kişiselleştirme konusunda kısıtlamalar gerektirir. Pekiştirmeli öğrenme ve blokzinciri gibi en son teknolojilerin dahil edilmesi, yazılım tanımlı çoklu ağlar hizmet kalitesi etkin yönlendirmeyi geliştirme potansiyeline sahiptir. Pekiştirmeli öğrenme kullanılması, bir ağ içindeki çeşitli trafiğin analizini mümkün kılar ve yönlendirme kararlarının optimizasyonunu kolaylaştırır. Bu yaklaşım, ağ içindeki çeşitli durumları gözlemleyerek ve bunlarla karşılaşarak bilgi edinir. Pekiştirmeli öğrenme algoritmaları, ağ trafiğini dinamik olarak izleme, hizmet kalitesi gereksinimlerini değerlendirme ve ağ kaynaklarını optimize etmek amacıyla en uygun yönlendirme kararlarını belirleme yeteneğine sahiptir. Bu yaklaşımı uygulayarak ağ performansını iyileştirmek, bant genişliği kullanımını optimize etmek ve kullanıcı deneyimini iyileştirmek mümkündür (Chen ve Wang, 2020).

Akıllı, güvenli ve verimli çerçeveler oluşturmak için blokzincir, SDN ve nesnelerin internetinin teknolojilerini birleştiren altı temel uygulama olan güvenlik, bilgi işlem paradigmaları, güven yönetimi, erişim kontrolü ve kimlik doğrulama, gizlilik ve ağ iletişimi hedefine ve fikrine göre kategorize edilir. Çalışma, pekiştirmeli öğrenme ve blokzincir teknolojilerinin entegrasyonu yoluyla ağ performansını ve kullanıcı deneyimini geliştirmeyi amaçlayan yazılım tanımlı çoklu ağlar yönlendirme çerçevelerinin genel olası mimarilerinin omurgasını sunmaktadır. Çalışma, ilgili mimarilerin olması gereken denetleyici modüllerini ve görevlerini, gerekli ağ denetleyici uygulamalarının fonksiyonlarının çerçevesini çizmektedir. Bu tür ağ mimarilerinde ağ katılımcıları, pekiştirmeli öğrenme çerçevesinden yararlanarak ve blokzincir teknolojisini birleştirerek, hizmet sağlayıcı yetkilendirmesi sınırları dahilinde güncel ve doğrulanmış bilgiler elde edebilir (Guler vd., 2023).

3. MATERYAL VE METOT

3.1 MODÜLLER

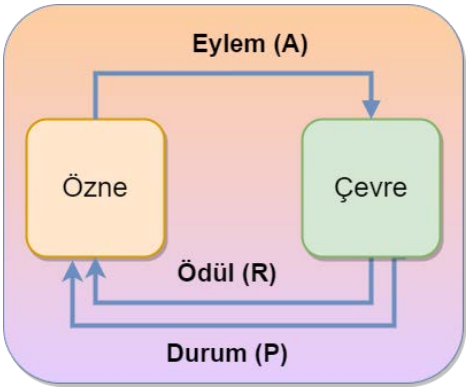
3.1.1 Hizmet Kalitesi Yönetim Modülü

Hizmet Kalitesi Yönetim Modülü, hizmet kalitesi metriklerini gerçek zamanlı olarak toplama, hizmet kalitesi politikalarını yürütme, hizmet kalitesi performansını değerlendirme ve bunlarla ilgili raporlar oluşturma işlevine hizmet eder. Söz konusu modül, ağ içerisinde hizmet kalitesinin yönetilmesi ve optimize edilmesinde çok önemli bir operasyonel rol üstlenmektedir. Başlangıçta modül, hizmet kalitesi ölçümlerinin gerçek zamanlı olarak alınmasını kolaylaştırır. Yukarıda bahsedilen ölçümler, ağ içindeki bant genişliği, gecikme ve paket kaybı dahil edilmekle birlikte, hizmet kalitesi parametrelerini değerlendirmek ve tutarlı bir şekilde gözlemlemek için kullanılır. Böylece, ağ içindeki hizmet kalitesi seviyesini tespit etmek ve mevcut performansını değerlendirmek mümkündür. Modülün önemli işlevi, hizmet kalitesi politikalarının uygulanmasını kapsar. Modül, ağ içindeki çeşitli hizmet sınıfları ve uygulamaları için önceliklerin tahsisi, bant genişliği ve trafik şekillendirmeyi içeren hizmet kalitesi politikalarını belirlemek ve uygulamaktan sorumludur. Bu politikalar, çeşitli hizmet taleplerini karşılamak ve ağ kaynaklarının kullanımını optimize etmek için uygulanır. Son olarak, modül hizmet kalitesi performansını değerlendirir ve ilgili raporları üretir. Toplanan hizmet kalitesi verileri analize tabi tutulur ve ardından performans ölçütleri, istatistikler ve raporlar şeklinde sunulmaktadır. Yukarıda belirtilen raporlar, ağ yöneticilerine hizmet kalitesi performansını izleme, sorunları belirleme ve gerektiğinde düzeltici önlemleri uygulama konusunda önemli bilgiler sağlar. Yazılım tanımlı çoklu ağlar ortamında hizmet kalitesi yönetilmesinden sorumlu olan modül, Hizmet Kalitesi Yönetim Modülüdür. Birincil işlevi, verimli hizmet kalitesi yönetimini sağlamak ve böylece istenen hizmet kalitesini korumaktır.

3.1.2 Pekiştirmeli Öğrenme Modülü

Pekiştirmeli öğrenme (Reinforcement Learning - RL) Modülü, ağ verilerini incelemek ve hizmet kalitesi önkoşulları ve geçmiş verileri temel alan makul yönlendirme belirlemelerini

formüle etmekle görevlidir. Mevcut modül, doğal çevreyi araştırır ve ağ içindeki verileri inceleyerek ampirik araştırmalar yürütür. Pekiştirmeli öğrenme modülü, veri ön işleme, model eğitimi, tahmin oluşturma ve devam eden model iyileştirme dahil olmak üzere birkaç temel işlemi kapsar. Başlangıçta, ağ verilerini manipüle etmek ve daha fazla analiz için uygun bir biçime dönüştürerek veri ön işleme işlevi kullanılır. İlk aşama, gürültüyü en aza indirmek, veri istikrarsızlığını azaltmak ve girdilerin uygun şekilde işlenmesini garanti etmek için verilerin temizlenmesini ve düzenlenmesini içerir. Daha sonra modelin eğitim süreci gerçekleştirilir. Pekiştirmeli öğrenme algoritması, bu fonksiyon içinde önceden işlenmiş veri setini kullanarak bir model oluşturur. Etmenin çevreyle olan etkileşiminde karar verme süreci, ödül ve ceza mekanizmaları aracılığıyla güncellemelere tabi olan bir model tarafından yönlendirilir. Ardından, tahmin oluşturma işlevi yürütülür. Tahmine dayalı model, optimum yönlendirme kararlarını belirlemek için ağ durumu ve hizmet kalitesi gereksinimleri gibi girdileri kullanır. Sonuçta, modeli geliştirme süreci sürekli olarak yürütülür. Modül, gelişen ağ koşullarına ve yeni veri girişlerine yanıt olarak modelin performansını iyileştirmek için sürekli güncellemeler ve geliştirmelerden geçer. Bu yaklaşım, tahminde gelişmiş hassasiyet ve etkinlik sağlar, böylece ağ performansının optimizasyonunu ve hizmet kalitesi kriterlerinin karşılanmasını sağlar.



Şekil 3. Pekiştirmeli Öğrenme Mimarisi

Pekiştirmeli öğrenme modülü, bir ağ içinde veri işleme, model eğitimi, tahmin oluşturma ve devam eden model geliştirme için çok önemli bir araçtır. Şekil 3'te pekiştirmeli öğrenme mimarisinin döngüsü gösterilmektedir.

3.1.3 Blokzincir Modülü

Blokzincir Modülü, yazılım tanımlı çoklu ağlar güvenli ve şeffaf işlemleri destekleyen temel bir unsurdur. Mevcut modül, farklı düğümler arasında bir blokzincir ağı kurarak güvenli bir iletişim ve veri alışverişi aracını kolaylaştırır. Blokzincir teknolojisi, merkezi bir yönetim organına olan ihtiyacı ortadan kaldırarak merkezi olmayan bir veri depolama yapısı sağlar. Böylece, tüm ağ düğümleri arasında veri paylaşımı için güvenli bir mekanizma kurulur ve verilerin güvenilirliği ve bütünlüğü garanti edilir. Hizmet kalitesi verileri, blokzincir ağında güvenli bir şekilde saklanır ve değiştirilemez şekilde kaydedilir. Veri koruma önlemlerinin uygulanması, hizmet kalitesi performansını değerlendirmek için güvenilir bir yol sağlarken potansiyel veri manipülasyonuna karşı koruma sağlar.

Blokzincir modülü, farklı düğümler arasında fikir birliği protokollerini içerir. Blokzincir ağı, işlemleri doğrulamak ve güvenliği garanti etmek için mutabakat mekanizmalarını kullanır. Düğümler arasında fikir birliği sağlanır ve işlemler işlem kanıtı veya hisse kanıtı gibi mekanizmalar kullanılarak doğrulanır. Yukarıda belirtilen önlemler, ağ içindeki işlemlerin güvenilirliğini, kesinliğini ve adil bir şekilde yürütülmesini garanti eder. Blokzincir modülü, güvenilir ve denetlenebilir bir hizmet kalitesi yönlendirme prosedürü oluşturmak için çeşitli mutabakat mekanizmalarını kullanır.

Blokzincir modülü, diğer modüllerle entegrasyonu sayesinde kapsamlı işlevsellik sergiler. Blokzincir ağı, Hizmet Kalitesi Yönetim Modülü ile entegrasyon yoluyla hizmet kalitesi ile ilgili verileri güvenli bir şekilde kaydedebilir. Benzer şekilde, diğer modüllerin blokzincir teknolojisinin avantajlarından yararlanmasını sağlar. Bu, hizmet kalitesi yönetimi için önemli bir mekanizma sunar ve aynı anda ağın kapsamlı güvenliğini artırır. Blokzincir modülü, yazılım tanımlı çoklu ağlar güvenli ve şeffaf işlemlerin sağlanmasında çok önemlidir ve hizmet kalitesi artırmada etkilidir.

Yönlendirme mimarisinde kullanılan yazılım tanımlı ağ denetleyicisindeki blok zinciri yetenekleri için, Şekil 1 yeni denetleyici modüllerinin yanı sıra mevcut olanları ve uygulanan ağ uygulamalarını göstermektedir.

Bir ağ denetleyicisindeki Blokzincir Yönetici modülü, blokzincir ile ilgili tüm işlemlerden sorumludur. Doğrulayıcı Aracı, blok zincirinin blok doğrulama kurallarına dayanarak, diğer denetleyicilerden gelen blokları doğrulamak içindir. Blokzincir ağına gönderilmeden önce işlemler ve bloklar, Hashing Agent modülü tarafından hashlenir. Blokzincir ağını oluşturan işlemlerin ve/veya blokların yanı sıra blokzincir ağının mutabakat algoritmasının uygulanması, sırasıyla İşlem/Blok Aracısı ve Mutabakat Protokolü İşleyicisi tarafından gerçekleştirilir. Modül sorumluluğu, bant genişliği ve gecikme gibi hizmet kalitesi ölçümlerinde kullanılan metriklerde herhangi bir değişiklik olduğunda ağ kaynaklarına göz kulak olmak ve blokzincir yöneticisini uyarmak ve Kaynak İzleme Yöneticisinin uygun işlemleri ayarlamaktır.

Küresel Yönlendirme Uygulaması, denetleyici bir ağlar arası hizmet talebi aldığında ağlar arası yönlendirme işlevlerini yerine getirmekten sorumludur. Blokzinciri Uygulama modülü, hizmet talebi mesajlarının işlenmesine ve blokzinciri ağı ile müşterileri arasında blokların aktarılmasına yardımcı olur.

Pathlet (Yol Parçacığı): Pathlet, iki sınır düğümü çiftini giriş ve çıkış uç noktalarında bağlayan açık bir yolun bir bölümüdür. Pathlet uç noktaları, aynı ağda bulunan giriş ve çıkış düğümleridir. Örneğin, iki sınır düğümü arasındaki yol, Şekil 2'de kısmi bir yeşil yol olarak gösterilir ve blokzincirde oluşan işlemlerde sadece uç nokta bilgilerini barındırarak tutulur.

Hizmet İsteği: Yönlendirme çerçevesine göre, bir Hizmet İsteği, bant genişliği ve gecikme gibi belirli hizmet kalitesi parametrelerini kullanan aynı veya farklı ağlardaki kullanıcılar arasındaki bağlantının sağlanması için bir taleptir. Teorik olarak, kullanıcılar herhangi bir hizmet oranı (bant genişliği ve/veya gecikme) isteyebilmeli ve sürekli hızlı bir ağ isteğe bağlı (hizmet) istekleri destekleyebilmelidir. Oluşturulacak çerçevedeki hizmet isteği veri yapıları Şekil 3'te gösterilmiştir. Bir hizmet isteği mesajı aşağıdaki bilgileri içerir:

Şekil 4. Hizmet İsteği veri yapısı

Hizmet Kimliği: Bir hizmet için kalıcı hizmet tanımlayıcısı.

Nonce: Rastgele oluşturulmuş farklı Hizmet İsteği Kimliği.

Kaynak ve Hedef ISS numarası: sırasıyla kaynak/hedef ISS'lerin ISS numaraları.

Kaynak ve Hedef IP: Sırasıyla kaynak ve hedef bilgisayarların IP adresleri.

Bant Genişliği: Bir hizmet talebinin uçtan uca yolu üzerindeki bant genişliği talebi.

Gecikme: uçtan uca yolu üzerinden bir hizmet talebi için kabul edilebilir gecikme.

Yol Önceliği: Yol seçimi tercih sırası hizmet desteği mesajı, kullanıcının bilgisayarındaki ilgili bir program tarafından oluşturulur ve daha sonra ilgili (kaynak) ağ denetleyicisine iletilir. Blokzinciri ağında, bir dizi genel ve özel anahtarla kriptografik işlemler kullanarak, ağdaki her düğüm kendi blok zinciri örneğini çalıştırır ve yönlendirme mimarisinde bir ağ denetleyicisiyle ilişkilidir.

İşlem: Yönlendirme çerçevesinde, blokzincir düğümleri, yollardan ve bunların hizmet kalitesi değerlerinden işlemler üretir. Ağ denetleyicileri, blokzinciri defterine eklemek için ilgili ağlarındaki düğümlere her kenarlık çifti için benzersiz yollar oluşturur. Şekil 5, yönlendirme çerçevesi içindeki bir işlem veri yapısını göstermektedir. Bir işlem aşağıdaki bilgileri içerir:

Tx ID: İşlemin belirgin ID'si

İmza: Özel anahtarını kullanarak işlemin dijital imzasını oluşturan blok zinciri düğümü (yani ISS denetleyicisi).

ISS Numarası: ISS'ler arası yönlendirmede tanımlayıcı olarak kullanılan benzersiz

ISS numarası.

Pathlet ID: Farklı bir pathlet'in benzersiz ID'si.

Giriş ve Çıkış Düğümü: Sırasıyla bir ISP'deki bir yolun bitiş noktaları (yani başlangıç ve bitiş düğümü). Her ISS, sınır düğümü kimliklerini blok zincirine katılan diğer ISS'lerle önceden paylaşacaktır.

Maks Bant Genişliği ve Min Gecikme: Pathlet'in maksimum bant genişliğini karşılaması ve minimum gecikme sağlaması.

Atlama Sayısı: Bir ISS'de karşılık gelen bir yoldaki atlama sayısı.

Şekil 5. Blokzincir özellikli hizmet kalitesi uyumlu ağlar arası yönlendirmedeki bir işlemin veri yapısı

Bir blok zinciri, blok zinciri ağına bağlı bir düğüme işlemin geçerliliğini doğrulayan bir işlem gönderir. Geçersiz olan işlemler iptal edilir. Diğer bağlı düğümler, daha önce düğüm tarafından bilinmeyen geçerli işlemleri alır. İşlemler, bunları daha fazla doğruladıktan ve akranlarına gönderdikten sonra sonunda ağdaki her düğüme ulaşacaktır. İşlem verileri doğrulama kuralları, bir işlemi temsil etmek için hangi verilerin gerekli olduğunu belirler. Çerçevedeki her blokzincir düğümünün, şunları sağlayan bir dizi kural kullanarak işlemleri doğrulaması gerekir: (i) işlemlerin dijital olarak imzalanması gerekir, (ii) bir işlemin hizmet kalitesi ile ilgili bant genişliği ve gecikme alanları pozitif değerlerdir, (iii) ISS numarasının geçerli bir numara olması ve (iv) işlemdeki giriş ve çıkış düğüm kimliklerinin, işlemin üretildiği ağa ait olması gerekir. Şekil 2'de gösterilen ISS2 ağındaki R10 ve R12

sınır cihazları arasındaki pathlet'ler için ISS2 denetleyicisi tarafından oluşturulan işlemleri gösterir. Yalnızca R10 ve R12 arasındaki pathlet'ler için yapılan işlemleri gösterir. Bir ISS blokzincir ağına katıldığında, ilkini yayarak başlar.

Şekilde gösterilen sınır ağ cihazları arasındaki yollar için uygun blok zinciri düğümlerine yapılan işlemler. Denetleyici yeni bir işlem üretir (güncelleme işlemi olarak anılır)bir bağlantıdaki bant genişliği güncellemesi gibi hizmet kalitesi ile ilgili bir ağ durumu değişikliği olduğunda yoldaki durum değişikliğini yansıtır.

3.1.4 Ağ Sanallaştırma Modülü

Ağ Sanallaştırma Modülünü kullanmak, hizmet kalitesi taleplerini karşılamak ve uyarlanabilir kaynak tahsisini sağlamak için çağdaş ağ ayarlarında çok önemlidir (Chowdhury ve Boutaba, 2009). Sanal ağlar, fiziksel ağ kaynaklarının sanal olarak bölünmesini veya ağ katmanlarının oluşturulmasını sağlayan bir mekanizmadır. Bu özellik, farklı hizmet kalitesi özelliklerine dayalı olarak çeşitli kullanıcılara veya hizmetlere uyarlanmış ağ hizmetlerinin sağlanmasını kolaylaştırır. Gerçek zamanlı uygulamaları desteklemek için düşük gecikmeli bir sanal ağ ve genel veri trafiğini barındırmak için yüksek bant genişliğine sahip ayrı bir sanal ağ kurmak mümkündür. Bu, yüksek düzeyde kaynak kullanım verimliliği ve optimizasyonunu garanti eder.

Ağ Sanallaştırma Modülü, dinamik kaynak tahsisi için bir özellik sunar. Bu, ağ kaynaklarının acil gereksinimlere göre otomatik olarak yeniden dağıtılma yeteneğine sahip olduğu anlamına gelir. Bir sanal ağın anlık trafik yükünün arttığı durumlarda, ek bant genişliği veya işleme kapasitesi tahsis etmek mümkündür. Bu özellik, kullanıcıların dinamik hizmet kalitesi taleplerini derhal ele almasını sağlar. Ayrıca, sanal ağların sürekli olarak izlenmesi, ağın genel performansının değerlendirilmesini ve analiz edilmesini sağlar. Bu yaklaşım sayesinde ağ yöneticileri, hizmet kalitesi mevcut durumunu değerlendirebilir, kaynakları gerektiği gibi yeniden atayabilir ve sürekli olarak ağın performansını artırabilir.

Ağ Sanallaştırma Modülü, eş zamanlı olarak hizmet kalitesi özelliklerine uyan ve uyarlanabilir kaynak tahsisini mümkün kılan sanal ağlar kurarken, kapsamlı ağ yapılandırmasında çok önemli bir işlev üstlenir. Bu sistem, diğer modüllerle entegrasyonu

sayesinde, hizmet kalitesi yönetim modülünü kullanarak verilerin sanal ağlara yönlendirilmesini kolaylaştırır ve veri toplama modülünü kullanarak sanal ağların performansını izler. Ayrıca, sanal ağların korunmasını ve tutarlılığını garanti etmek için Güvenlik Modülü ile iş birliği yapar. Ağ Sanallaştırma Modülü, yazılım tanımlı çoklu ağlar hizmet kalitesi taleplerinin karşılanmasında ve kaynak tahsisinin geliştirilmesinde çok önemlidir.

3.1.5 Yönlendirme Optimizasyon Modülü

Yönlendirme Optimizasyon Modülü, hizmet kalitesi metriklerini, ağ koşullarını ve makine öğrenimi teknikleriyle yapılan tahminleri dikkate alarak yönlendirme yollarını optimize etmekten sorumludur (Wang vd., 2008). Mevcut modül, çeşitli ağ kaynakları arasında en uygun yönlendirme yollarını belirlemek için kapsamlı bir veri toplama ve analiz süreci yürütür. Hizmet kalitesi ölçümlerini ve ağ koşullarını değerlendirmek, en uygun yönlendirme rotalarını belirlemede çok önemlidir. Bu, ağ kapasitesi, gecikme süresi ve mevcut kaynakların bant genişliği gibi çeşitli faktörlerin değerlendirilmesini içerir. Bu değerlendirmeden toplanan bilgiler daha sonra en uygun yönlendirme rotalarını belirlemek için kullanılır.

Yönlendirme Optimizasyon Modülü, yönlendirme kararlarını iyileştirmek için makine öğrenimi tekniklerini kullanır. Makine öğrenimi algoritmalarını kullanmak, önceki olayları göz önünde bulundururken mevcut ağ verilerini analiz ederek gelecek trafik modellerini tahmin etmeyi mümkün kılar. Bu tahminlerin kullanılması, yönlendirme belirlemelerinin daha akıllı ve öngörülü bir şekilde uygulanmasını kolaylaştırır. Modül, bir makine öğrenimi modelinin eğitimini kolaylaştırır ve mevcut en son verilere yanıt olarak dinamik olarak uyarlanmasını sağlar.

Yönlendirme Optimizasyon Modülü, dinamik değişikliklere yanıt olarak veri toplama, analiz yapma, makine öğrenimini dahil etme ve yönlendirme kararlarını ayarlama gibi çeşitli işlevleri yerine getirir. Veri toplama süreci boyunca, ağ içindeki çeşitli kaynaklardan elde edilen hizmet kalitesi metrikleri, ağ durumu ve trafik verileri üzerinde bir analiz yapılır. Yukarıda belirtilen veriler, yönlendirme kararlarının alınması için temel bir temel olarak hizmet eder. Modül, makine öğrenimi modellerinin eğitimini kolaylaştırmak için mevcut verileri kullanır ve yaklaşan trafik modellerinin tahmin edilmesini sağlar.

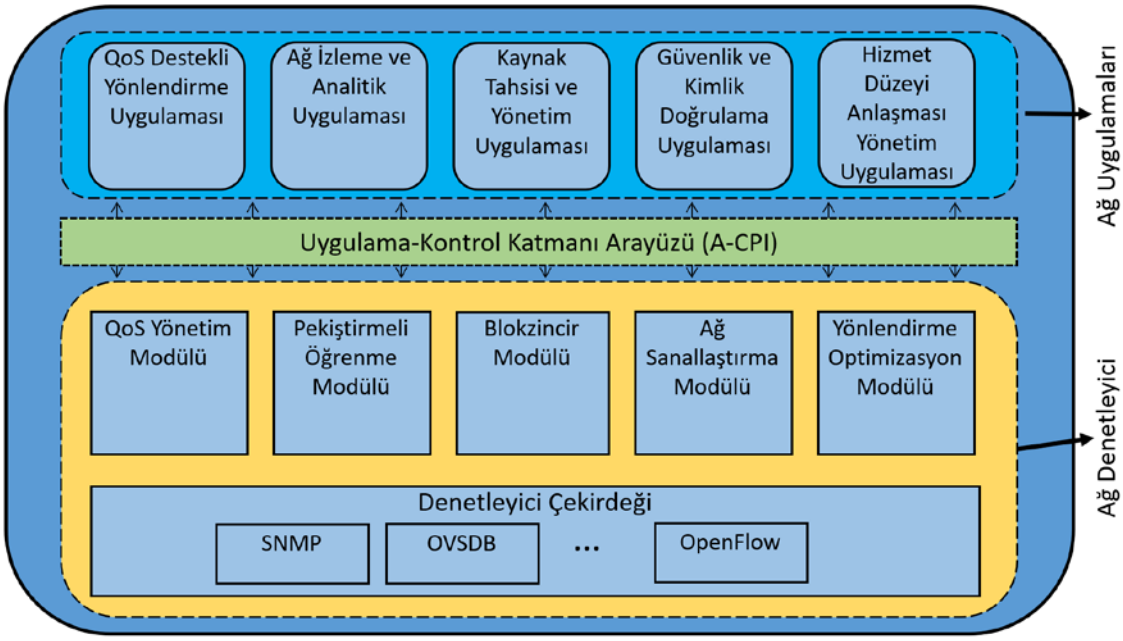
Tahminlerin dinamik yönlendirme kararlarına uyarlanması, yönlendirme optimizasyonunun elde edilmesini sağlar. Yönlendirme Optimizasyon Modülü, yazılım tanımlı çoklu ağlar içinde hizmet kalitesi düzeylerini artırma ve ağ performansını optimize etmede çok önemlidir.

3.2 UYGULAMALAR

3.2.1 Hizmet Kalitesi Destekli Yönlendirme Uygulaması

Hizmet kalitesi destekli yönlendirmenin uygulanması, hizmet kalitesi ön koşullarına dayalı akıllı yönlendirme kararlarının kolaylaştırılmasında çok önemli kabul edilir. Uygulama, ağ için hizmet kalitesi spesifikasyonlarının elde edilmesi, Yönlendirme Optimizasyon Modülünün sonuçlarının kullanılması ve optimize edilmiş yönlendirme yollarının uygulanması dahil olmak üzere çeşitli özelliklere sahiptir. Hizmet Kalitesi -Destekli Yönlendirme Uygulaması, ağ üzerinde çalışan kullanıcılardan veya hizmetlerden hizmet kalitesi özelliklerini almak için tasarlanmıştır. Şartlar, hizmet kalitesi hedefleri gibi kriterlerin yanı sıra gecikme, bant genişliği ve paket kaybı gibi faktörleri kapsar. Uygulama, bireysel hizmet kalitesi gereksinimlerini birleştirerek kullanıcılar veya hizmetler için yönlendirme kararlarını kolaylaştırır.

Yönlendirme Optimizasyon Modülünün çıktıları, bilinçli yönlendirme kararları vermek için bu uygulama tarafından kullanılır. Yönlendirme Optimizasyon Modülü, hizmet kalitesi özelliklerini etkin bir şekilde yerine getirmek için uyarlanmış optimize edilmiş yönlendirme yolları oluşturur. Hizmet Kalitesi - Destekli Yönlendirme Uygulaması, optimize edilmiş yönlendirme rotalarını almaktan ve ardından ağdaki ilgili yönlendirme tablolarını güncellemekten sorumludur. Hizmet Kalitesi - Destekli Yönlendirme Uygulaması, hizmet kalitesi spesifikasyonlarını almak, Yönlendirme Optimizasyon Modülünün sonuçlarını kullanmak ve optimize edilmiş yönlendirme yollarını yürütmek gibi önemli görevleri yerine getirir. Bu özel uygulamanın kullanılması, yazılım tanımlı çoklu ağlar hizmet kalitesi standartlarını garanti etmek ve ağ kaynaklarının tahsisini optimize etmek için gerekli görülmektedir. Şekil 4, yazılım tanımlı çoklu ağlarda makine öğrenmesi ve blokzinciri ile geliştirilmiş servis kalitesi destekli yönlendirme mimarisinde ağ denetleyicisin olası genel yapısını göstermektedir. Bu denetleyicide olması gereken modüller ve ağ uygulamaları gösterilmiştir.



Şekil 6. Yazılım tanımlı çoklu ağlarda makine öğrenmesi ve blokzincir ile geliştirilmiş servis kalitesi destekli yönlendirme mimarisinde ağ denetleyicisin genel yapısı.

3.2.2 Ağ İzleme ve Analitik Uygulaması

Ağ İzleme ve Analitik Uygulaması, gerçek zamanlı ağ performansı ölçümlerini toplamak, ağ davranışını analiz etmek ve hizmet kalitesi ihlallerini belirlemek için kullanılan bir araçtır. Bu uygulama, ağın genel sağlığını değerlendirir ve gerçek zamanlı performans verilerini toplayarak hizmet kalitesi ihlallerini tanımlar. Ağ İzleme ve Analitik Uygulaması, ağdaki veri akışını sürekli olarak gözlemler ve ağ performansına ilişkin çeşitli ölçümler toplar. Metrikler, bant genişliği kullanımı, gecikme süresi ve paket kaybı gibi parametreleri kapsar. Yazılım, toplanan verileri inceler ve ağın davranışını değerlendirir. Bu değerlendirme, hizmet kalitesi spesifikasyonlarına uyumu doğrulamayı, ağ işlevselliği ile ilgili olası sorunları belirlemeyi ve hizmet kalitesi ihlal örneklerini tespit etmeyi amaçlar (Tsai vd., 2018).

Ağ izleme ve analitik uygulaması, toplanan verilerin görselleştirilmesini ve raporlanmasını kolaylaştırır. Görsel grafikler ve raporlar, ağ yöneticilerinin ve sistem operatörlerinin ağdaki performans eğilimlerini ve uç noktaları ayırt etmelerini sağlar. Ayrıca söz konusu yazılım, hizmet kalitesi ihlallerini veya ağ performansındaki düzensizlikleri tespit ettiğinde

bildirim veya uyarı üretme yeteneğine sahiptir. Ağ İzleme ve Analitik Uygulaması, canlı ağ performansı verilerini toplama, ağ davranışını inceleme ve hizmet kalitesi ihlallerini belirleme gibi önemli görevleri yerine getirir. Yukarıda belirtilen uygulama, hizmet kalitesi standartlarını denetlemek ve yazılım tanımlı çoklu ağlar içindeki olası sorunları belirlemek için çok önemlidir.

3.2.3 Kaynak Tahsisi ve Yönetim Uygulaması

Hizmet kalitesi standartlarına ve talebine bağlı kalarak kaynak kullanımını artırmada kaynak tahsis ve yönetim uygulamasının önemi vurgulanmalıdır. Kaynak ayırma isteklerinin işlenmesi, optimizasyon algoritmalarının uygulanması ve dinamik kaynak ayarı dahil olmak üzere işlemleri açıklanmalıdır. Yazılım tanımlı çoklu ağlarda kaynakların verimli kullanımı, kaynak tahsisi ve yönetim uygulamasının kritik bileşenine bağlıdır. Mevcut yazılım uygulaması, kullanıcı taleplerini ve ağ hizmet kalitesi gereksinimlerini analiz ederek kaynak tahsis prosedürlerini yürütür. Yazılım programı, kullanıcılar tarafından gönderilen kaynaklar için istekleri alabilir ve yürütebilir. Bu prosedür, hizmet kalitesi özelliklerini ve mevcut kaynak durumunu dikkate alır (Akella ve Xiong, 2014).

Kaynak tahsis ve yönetim uygulaması, kaynak tahsis sürecini geliştirmek için optimizasyon algoritmalarını kullanır. Bu algoritmalar, mevcut kaynakların optimum tahsisini garanti eder. Yazılım uygulaması, kaynaklar için gereksinimleri değerlendirir ve mevcut kaynakların mevcut durumu, talepleri karşılamak için kaynakların en uygun dağıtımını seçer ve optimizasyon sürecini yürütür. Ayrıca, kaynak tahsisi ve yönetimi uygulaması, kaynakları dinamik olarak ayarlama görevini yürütebilir. Bu işlev, ağ içindeki talep değişikliklerine ve hizmet kalitesi ön koşullarına anında yanıt verilmesini sağlar. Yazılım, kaynakların kullanımını sürekli olarak izler ve kaynakların tahsisini gerektiği gibi dinamik olarak ayarlar. Bu, gereksinimlerin karşılandığını ve hizmet kalitesi hedeflerine ulaşıldığını garanti eder. Kaynak tahsis ve yönetimi uygulaması, hizmet kalitesi gerekliliklerine ve ihtiyaçlarına göre kaynak kullanımının artırılmasında önem taşımaktadır. Yazılım uygulaması, kaynak tahsisi için işleme isteklerini kolaylaştırır, optimizasyon algoritmalarını yürütür ve mevcut kaynakların en verimli şekilde kullanılmasını sağlamak için kaynak tahsisini dinamik olarak uyarlar. Yazılım tanımlı çoklu ağlar verimli kaynak yönetimi ve hizmet kalitesi hedeflerine ulaşılması büyük ölçüde

bu çok önemli bileşene dayanır.

3.2.4 Güvenlik ve Kimlik Doğrulama Uygulaması

Ağ kaynaklarına güvenli erişimin sağlanmasında ve hizmet kalitesi ile ilgili verilerin korunmasında güvenlik ve kimlik doğrulama uygulamasının önemi tartışılmalıdır (Iqbal vd., 2023). Bu, kimlik doğrulama mekanizmaları, şifreleme protokolleri, erişim kontrol politikaları ve güvenlik tehdidi izleme dahil olmak üzere çeşitli işlevlerinin kapsamlı bir şekilde açıklanmasını gerektirir. Güvenlik ve kimlik doğrulama önlemlerinin uygulanması, güvenli bir erişim ortamı sağlamanın ve yazılım tanımlı çoklu ağlar hizmet kalitesi ile ilgili verileri korumanın çok önemli bir yönüdür. Mevcut uygulama, ağ kaynaklarına yetkisiz erişimi önlemek ve güvenlik tehditlerine karşı koruma sağlamak için çeşitli güvenlik mekanizmaları kullanır.

Bahsi geçen uygulamanın temel amacı, çeşitli kimlik doğrulama mekanizmaları kullanılarak hem kullanıcıların hem de cihazların kimliklerinin doğrulanmasıdır. Yetkisiz erişimin engellenmesi ve ağ kaynaklarına yalnızca kimliği doğrulanmış kullanıcılar için erişimin sağlanması büyük önem taşımaktadır. Güvenlik ve kimlik doğrulama uygulaması, kullanıcıların kimliğini doğrulamak için parolalar, anahtarlar ve sertifikalar dahil olmak üzere çeşitli teknikler kullanır. Bahsi geçen uygulamanın ek bir önemli özelliği, şifreleme protokollerinin kullanılması yoluyla bilgilerin güvenli bir şekilde iletilmesini garanti etmektir. Protokollerin kullanılması, verilerin şifrelenmesi ve şifrelerinin çözülmesi sürecini kolaylaştırır, böylece yetkisiz kişilerin verilere erişmesini veya verileri manipüle etmesini engeller. Güvenlik ve kimlik doğrulama uygulaması, ağ kaynaklarına erişimi düzenleyen erişim kontrol ilkelerini uygulamak için tasarlanmıştır. Yukarıda belirtilen politikalar, aynı anda izinleri kısıtlarken hem kullanıcılar hem de cihazlar tarafından belirlenen kaynaklara izin verilen erişim için parametreleri belirler. Sonuç olarak, yalnızca uygun yetkiye sahip kişilere mevcut kaynaklara erişim izni verilir. Güvenlik ve kimlik doğrulama uygulaması, olası güvenlik ihlallerinin izlenmesinden ve tanımlanmasından sorumludur. Bu, ağ içindeki potansiyel güvenlik risklerinin tanımlanmasını ve bunlara yanıt verilmesini kolaylaştırır. Yazılım uygulaması, güvenlik olaylarını denetlemek, olası güvenlik açıklarını belirlemek ve olası saldırıları azaltmak için koruyucu önlemler uygulamak üzere tasarlanmıştır. Güvenlik ve Kimlik Doğrulama önlemlerinin uygulanması, yazılım tanımlı çoklu ağlar içinde güvenli bir erişim ortamını sürdürmek ve

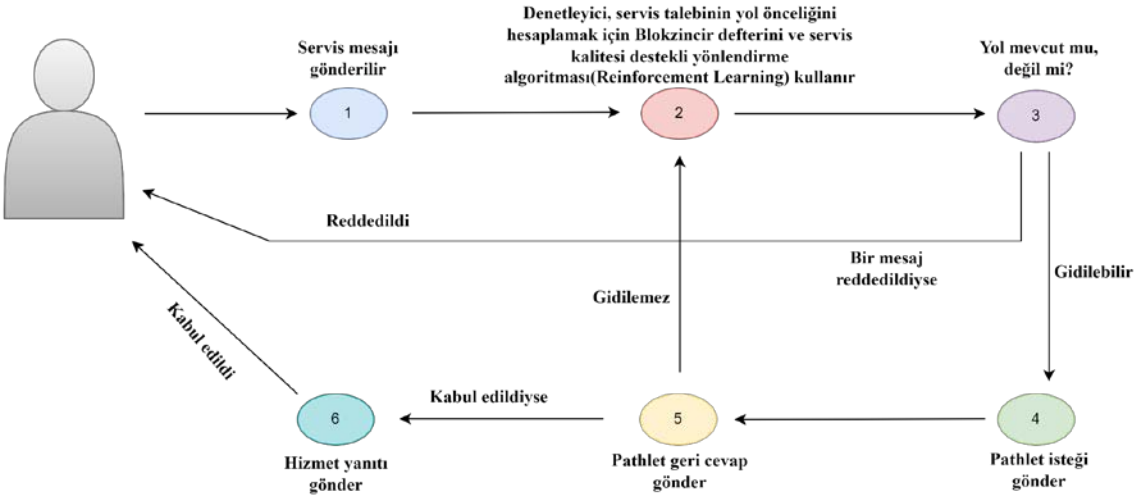
hizmet kalitesi verilerini korumak için son derece önemlidir.

3.2.5 Hizmet Düzeyi Anlaşması Yönetimi Uygulaması

Hizmet Düzeyi Anlaşması (SLA) Yönetim Uygulaması, SLA'ların müzakere edilmesini, özelleştirilmesini, izlenmesini ve uygulanmasını sağlayan çok önemli bir unsurdur. SLA müzakere mekanizmaları, uyumluluk izleme ve raporlama/bildirim özellikleri dahil olmak üzere işlevleri kapsamlı bir şekilde incelenmeli ve tartışılmalıdır. Yazılım tanımlı çoklu ağlar SLA'ların etkin yönetimi ve uygulanması, büyük ölçüde SLA Yönetim Uygulamasının uygulanmasına dayanır. Mevcut uygulama, hizmet sağlayıcılar ve müşteriler arasında hizmet düzeyi anlaşması görüşmelerini mümkün kılmak ve ayrıca kesin SLA gerekliliklerini belirlemek için çeşitli mekanizmalar kullanır (Son vd., 2017).

Yazılımın temel amacı, hizmet düzeyi anlaşması müzakere sürecini denetlemektir. Bu, hizmet düzeyi anlaşması parametrelerinin oluşturulmasını ve hizmet sağlayıcılar ile müşteriler arasındaki müzakerelerin kolaylaştırılmasını kapsar. SLA Yönetim Uygulaması, müşterilerin hizmet ihtiyaçlarını belirleme sürecini kolaylaştırırken, hizmet sağlayıcıların söz konusu gereksinimlere uygun teklifler önermesine olanak tanır. Yukarıda belirtilen uygulamanın ikincil amacı, hizmet düzeyi anlaşması gerekliliklerine uyumu denetlemektir. Yukarıda belirtilenler, hizmet düzeyi anlaşması parametreleri için eş zamanlı gözetim ve manipülasyon mekanizmalarını kapsar. Yazılım uygulaması, hizmet sağlayıcıların SLA hedeflerine uygun olarak hizmet sağlamasını kolaylaştırır ve hizmet düzeyi anlaşması ihlalleri durumunda uygun düzeltici eylemleri uygular. SLA Yönetim Uygulaması, Hizmet Seviyesi Sözleşmelerine ilişkin raporlama ve bildirim işlevleriyle donatılmıştır. Yukarıda belirtilen özellikler, kullanıcılara hizmet düzeyi anlaşması performansıyla ilgili periyodik güncellemeler sağlar ve hizmet sağlayıcılara, müşterileri herhangi bir SLA ihlali veya endişesi konusunda bilgilendirme yetkisi verir. SLA Yönetim Uygulaması, hizmet düzeyi anlaşması müzakeresi ve özelleştirme sürecini mümkün kılarken, uyumluluğun izlenmesini ve yazılım tanımlı çoklu ağlarda SLA'lar için raporlama ve bildirim işlevleri sunulmasını sağlar. Bu uygulama, sağlayıcılar ve müşteriler arasında hizmet seviyelerini yönetmek için güvenilir ve adil bir sürecin kurulmasını kolaylaştırır.

3.3 Pekiştirmeli Öğrenme Tabanlı Yönlendirme Mimarisi Çalışma Alt Yapısı



Şekil 7. Pekiştirmeli Öğrenme Tabanlı Yönlendirme Mimarisi Çalışma Alt Yapısı

Şekil 7, blokzincir teknolojisine dayanan ve bir servis talebinin QoS'sine öncelik veren uçtan uca (End-to-End - uçtan uca) yolu seçim yönteminin izlediği süreci göstermektedir. Adım 2'de, bir kullanıcıdan bir servis talebi aldıktan sonra, ağ denetleyicisi blok zinciri defterini kullanarak bir uçtan uca yolu hesaplar. Servis talebi mesajında belirtilen hizmet kalitesi parametreleri ve yol seçimi öncelikleri (yani, Yol_Önceliği) dikkate alındığında, uçtan uca yolu, bir kaynak ağın uç düğümünü hedef ağın ucundaki bir uç noktaya bağlayan yollardan oluşur. Kullanılabilir uçtan uca yollarını hesapladıktan sonra, hizmet kalitesi tabanlı blokzincir yönlendirme çerçevesi, Adım 3'te kullanılabilir uçtan uca yolu olmadığında kullanıcıya bir Reddetme mesajı göndererek hizmet talebini reddeder. Herhangi bir uçtan uca yolu, servis talebinin gerekli hizmet kalitesi parametrelerini karşılıyorsa, Adım 4'te kaynak ISS'nin denetleyicisi, hesaplanan uçtan uca yoluna bir yolcuğu katkıda bulunan her bir ISS denetleyicisine yol istek mesajları gönderme sürecini başlatır. Adım 5'te, tüm pathlet'ler hizmet kalitesi gereksinimlerini karşılamada başarılı olursa, Adım 4'teki pathlet isteklerinin mesajlarını aldıktan sonra çerçeve, tüm yanıtları bir uçtan uca yolu üzerinden her bir ağ denetleyicisine yayma sürecini başlatır. Adım 6'da çerçeve, karşılık gelen hizmet kalitesi gereksinimleri temelinde "Kabul Et" yazan bir hizmet yanıt mesajı verir. Adım 5'te, kaynak ağın denetleyicisi, gereksinimleri karşılayan başka bir uçtan uca yolu aramaya başlayabilir. uçtan uca yolu üzerindeki herhangi bir ağ denetleyicisi kaynak ağ denetleyicisine bir Reddetme yanıtı gönderirse aynı ölçüt. Ağ içi

yönlendirmelerinin yardımıyla, kaynak ağın ve hedef ağın denetleyicileri, yolun diğer ucundaki ağ ve ana bilgisayar kullanıcı ile kaynak ağın sınır düğümü arasındaki ve hedefin sınır düğümü arasındaki uçtan uca yolunun bölümlerini denetlemekten sorumludur.

4. BULGULAR

Uçtan uca hizmet kalitesi gereksinimlerini göz önünde bulundurarak blokzinzir tabanlı yol bulma işlemi için tanımlanması gereken tüm yapıların nasıl tanımlandığını bu bölümde detaylandıracağız.

İlk olarak blokzincir mimarisinde yolların bilgilerinin tutulduğu ve işlem olarak adlandırılan "Transaction" modülünün tanımlanması gerekmektedir. Bu modülü tanımlarken önceden belirlemiş olduğumuz veri yapısı modelini kullanıyoruz ve gerekli tüm özellikleri sırasıyla tanımlamamız gerekmektedir.

```
public class Transaction {

        private int transactionID;
        private int asNo;
        private int pathletID;
        private String ingress;
        private String egress;
        private double minBandwidth;
        private double maxDelay;
        private List<String> fullpath;
        private boolean status;

        public Transaction(int tID, int as, int pathId) {
                this.transactionID = tID;
                this.asNo = as;
                this.pathletID = pathId;
                this.minBandwidth = 0.0;
                this.maxDelay = 0.0;
                this.fullpath = new ArrayList<String>();
                this.status = true;
        }
```

Şekil 8. Blokzincir İşlem Modülünün Tanımlanması

Şekil 8'de oluşturulan modül ile numerik bir değer olarak her bir işlemin kimliği "transactionID" olarak belirlenmektedir ve bu numara her bir işlem için eşsiz olmasını

gerektirmektedir. Eğer hali hazırda kullanılan bir işlem üzerinde güncelleme yapılması gerekiyorsa, güncellenen işlemin yeni değerlerini tutmak için farklı bir kimlik numarası alması gerekebilmektedir. Her bir işlemin oluşturulduğu internet servis sağlayıcının kimliğini bilmek için ise "asNo" değişkeni numerik olarak tanımlanmaktadır. Her bir işlem içerisinde bulunmakta olan "pathletID" değişkeni, internet servis sağlayıcının giriş ve çıkış düğümleri olan uç noktalarının düğüm numaralarını tutar. İşlemlerde, sadece bilgi olarak blokzincirde tüm denetleyicilerin sahip olabileceği bilgi olan giriş ve çıkış düğüm bilgileri sırasıyla "ingress" ve "egress" olarak düğümlerin numaralarını belirten değerler olarak tutulmaktadır. Her bir işlemde bulunan yolların hizmet kalitesi metrikleri de sırasıyla işlem içerisinde gerekli tanımlamalar yapılarak düzenlemektedir.

Düzenlenen tüm işlemler blokzincir altyapısında oluşturulan bloklar halinde Şekil 9'da gösterildiği gibi tutulmaktadır. Her blok için oluşturulan "Block" modülünde her bir işlem ayrı ayrı tutulmaktadır. Hizmet kalitesi destekli uçtan uca yol bulma işlemini oluşturulan bu blokların sahip olduğu işlemler üzerinden yapmaktayız.

```
public class Block {

        private int index;
        private String timestamp;
        private long millisecond;
        private String data;
        private String previousHash;
        private String hash;
        private int ISPNumber;

        public Block(int index, String data, String previousHash) {
                this.index = index;
                DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd HH:mm:ss");
                LocalDateTime now = LocalDateTime.now();
                this.millisecond = System.currentTimeMillis();
//              this.millisecond = System.nanoTime();
                this.timestamp = dtf.format(now); // get current system time
                this.data = data;
                this.previousHash = previousHash.isEmpty() ? "" : previousHash;
                this.hash = this.calculateHash();
        }
```

Şekil 9. Blok Yapısı

```
public class Blockchain {

	private List<Block> chain = new ArrayList<Block>();
	public Blockchain() {

		this.chain.add(createGenesisBlock());
	}

	public Blockchain(Block block) {

		this.chain.add(block);
	}

	public Block createGenesisBlock() {
		return new Block(0, "Genesis block", "0");
	}

	public Block getLatestBlock() {
		return this.chain.get(this.chain.size()-1);
	}

	public void addBlock(Block newBlock) {
		newBlock.setPreviousHash(this.getLatestBlock().getHash());
		newBlock.setHash(newBlock.calculateHash());
		this.chain.add(newBlock);
	}

	public List<Block> getChain(){
		return this.chain;
	}

	public boolean isChainValid() {

		for (int index = 1; index < this.chain.size(); index++) {

			Block currentBlock = this.chain.get(index);
			Block previousBlock = this.chain.get(index - 1);

			if (!currentBlock.getHash().equals(currentBlock.calculateHash())) {
				return false;
			}

			if (!currentBlock.getPreviousHash().equals(previousBlock.getHash())) {
				return false;
			}

		}

		return true;
	}

	public boolean isEmpty() {

		return this.chain.size()==0?true:false;
	}

	public List<Block> getBlocks() {
		return this.chain;
	}

}
```

Şekil 10. Blok zincir Modülü

Her bir blok sisteme oluşturulduğu zaman dilimi, bloğu oluşturan internet servis sağlayıcının kimlik numarası, bloğun oluşturulan denetleyici tarafından hash dönüşümü ve oluşturulan yeni bloğun bağlantısının sağlanacağı blok zincirdeki en son bloğun hash adresi eklenmektedir. Bu sayede blok zincirin içerisindeki her bir blok bir önceki bloğa bağlantısını yapmış olmaktadır.

Şekil 10'da gösterilen blok zincir modülü ile tüm blokların tek bir bağlantılı zincir olarak sistemde bulunan tüm internet servis sağlayıcı denetleyicileri tarafından tutulması sağlanmaktadır. Bu sayede tüm denetleyiciler birbirinin aynısı olan blok zincirlere sahip olarak, hizmet kalitesi destekli internet servis sağlayıcılar arasındaki uçtan uca yolu blok zincir üzerinde bulabilecektir. Burada işlemlerin yapılabilmesi için blok zincir altyapısında zincirin bozulmaması ve oluşturulan blokların doğru şekilde oluşturulup sisteme eklenebilmesi için konsensüs mekanizması devreye girmektedir. Şekil 11'de belirtilen "BlockchainManager" modülü her bir blok ekleme işleminde onaylama işlemini yapabilmek için tasarlanmıştır. Her bir yönetici, blokları onaylamak için doğrulukları hesaplamaktadır. Bu hesaplamalar Şekil 12'de belirtilen otorite onayı konsensüs modülü ile yapılmaktadır. Konsensüs yapısı içerisinde tüm onaylayıcı olarak blok zincire dahil denetleyiciler, hali hazırda oluşturulan blok ("current") ile blok zincirde bulunan en son ve yeni bloğun bağlanması gereken bloğu ("previous") tanımlayarak hash değerlerinin kontrollerini yapmaktadırlar. Bu sayede, oluşturulan yeni blok modülünün blok zincirdeki diğer bloklar ile uyumluluğu sağlanmış olmakla birlikte sistemin devamlılığı da herhangi bir bozulma olmadan sağlanmış olmaktadır.

```
public class BlockChainManager {

        private int ISPNumber;
        private List<Block> blockchain;

        public BlockChainManager(int interISPNumber, List<Block> blockList) {

                this.ISPNumber = interISPNumber;
                this.blockchain = blockList;
        }
```

Şekil 11. Blokzincir Yöneticisi

```
public class ProofOfAuthorityClique {

	private BlockChainManager validators[];
	private Block current;
	private Block previous;

	public ProofOfAuthorityClique(BlockChainManager validators[], Block current, Block previous) {
		this.validators = validators;
		this.current = current;
		this.previous = previous;
	}

	public boolean controlBlockValidation() {

		for (BlockChainManager blockChainManager : this.validators) {

			if (!blockChainManager.ValidatorAgent(this.current, this.previous)) {
				return false;
			}
		}

		return true;
	}
}
```

Şekil 12. Konsensüs Modülü

Blokzincirde tüm işlemler oluşturulduktan sonra denetleyiciler uçtan uca hizmet kalitesi gereksinimlerini de göz önünde bulundurarak bulmak istedikleri yolları oluşturmak için üst seviye bir ağ bağlantı yapısını belirlemektedir.

Belirlenen üst seviye ağ bağlantıları üzerinde hizmet kalitesi parametreleri kullanılarak uçtan uca veri iletimi isteklerinin tanımlanması gerekmektedir. Bu işlemler için iki modül oluşturduk. İlk olarak, isteklerimizin türünü ve hizmet kalitesi parametrelerindeki özellikleri de belirlemek amacıyla "Request" modülüni Şekil 13'de belirtildiği gibi tasarladık. "Request" nesnesi içerisinde sırasıyla, başlangıç düğümü ile hedef düğümleri arasında bulunması planlanılan uçtan uca yol için talep edilen bant genişliği, gecikme toleransı ve güvenilirlik metiklerini numerik değerler alacağını öngörerek tanımlamaları yapılmaktadır. Hizmet kalitesi destekli servis taleplerini oluşturmak için ayrı bir modül olan "RequestGenerator" nesnesini tanımlayarak denetleyiciler tarafından uçtan uca yol bulmanın yol haritasını oluşturmayı planlayarak çoklu servis taleplerimizi Şekil 14'da gösterildiği şekilde oluşturduk.

```
public class Request {

        private int bandwidth;
        private double delay;
        private double reliability;
        private int sourceASId;
        private int destinationASId;
        private int source;
        private int destination;
        private List<String> priority;
        private int price;

        public Request() {
                this.bandwidth = 0;
                this.source = 0;
                this.destination = 0;
                this.delay = 0.0;
                this.reliability = 0.0;
                this.sourceASId = 0;
                this.destinationASId = 0;
                this.priority = new ArrayList<String>();
                this.price = 0;
        }
```

Şekil 13. Hizmet Kalitesi Destekli Servis Talebi Modülü

Tüm oluşturulan taleplerin cevaplandırılabilmesi amacıyla isteği talep eden kaynak düğüme sahip internet servis sağlayıcının denetleyici modülü blok zincir yapısındaki Şekil 11'de gösterilen blok yapılarının içerisinde bulunan tanımlanmış işlemleri kullanarak hizmet kalitesi destekli uçtan uca yol bulma işlemini başlatmaktadır.

```java
public class RequestGenerator {

	private int numOfAS;
	private int numOfVertices;
	private int numOfRequest;
	private int bandwidth;
	private int[][] edgeRouter;

	public RequestGenerator(int numOfAS, int numOfVertices, int numOfRequest, int minBw,
			int[][] edgeRouter) {
		this.numOfAS = numOfAS;
		this.numOfVertices = numOfVertices;
		this.numOfRequest = numOfRequest;
		this.edgeRouter = edgeRouter;
		Random rndValue = new Random();
		this.bandwidth = rndValue.nextInt(maxBw - minBw) + minBw;
	}

	public List<Request> generateRequests() {

		List<Request> requestList = new ArrayList<Request>();

		for (int i = 0; i < this.numOfRequest; i++) {

			Request request = new Request();
			Random rndValue = new Random();

			// Create as network list to hold AS IDs
			List<Integer> asList = new ArrayList<Integer>();
			for (int j = 0; j < this.numOfAS; j++) {
				asList.add(j);
			}

			int indexAS = rndValue.nextInt(asList.size());
			int pickSourceAS = asList.get(indexAS);
			asList.remove(indexAS);

			indexAS = rndValue.nextInt(asList.size());
			int pickDestAS = asList.get(indexAS);

			int sourceNode = getNodeIndex(pickSourceAS);
			int destNode = getNodeIndex(pickDestAS);

			request.setBandwidth(this.bandwidth);
			request.setDelay(0.0);
			request.setDestination(destNode);
			request.setDestinationASId(pickDestAS);
			request.setSource(sourceNode);
			request.setSourceASId(pickSourceAS);

			requestList.add(request);

		}

		return requestList;
	}
```

Şekil 14. Servis Talebi Oluşturma Modülü

Tüm bu sistemler oluşup çoklu alan ağları arasında gelen hizmet kalitesi destekli servis taleplerini karşılamak ve talepleri onaylama işlemini yapabilmek için uçtan uca yolu internet servis sağlayıcı düzeyinde pekiştirmeli öğrenme algoritması kullanarak yapmaktayız. Her bir talebi direk olarak işleme alarak blok zincir altyapısında uygun durumda bulunan çok sayıda işlem üzerinden yol bularak cevaplandırmak oluşturulan yazılım çerçevesinde yapılmaktadır.

Pekiştirmeli öğrenme, bir ağ içindeki iki düğüm arasındaki en verimli rotanın belirlenmesinde kullanılma potansiyeline olmaktadır. Bu çalışmada, Şekil 15'de belirtilen modül ile, bir ağdaki en kısa uçtan uca yolu bulmak için bir pekiştirmeli öğrenme algoritması olan Q-öğrenmenin bir uygulamasını sunulmaktadır. Bununla birlikte, pekiştirmeli öğrenme teknikleri ve uygulamaları karmaşık olabilir ve çeşitli problem alanlarına hitap etmek için titiz ayarlamalar gerektirmektedir. Belirtmekte olduğumuz kod bloğu, belirli kullanım durumuna ve ağ yapısına uyacak şekilde potansiyel optimizasyon ve özelleştirme gerektiren bir başlangıç çerçevesi olarak işlev görmektedir.

Modül içindeki her yöntem aşağıdaki şekilde çalışmaktadır:

Modülün başlangıç bölümü "ReinforcementLearning":

Belirlenen yöntem programın giriş noktası olarak hizmet eder. Ağ bir komşuluk sistemini blok zincir mimarisi ile oluşturulmuş bir bağlantılı yapı olarak başlatır ve düğüm sayısı, maksimum deneme, öğrenme oranı, indirim faktörü ve keşif olasılığı dahil olmak üzere birçok sabit tanımlanmaktadır. Q-değerleri matrisi, her bir öğe bir durumdan başka bir duruma eylemde bulunmayla ilişkili tahmin edilen kümülatif ödülü temsil edecek şekilde başlatılmaktadır.

Eylem Seçme (SelectAction) yöntemi:

Mevcut yaklaşım, keşif şansını göz önünde bulundurarak mevcut durumda üstlenilecek bir eylem rotası belirlemektedir. Rastgele üretilen bir sayı, belirlenen keşif olasılığından daha düşük bulunursa, rastgele bir eylem seçilerek keşif sürecini kolaylaştırmaktadır. Alternatif olarak, mevcut Q değerleri değerlendirilerek en yüksek Q değeri tarafından belirlenen en uygun eylem yolu seçilmektedir. Bu yöntem içerisinden rasgele eylem seçimi, belirli bir

durumdaki uygulanabilir eylemler koleksiyonundan stokastik bir eylemin seçilmesinden sorumlu olmaktadır.

Algoritma, potansiyel eylemlerin toplam sayısını hesaplar ve seçim için bir temel olarak sayımı kullanarak rastgele bir tane seçer. Tüm bu seçimler arasından en iyi eylemin belirlenmesi oldukça önemlidir. Bu sebeple, Q-değerleri matrisini kullanarak belirli bir durum içindeki en uygun eylem yolunu (yani en yüksek Q-değerine sahip eylemi) belirlemek gerekmektedir. Algoritma her bir potansiyel eylemi sistematik olarak değerlendirerek en büyük Q değerine sahip eylemi seçmektedir.

Maksimum Q Değeri (GetMaxQValue) yöntemi:

Yukarıda bahsedilen yaklaşım, tüm uygulanabilir eylemler arasında belirli bir durum için en büyük Q değerini hesaplamaktadır. Bu fonksiyon, mevcut durumda en yüksek Q-değeri tarafından belirlenen en uygun eylemi değerlendirerek geçiş yapılacak bir sonraki düğümü de belirlemektedir.

"ReinforcementLearning" modülü içerisinde, teşvikleri dikkate alarak ve en verimli yol hakkında bilgi edinerek Q değerlerini güncellemek için bölümler arasında yineleme yaptığı birincil Q-öğrenme döngüsüne girmeye devam etmektedir. Q-öğrenme döngüsü içerisinde, program belirlenen başlangıç (start) ve bitiş (end) düğümlerine ulaşılana kadar iterasyonlarla devam etmektedir. Mevcut duruma bağlı olarak uygun bir eylem belirlemek için "selectAction" yöntemi kullanılmaktadır. Sonraki durum, bir eylemin seçilmesiyle tespit edilmektedir. Ödül, sonraki durumun bitiş düğüme karşılık gelip gelmediği değerlendirilerek belirlenmektedir. Q-öğrenme algoritması mevcut durum ve eylem için Q-değerini güncellemek için kullanılarak yineleme bitiş düğümüme ulaşılana kadar devam etmektedir. Q-öğrenme döngüsünün tamamlanmasının ardından yazılım, elde edilen Q-değerlerini kullanarak belirlenen başlangıç düğümünden belirtilen bitiş düğümüne giden en kısa yolu yeniden oluşturmaya ve ardından çıktı vermeye devam etmektedir.

Oluşturulan çerçeve, verilen iki düğüm arasındaki en verimli yola ilişkin bilgi edinilmesini sağlayan bir Q-öğrenme algoritması uygulamak için blok zincirinde depolanan yol bilgilerini kullanmaktadır. Daha karmaşık durumlarda, çeşitli ağ temsillerini, durum uzaylarını ve pekiştirmeli öğrenme algoritmasının ayarlarını barındırmak için kodu

değiştirmek ve iyileştirmek gerekebilir.

```
public class ReinforcementLearning {

    private static final int NUM_NODES = NETWORK.length;
    private static final int MAX_EPISODES = 1000;
    private static final double LEARNING_RATE = 0.1;
    private static final double DISCOUNT_FACTOR = 0.9;
    private static final double EXPLORATION_PROBABILITY = 0.1;

    public ReinforcementLearning(int start, int end, int totalNumOfNodes) {

        // Initialize the Q-values
        double[][] qValues = new double[NUM_NODES][NUM_NODES];
        for (int i = 0; i < NUM_NODES; i++) {
            for (int j = 0; j < NUM_NODES; j++) {
                qValues[i][j] = 0.0;
            }
        }

        // Q-learning algorithm
        for (int episode = 0; episode < MAX_EPISODES; episode++) {
            int currentState = startNode;
            while (currentState != endNode) {
                int nextAction = selectAction(currentState, qValues);
                int nextState = nextAction;

                double reward = (nextState == endNode) ? 100 : -1; // Assign rewards

                double maxNextQValue = getMaxQValue(nextState, qValues);
                double updatedQValue = (1 - LEARNING_RATE) * qValues[currentState][nextAction]
                        + LEARNING_RATE * (reward + DISCOUNT_FACTOR * maxNextQValue);

                qValues[currentState][nextAction] = updatedQValue;

                currentState = nextState;
            }
        }

        // Finding the shortest path
        int currentNode = startNode;
        System.out.print("Shortest Path: " + currentNode);
        while (currentNode != endNode) {
            int nextNode = getNextNode(currentNode, qValues);
            System.out.print(" -> " + nextNode);
            currentNode = nextNode;
        }
    }
```

Şekil 15. Pekiştirmeli Öğrenme Tabanlı Uçtan Uca En Kısa Yol Bulma Modülü

5. TARTIŞMA

Bu çalışmada, şu ana kadar bir kavram kanıtı perspektifi sağlamış olsak da, bu heyecan verici alanda daha fazla iyileştirme ve ince ayar gerektiren birçok yön ve daha fazla çalışma olasılığı bulunmaktadır. Bu nedenle, bu bölümde, önerilen gelecek vaat eden trafik yönetimi çerçevesi pekiştirmeli öğrenme tabanlı blok zinciri yönlendirme hakkında aşağıdaki noktalar aracılığıyla daha fazla araştırma başlatmak amacıyla çeşitli kısıtlamalar, zorluklar ve gelecekteki potansiyel çalışma fikirleri hakkında ayrıntılı bir tartışma sunmayı amaçlamaktayız.

Blokzincir Defter Eskimesi: Pekiştirmeli öğrenme tabanlı blok zinciri yönlendirmedeki pathlet'leri bünyesinde barındıran işlemler, ağ dinamikleri ve trafik koşulları nedeniyle sürekli değişikliklere tabidir. Durumundaki herhangi bir güncellemeden sonra bir pathlet için bir işlem oluşturulur. Bir işlem oluşturulduktan sonra, blokzincir defterine yansıyana kadar bir dizi adımdan geçer (teklif düğümlerini bloğa gönderme, yeni bloğa dahil edilmeyi bekleme, yeni bloğu blokzincir ağına yayma vb.). İşlemdeki pathlet'in yansıyan durum değişikliği, blokzincir defterine yansıtılana kadar benzer verileri tutmayabilir, çünkü ağda karşılık gelen pathlet'in durumunda başka bir güncelleme olabilir. Bu nedenle, bazı işlemler blokzincirdeki gerçek güncel pathlet durumlarını yansıtmayabilir. Bu, geleneksel ağlardaki genel yönlendirme güncellemelerine benzemektedir.

Blokzincir Defter Boyutu: İşlemler, önerilen modelde ağlardaki verilerin iletim yolculuklarını yansıtır. Topolojik ve hizmet kalitesi ile ilgili verileri tutarlar. Ağlar boyut, topoloji, hizmetler, kullanıcılar vb. ile ilgili karmaşık yapılara sahip olduklarından, operasyonları sırasında sık sık güncellemeleri olabilir. Bu güncellemeler yeni işlemlere yansıtılır ve tüm katılımcı ağlara yayılır. Bu nedenle, depolama çok uzun vadede bir kısıtlama haline gelebilir. Bu durum, denetleyicilerin uğraştığı ağ yükünü de etkileyebilir.

Blok Süresi : Bu süre aynı zamanda blok aralığı olarak da bilinir. Bu önemli bir ölçüdür, çünkü tüm modelin daha yüksek işlem hacmi elde etmesine yardımcı olabilir ve ISS'lerden gelen pathlet durumlarıyla ilgili olarak blokzincir defterini güncel tutabilir. Öte yandan, bu aralık ağ denetleyicilerinde daha fazla hesaplama yapılmasına ve ağlarda daha fazla bağlantı kaynağı kullanılmasına neden olabilir. Bu nedenle, blok oluşturma süresi, ilgili

hesaplama yükü ve blokzincir defter eskimesi arasında hassas bir değiş tokuş vardır.

Mutabakat Protokolü Yükü: Mutabakat protokolleri, yazılım tanımlı ağ atmosferindeki ağ denetleyicileri üzerindeki ek yüke ilişkin olarak pekiştirmeli öğrenme tabanlı blokzinciri yönlendirme çerçevesinin genel performansında önemli bir rol oynar. Çeşitli değiş tokuşlara ilişkin fikir edinmek için çeşitli hafif ancak etkili ve verimli fikir birliği protokollerini keşfetmek, gelecekteki çalışma olarak üstlenmeyi planladığımız bir çalışmadır. Önerilen modelde blokzincir düğümleri olarak ISS denetleyicileri, ağla ilgili görevler ve mesajlarla zaten meşgul olmaktadır. Yeni bir blok oluşturmak için kriptografik sorunları çözmek gibi yoğun hesaplama gerektiren görevlerle yüklemek, ağla ilgili performanslarını sınırlayabilir ve bu nedenle hesaplama gücü gerektirebilir. Sonuç olarak, daha hafif ancak verimli bir mutabakat protokolü kullanmak, denetleyicilerin yükünü hafifletecektir.

Yukarıda belirtilen sınırlamaların ve zorlukların mekansal boyutu, ağ ve denetleyici ek yükü ile ilgili daha derinlemesine çalışmalarda fayda sağlayacaktır.

6. SONUÇ VE ÖNERİLER

Bu araştırma, geliştirilmiş hizmet kalitesi ile desteklenen yazılım tanımlı çoklu ağlarda Yönlendirme Mimarisi için kapsamlı bir yapı ortaya koymaktadır. Hizmet kalitesi performansının arttırılması, pekiştirmeli öğrenme ve blokzincir gibi yeni metodolojiler kullanılarak elde edilebilir. Çerçeve, Hizmet kalitesi Yönetim Modülü, pekiştirmeli öğrenme Modülü ve Blokzincir Modülü gibi birkaç temel modül ve uygulama bileşeninden oluşur. Yukarıda belirtilen bileşenler, hizmet kalitesi metriklerinin toplanması, politika uygulama önlemlerinin uygulanması, pekiştirmeli öğrenme tekniklerinin kullanılması, blokzinciri ağının oluşturulması ve hizmet kalitesi verilerinin korunması dahil ancak bunlarla sınırlı olmamak üzere çeşitli görevleri yerine getirmekten sorumludur. Önerilen çerçeve, hizmet kalitesi performansını artırmak ve yazılım tanımlı çoklu ağlar akıllı yönlendirme kararlarını etkinleştirmek için uygun bir yaklaşım sunar. Entegre modüllerin ve işlevlerin kullanımı, gelişmiş kaynak kullanımı, akıllı yönlendirme kararları ve güvenli veri işleme gibi potansiyel avantajlarla ilişkilendirilir.

KAYNAKLAR

Akella, A. V, & Xiong, K. (2014). Quality of service (QoS)-guaranteed network resource allocation via software defined networking (SDN). *2014 IEEE 12th International Conference on Dependable, Autonomic and Secure Computing*, 7-13.

Chen, Z., & Wang, X. (2020). Decentralized computation offloading for multi-user mobile edge computing: A deep reinforcement learning approach. *EURASIP Journal on Wireless Communications and Networking*, *2020*(1), 1-21.

Chowdhury, N. M. M. K., & Boutaba, R. (2009). Network virtualization: state of the art and research challenges. *IEEE Communications magazine*, *47*(7), 20-26.

Farhady, H., Lee, H., & Nakao, A. (2015). Software-defined networking: A survey. *Computer Networks*, *81*, 79-95.

Guler, E., Karakus, M., & Uludag, S. (2023). Blockchain-enhanced cross-ISP spectrum assignment framework in SDONs: SpectrumChain. *Computer Networks*, *223*, 109579.

Gupta, R., Tanwar, S., Kumar, N., & Tyagi, S. (2020). Blockchain-based security attack resilience schemes for autonomous vehicles in industry 4.0: A systematic review. *Computers & Electrical Engineering*, *86*, 106717.

Helebrandt, P., & Kotuliak, I. (2014). Novel SDN multi-domain architecture. *2014 IEEE 12th IEEE International Conference on Emerging eLearning Technologies and Applications (ICETA)*, 139-143.

Iqbal, W., Abbas, H., Deng, P., Wan, J., Rauf, B., Abbas, Y., & Rashid, I. (2023). ALAM: Anonymous lightweight authentication mechanism for SDN enabled smart homes. *Journal of Network and Computer Applications*, 103672.

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, *4*, 237-285.

Karakus, M., Guler, E., & Uludag, S. (2021). Qoschain: Provisioning inter-as qos in software-defined networks with blockchain. *IEEE Transactions on Network and*

Service Management, *18*(2), 1706-1717.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., & Turner, J. (2008). OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM computer communication review*, *38*(2), 69-74.

Nofer, M., Gomber, P., Hinz, O., & Schiereck, D. (2017). Blockchain. *Business & Information Systems Engineering*, *59*, 183-187.

Son, J., Dastjerdi, A. V., Calheiros, R. N., & Buyya, R. (2017). SLA-aware and energy-efficient dynamic overbooking in SDN-based cloud data centers. *IEEE Transactions on Sustainable Computing*, *2*(2), 76-89.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Tomovic, S., Prasad, N., & Radusinovic, I. (2014). SDN control framework for QoS provisioning. *2014 22nd telecommunications forum Telfor (TELFOR)*, 111-114.

Tsai, P.-W., Tsai, C.-W., Hsu, C.-W., & Yang, C.-S. (2018). Network monitoring in software-defined networking: A review. *IEEE Systems Journal*, *12*(4), 3958-3969.

Wang, N., Ho, K. H., Pavlou, G., & Howarth, M. (2008). An overview of routing optimization for internet traffic engineering. *IEEE Communications Surveys & Tutorials*, *10*(1), 36-56.

Yeganeh, S. H., Tootoonchian, A., & Ganjali, Y. (2013). On scalability of software-defined networking. *IEEE Communications Magazine*, *51*(2), 136-141.

Zhang, Y., Cui, L., Wang, W., & Zhang, Y. (2018). A survey on software defined networking with multiple controllers. *Journal of Network and Computer Applications*, *103*, 101-118.