



İnternet Servis Sağlayıcıları Arasında Hizmet Kalitesi Temelli Akıllı Yönlendirme Modeli: Yazılım Tanımlı Ağ, Blokzincir Ve Makine Öğrenmesi Teknolojilerinin Kesişimi

Program Kodu:3501

Proje No:120E448

Proje Yürütücüsü:

Dr. Öğr. Üyesi MURAT KARAKUŞ

Araştırmacı(lar)

EVRİM GÜLER

Bursiyer(ler)

ALİ BERKAY GÖRGÜLÜ

NİSAN 2024

ANKARA

ÖNSÖZ

Bu proje ile, Yazılım Tanımlı Ağ (YTA), Blokzincir ve Makine Öğrenmesi teknolojilerin sağladığı yeteneklerden yararlanarak kullanıcılarından gelen hizmet kalitesi odaklı bağlantı isteklerinin kurulum süresi ve ağların mesaj alış-verışı azaltılırken, ağ ölçeklenebilirliğini artıracak İnternet Servis Sağlayıcıları (İSS) arası akıllı ve özgün bir yönlendirme modeli geliştirilmiştir. Bu amaç doğrultusunda, proje aşağıdaki hedefleri gerçekleştirmiştir:

- Rekabet içerisinde olan İSS'ler için blokzincir, Yazılım Tanımlı Ağlar ve Pekiştirmeli Öğrenme (Reinforcement Learning) kullanarak yönlendirme algoritması oluşturulması,
- Blokzincir teknolojisinin âdemi merkeziyetçilik özelliklerinden yararlanarak İSS'ler için merkeziyetsiz yönlendirme yapısı kurulması,
- Hizmet bağlantı istekleri için hizmet kalitesi sinyalizasyon mesajlarının sayısını ve akış kurma süresini azaltarak ve servis sağlanan istek sayısını artırarak ağ ölçeklenebilirliğinin artırılması,
- İSS'lerin yönlendirme hizmeti kaynaklı gizlilik kaygılarını azaltılması,
- Makine öğrenmesi destekli blokzincir teknolojisi için potansiyel uygulama ve araştırma alanı oluşturulması

Proje, konusu itibarı ile hizmet kalitesi, yönlendirme, makine öğrenmesi, blokzincir ve YTA gibi farklı araştırma disiplinlerini kapsamaktadır. Önerilen modelin performansının değerlendirilmesi için öncelikle simülasyon/emülasyon bazlı bir test ortamları kullanılmıştır. Sonrasında ise, fiziksel bir sınama ortamında geliştirilen modeller test edilmiştir.

Bu proje; Türkiye Bilimsel ve Teknolojik Araştırma Kurumu (TÜBİTAK) tarafından 120E448 numara ile “İnternet Servis Sağlayıcıları Arasında Hizmet Kalitesi Temelli Akıllı Yönlendirme Modeli: Yazılım Tanımlı Ağ, Blokzincir ve Makine Öğrenmesi Teknolojilerinin Kesişimi” projesi adı altında 15/02/2021 ve 15/02/2024 tarihleri arasında desteklenmiştir. Bu bağlamda, yaptıkları tavsiyelerden ötürü proje süresince çalışmalarımızı izleyen anonim hakemlere ve proje süresince bürokratik konularda bize destek veren bütün TÜBİTAK personeline kendim ve proje ekibim adına teşekkür ederim.

Proje Yürütücüsü
Dr. Öğr. Üyesi Murat KARAKUŞ
Yazılım Mühendisliği Bölümü
Ankara Üniversitesi

İÇİNDEKİLER

GİRİŞ.....	1
Motivasyon.....	3
YTA ve Blokzincir Teknolojileri.....	5
Proje Katkıları	8
Sonuç Raporu Organizasyonu.....	9
LITERATÜR ÖZETİ.....	9
YÖNTEM	14
QoSChain (QC): Yazılım Tanımlı Ağlarda Blokzincir ile İnternet Servis Sağlayıcıları (İSS) Arası QoS Sağlama	14
Yazılım Tanımlı Ağlarda Trafik Yönetimi İçin Blokzincir Destekli Kaynak-Yönlendirmenin (Source Routing – SR) Uygulanması.....	26
Çok Alanlı YTA'da Blokzinciri Tabanlı Yönlendirme Üzerinden Yol Seçim Stratejilerinin Etki Analizi	31
SmartContractChain (SC ²): SDN ve Blokzincir ile Çoklu-İSS QoS Trafik Yönetim Çerçevesi	32
Yazılım Tanımlı Optik Ağlar (YTOA)'da Blokzincir ile Geliştirilmiş Çoklu-İSS Spektrum Atama Çerçevesi: SpectrumChain.....	41
Blokzincir Destekli Çok Aktörlü Pekiştirmeli Öğrenme Algoritması ile QoS Merkezli Uçtan Uca Yol Hesaplanması	52
BULGULAR	60
QoSChain (QC): Yazılım Tanımlı Ağlarda Blokzincir ile Otonom Sistemler Arası QoS Sağlama	60
Yazılım Tanımlı Ağlarda Trafik Yönetimi İçin Blokzincir Destekli Kaynak-Yönlendirmenin (Source Routing – SR) Uygulanması	68
Çok Alanlı YTA'da Blokzinciri Tabanlı Yönlendirme Üzerinden Yol Seçim Stratejilerinin Etki Analizi	73
SmartContractChain (SC ²): SDN ve Blokzincir ile Çoklu-İSS QoS Trafik Yönetim Çerçevesi	80
Yazılım Tanımlı Optik Ağlar (YTOA)'da Blokzincir ile Geliştirilmiş Çoklu-İSS Spektrum Atama Çerçevesi: SpectrumChain.....	85
TARTIŞMA.....	91
Blokzincir Defterinin Güncellenme Süresi.....	91
Blokzincir Defter Boyutu.....	92
Blok Süresi.....	92
Uzlaşma Protokolü Yükü.....	92
SONUÇ	93

TABLO LİSTELERİ

Tablo 1: İlgili Bazı Çalışmaların Özeti.....	11
Tablo 2: YTA, Blokzincir ve akıllı sözleşmeleri kullanan çalışmaların ağ türleri, blokzincir modeli ve akıllı sözleşmelerin kullanımına ilişkin özeti	12
Tablo 3: Şekil 3'da İSS2'de R5 ve R7 arasındaki parça yolları için İSS2	21
Tablo 4: QoSChain Çerçevesinin Tasarım prensipleri	22
Tablo 5: Şekil 21'de gösterildiği gibi R5 ve R7 sınır aygıtları arasındaki parça yolları için İSS2 denetleyicisi tarafından oluşturulan işlemler.....	47
Tablo 6: Notasyon Tablosu	61
Tablo 7: Performans değerlendirmesinde kullanılan kısaltmalar ve anlamları	85
Tablo 8: HWS, BWS ve NWS senaryolarına göre SC, HRA ve DRA'da İletilen ve İşlenen Mesajların (MEP) sayısı	88

ŞEKİL LİSTELERİ

Şekil 1: Ana düzlemleri ve ara yüzleri ile SDN mimarisi	6
Şekil 2: Blokzincir ve blok veri yapılarına genel bakış	7
Şekil 3: 4 YTA İSS'den oluşan temsili bir ağ modeli	15
Şekil 4: Şekil 3'deki İSS'lerden soyutlanmış bir katman topolojisi	15
Şekil 5: Blokzincir ile uyumlu bir SDN denetleyicisinin genel bakışı: Yeşil renkli bloklar içinde beyaz metinle gösterilen yeni denetleyici modülleri ve ağ uygulamalarını içerir.	16
Şekil 6: İSS2 ağında R5 ve R7 ağ cihazları arasındaki yollar	17
Şekil 7: QoSChain çerçevesindeki mesaj veri yapıları	18
Şekil 8: QC çerçevesinde parça yolların blokzincir işlemi veri yapısı	21
Şekil 9: Clique'de birincil (yeşil) ve ikincil (sarı) blokzincir düğümlerinin seçim süreci	24
Şekil 10: QoSChain çerçevesinde uçtan uca yol bulma süreci	25
Şekil 11: SoRBlock da 4 SDN alanı için blokzincir destekli alanlararası yönlendirme ve Kaynak-Yönlendirme tabanlı alan-içi yönlendirme düzenleri	26
Şekil 12: PIDTTL tabanlı Kaynak-Yönlendirme uygulamasında paket başlıklar	29
Şekil 13: Uçtan uca yolun giriş düğümlerinin eklenen akış girdisi veri yapısı	30
Şekil 14: SC ² çerçevesinde denetleyicilerin, özelleştirilmiş akıllı sözleşme yöneticisi, blokzincir yöneticisi ve uygulamaların bir temsili	33
Şekil 15: SmartContractChain (SC ²) çerçevesinde kullanılan blokzincir, blok ve blokzincir işleminin (parça yollar için) veri yapıları	34
Şekil 16: İSS'lerdeki blokzincir destekli SDN denetleyicilerinin Akıllı Sözleşme Yöneticisi (Smart Contract Manager - SCM) modülünün yapısı	37
Şekil 17: SC ² Durum Geçiş Diyagramı	39
Şekil 18: Bir servis talebi senaryosu ele alınırken SC ² nin işleyişinde yer alan süreçler ve aktörlerin çalışma modeli	40
Şekil 19: Bir akıllı sözleşmenin sırasıyla Reservations ve Collaborations değişkenlerinde tutulan örnek bir rezervasyon ve ortaklar listesi	41
Şekil 20: YTOA İSS ağ modelinin ve onun soyutlanmış yer paylaşımı ağının bir örneği.....	42
Şekil 21: İSS2 ağında örnek bağlantı spektrumu kullanılabilirliği olarak R5 ve R7 ağ cihazları arasındaki optik parça yollar (R5-R7, R5-R8-R7 ve R5-R6-R8-R7).	43
Şekil 22: SpectrumChain'de ki transaction (işlem) veri yapısı	46
Şekil 23: Şekil 21'de gösterildiği gibi İSS2'nin R5 ve R7 sınır cihazları arasındaki 4 farklı durumda parça yollar üzerinden esnek optik iletişim kaynak tahsisinin bir tasviri: Spektrum tahsisinden önce, HWS senaryosu, BWS senaryosu ve NWS senaryosu	49
Şekil 24: Bir hizmet isteği için İSS'lerarası yönlendirmenin iş akışı	50
Şekil 25: SpectrumChain'de uçtan uca optik yol bulma işleminin sıra diyagramı	51
Şekil 26: QoSChain (QC), DRA ve HRA'nın Akış Kurulum Süreleri (FST)	64
Şekil 27: QoSChain (QC), DRA ve HRA'nın Gönderilen ve İşlenen Mesajları (MEP)	66
Şekil 28: QoSChain (QC), DRA ve HRA'nın Hizmet Verilen Talepleri (RS)	68
Şekil 29: SoRBlock ve HRA çerçevelerinin PST performansları	70
Şekil 30: SoRBlock ve HRA çerçevelerindeki CMP	72
Şekil 31: Bir Hizmet Talebinde artan bant genişliği talebi (NSFNET İSS Ağı)	74
Şekil 32: Ağ başına anahtar(lar) sayısının artırılması (NSFNET İSS Ağı)	75
Şekil 33: Ağ başına Ağ başına artan anahtar(lar) sayısı (USNET İSS Ağı)	77
Şekil 34: Bir Hizmet Talebinde artan bant genişliği talebi (USNET İSS Ağı)	79
Şekil 35: SC ² , QC, HRA ve DRA'nın Akış Kurulum Süresi (FST)	81
Şekil 36: SC ² , QC, HRA ve DRA'nın İletilen ve İşlenen Mesaj (MEP) sayıları	83
Şekil 37: SC, HRA, ve DRA için Akış Kurulum Süresi	86
Şekil 38: SC, HRA ve DRA için Hizmet Verilen Talepler (RS)	89
Şekil 39: HWS, BWS ve NWS senaryolarına göre Hizmetlerin Kabul Oranı	91

ALGORİTMA LİSTELERİ

Algoritma 1: Dalga-boyu dönüştürücü-etkin altyapıda bitişik spektrumları bulma	47
Algoritma 2: Dalga-boyu dönüştürücüsüz altyapıda bitişik ve sürekli spektrumları bulma	48
Algoritma 3: QoS bazlı yönlendirme için Pekiştirmeli Öğrenme algoritması.....	56
Algoritma 4: Eylem (action) seçim fonksiyonu.....	57
Algoritma 5: Rastgele eylem (action) seçim fonksiyonu	57
Algoritma 6: En iyi eylem (action) seçim fonksiyonu	58
Algoritma 7: Pathlet bilgileri kullanılarak ödülün hesaplanması.....	58
Algoritma 8: Maksimum Q değerinin hesaplanması	59

ÖZET

Bu proje, internet hizmet sağlayıcıları arasında daha akıllı ve verimli bir yönlendirme modeli geliştirmek üzere Yazılım Tanımlı Ağ (YTA), Blokzincir ve Makine Öğrenmesi teknolojilerini birbirlerine entegre etmiştir. Temel hedef, kullanıcıların hizmet kalitesi odaklı bağlantı isteklerini daha hızlı ve daha verimli bir şekilde işleyerek ağların ölçeklenebilirliğini artırmaktır. Bu doğrultuda, proje bir dizi önemli çıktı üretmiştir. Öncelikle, Blokzincir tabanlı Servis/Hizmet Kalitesi (Quality of Service – QoS) odaklı İnternet Servis Sağlayıcı (İSS)'lar arası yönlendirme çerçevesi olan *QoSChain* (QC) geliştirilmiştir. QC, hizmet düzeyi anlaşmaları, güvenlik ve gizlilik riskleri gibi konularda tarafları bir araya getirir ve yeni yönlendirme yolları oluşturur. QC, ağlar arası iletişimde daha verimli ve güvenli bir yapı sağlayarak, internet hizmet sağlayıcılarının iş birliği yapmasını kolaylaştırır. Sonrasında, iç ağ trafiği yönetimi için Kaynak-Yönlendirme şeması olan *SoRBlock*, QC çerçevesine entegre edilmiştir. Bu entegrasyon, ağ içi iletişim etkinliğini artırırken, dış ağ iletişimini de blokzincir teknolojisiyle güvence altına almaktadır. Bu sayede, ağlar arasındaki trafiği daha iyi yönetmek ve kaynakları daha verimli kullanmak mümkün hale gelmektedir. Projede ayrıca, çeşitli yol seçim stratejilerinin QC'in performansı üzerindeki etkisi de detaylı bir şekilde incelenmiştir. Bu analiz, internet trafiğinin farklı özelliklerine uygun olarak, en uygun yönlendirme stratejisinin belirlenmesine yardımcı olur. Bununla birlikte, *SmartContractChain* (SC²) adı verilen akıllı ve dağıtılmış bir koordinasyon çerçevesi oluşturularak, İSS ağları arası QoS özellikli trafik yönetimi geliştirilmiştir. SC², ağların otomatik olarak uyum sağlamaşını ve hizmet kalitesini iyileştirmesini sağlayan akıllı sözleşmelerin kullanımını içerir. *SpectrumChain* adı verilen bir başka çıktı ise Yazılım Tanımlı Optik Ağlarda QoS odaklı bir İSS'ler arası spektrum atama ve yönlendirme çerçevesidir. Bu çerçeve, farklı frekans bantlarına otomatik olarak kaynak tahsis yaparak, ağlar arası iletişim kalitesini artırır. Son olarak, Pekiştirmeli Öğrenme (RL) algoritması kullanılarak, QoS motivasyonlu ağlar arası trafik yönlendirilmesi gerçekleştirilmiştir. Bu yönlendirme modeli, ağların dinamik koşullarına uyum sağlayarak, hizmet kalitesini sürekli olarak iyileştirir. Bu çeşitli çıktılar, internet hizmet sağlayıcılarının ağ yönetimini daha bütünsel, akıllı ve etkili bir şekilde yapmalarına yardımcı olarak, kullanıcı deneyimini ve ağ performansını artıracaktır.

Anahtar Kelimeler: Yazılım Tanımlı Ağ (YTA), Blokzincir, Makine Öğrenmesi, Yönlendirme, Hizmet Kalitesi, Pekiştirmeli Öğrenme



ABSTRACT

This project has integrated Software Defined Networking (SDN), Blockchain, and Machine Learning technologies to develop a smarter and more efficient routing model among Internet Service Providers (ISPs). The primary aim is to process user-requested connections focused on service quality faster and more efficiently, thereby enhancing network scalability. In this regard, the project has produced a series of significant outputs. Firstly, a Blockchain-based Quality of Service (QoS) centric inter-ISP routing framework named *QoSChain* (QC) has been developed. QC brings together stakeholders with conflicting interests such as Service Level Agreements (SLAs), security, and privacy risks, and establishes new routing paths. By providing a more efficient and secure structure for inter-network communication, QC facilitates collaboration among ISPs. Additionally, a Source-Routing scheme called *SoRBBlock*, which manages intra-network traffic, has been integrated into the QC framework. This integration enhances intra-network communication efficiency while ensuring external network communication through blockchain technology. This enables better traffic management between networks and more efficient resource utilization. Furthermore, the project extensively examined the impact of various path selection strategies on QC's performance. This analysis assists in determining the most suitable routing strategy based on the different characteristics of internet traffic. Moreover, an intelligent and distributed coordination framework called *SmartContractChain* (*SC²*) has been developed to enhance inter-ISP traffic management with QoS capabilities. *SC²* includes the use of smart contracts, enabling networks to automatically adapt and improve service quality. Another output, *SpectrumChain*, is a QoS-enabled inter-ISP spectrum allocation and routing framework in Software Defined Optical Networks. This framework enhances inter-network communication quality by automatically allocating resources to different frequency bands. Lastly, using Reinforcement Learning (RL) algorithms, QoS-motivated inter-domain traffic routing has been achieved. This routing model continuously improves service quality by adapting to dynamic network conditions. These various outputs can assist internet service providers in managing networks more holistically, intelligently and effectively, thereby enhancing user experience and network performance.

Keywords: Software Defined Network (SDN), Blockchain, Machine Learning, Routing, Quality of Service, Reinforcement Learning

1. GİRİŞ

Son yıllarda küresel İnternet trafiğinde yaşanan hızlı artış, yüksek bant genişliği, düşük gecikme gibi spesifik gereksinimleri olan video konferans, Seç-İzle (Video-on-Demand/VoD), IP-überinden-Ses (Voice-over-IP/VoIP), çevrimiçi oyun gibi çeşitli uygulamalardan kaynaklanmaktadır. Bu operasyonel gereksinimlerin sağlanması bir şemsiye terim olarak Servis/Hizmet Kalitesi (Quality of Service/QoS) olarak adlandırılır. QoS'in birincil amacı, ağ trafiği için bant genişliği, gecikme, titreşim gibi belirli özelliklere ilişkin önceliklendirme yapmaktadır.

Yazılım Tanımlı Ağ - YTA (Software Defined Networks - SDNs) mimarisi kullanan İnternet Servis Sağlayıcılarında (İSS) İSS-içi (intra-İSS) QoS yönlendirme problemi önemli olsa da İSS'ler arası seviyede QoS'i desteklemek tartışmasız daha önemlidir. Ancak sıkı iş gizliliği kısıtlamaları altında birden fazla İSS'ye (Labovitz vd., 2010) yayılan İnternet trafiğinin hacmi ve hızı nedeniyle, bu ağ yöneticileri için zorlayıcı bir süreç olmaktadır. Günümüzde QoS tabanlı İSS'ler arası yönlendirmenin sağlanması sırasında yaşanan zorlukların başlıca nedenleri aşağıda özetlenmiştir:

- **Çoklu İSS Atlama Sorunu:** Ağlar, Hizmet Seviyesi Anlaşmaları (Service Level Agreements/SLA) ve doğrudan bağlantılar aracılığıyla QoS özellikli yönlendirmeyi yalnızca en yakın komşu ağın etki alanları üzerinden garanti edebilir.
- **Bilgi Paylaşımı Sorunları:** İSS'ler arası yönlendirme için çevik ve ölçülebilir bir iletişim çerçevesi, İSS'ler arasında belirli bir düzeyde bilgi paylaşımına ihtiyaç duyar. Ancak bu kadar detaylı bir bilginin paylaşılması aşağıda özetlenen bazı sorunları da beraberinde getirmektedir:
 - *Bilgi Paylaşımında İsteksizlik:* Ağ yöneticileri, hassas bilgilerin gizliliğini korumak için ağlarıyla ilgili ayrıntıları (bant genişliği, gecikme vb.) başkalarıyla paylaşmaktan çekinmektedir.
 - *QoS Sinyalleşme Yükü:* QoS tabanlı yönlendirme için İSS'ler arası iletişim hem ağ kaynaklı hem de transit trafik akışları (flows) için akış istek (flow request) sıklığı göz önüne alındığında büyük bir mesaj alışveriş yüküne (message overhead) neden olur.

Yukarıda aktarıldığı üzere, YTA da İSS'ler arası uçtan uca (End-to-End/E2E) ağ trafiği için QoS gereksinimlerinin tek tip ve tutarlı bir şekilde uygulanması, farklı ağlar arasında kolay değildir. Bununla birlikte, başarılı bir uçtan uca QoS sağlama için İSS'ler arası iletişim zorunludur. Bu nedenle, QoS tabanlı İSS'ler arası yönlendirmenin yukarıda belirtilen sorunlarını

hafifletmek için çevik, güvenilir, güvenli ve uyarlanabilir bir koordinasyon mekanizması gereklidir.

YTA'larda ki merkezi işlemler ve yönetim, son zamanlarda akıllı, esnek ve politika odaklı hizmet talepleri ile ağlardaki her akışa özel yönlendirme durumu talepleri nedeniyle ölçeklenebilirlik zorluklarına yol açmaktadır (Kreutz vd., 2015). Burada yaşanan bir zorluk; WAN'ların akış rotası boyunca, her veri düzlemi (data plane) cihazına akış kurallarını dağıtmak için denetleyicinin gereksiniminden kaynaklanan uzun yayılma gecikmeleriyle (propagation delay) karşılaşmasıdır. Uzun yayılma gecikmeleri, coğrafi olarak yayılmış çok sayıda anahtarı olan büyük ölçekli YTA ağlarında uzun yol kurulum sürelerine neden olmaktadır. Rota kurulum gecikmesi ağıda ki akışları etkileyebilir. Ancak, gecikme duyarlı akışların hizmet kalitesi (Karakus & Durresi, 2017b), bu uzun yol kurulum sürelerinden dolayı önemli ölçüde bozulabilir. Başka bir ölçeklenebilirlik zorluğu da veri düzlemi cihazları ve YTA denetleyicileri tarafından işlenen mesaj yüküdür. YTA anahtarları (switch), sınırlı paket işleme yetenekleri nedeniyle OpenFlow mesajlarını işlemekte zorluk yaşamayıpabilirler. YTA cihazları, akış tablolarında belirli miktarda akış kuralı tutabilirler ve YTA'da ki "esleştirme-ve-eylem (match-and-action)" kavramı, akışların nasıl kontrol edileceğini etkileyebilir. YTA denetleyicisi olarak görev yapan bilgisayarlar da sınırlı bir işlemci ve depolama kapasitesine sahiptirler. Ağ genişledikçe denetleyici mesaj yükü artar ve bu da bir ağ hesaplama darboğazına neden olabilir. Son olarak, YTA anahtarlarının akış kurallarını (flow rules) tutmak için sınırlı depolama kapasiteleri vardır. YTA cihazları genellikle TCAM tabanlı belleklerde akış tablolarını ve kurallarını saklarlar (Ghiasian, 2020). Ancak, TCAM bellekleri pahalı, güç tüketen ve sınırlı depolama kaynaklarına sahiptir. YTA mimarisinin bahsedilen ölçeklenebilirlik zorluklarıyla başa çıkmak için, kontrol katmanında ele alınan ağ olaylarının sayısını azaltmak ve yalnızca gereken ağ olaylarını işlemek gereklidir (Karakus & Durresi, 2017a).

Düger taraftan, Dünya çapında oluşturulan, kopyalanan ve tüketilen veri hacmi 2020 yılında 64,2 zettabayta ulaşmıştır ve 2025 yılına kadar üç katına çıkararak 181 zettabayta ulaşması beklenmektedir (Holst, 2021). Optik ağlar, yukarıda bahsedilen veri trafiği tufanıyla başa çıkmak için muazzam iletim kapasitelerine sahiptir (Thyagaturu vd., 2016). Bununla birlikte, geleneksel optik ağlar, dalga-boyu kanallarında devre, patlama ve paket anahtarlama gibi spesifik optik (fotonik) iletim ve anahtarlama özellikleriyle, bu zorlayıcı trafik dinamikleri ve büyümesi altında sağlam trafik yönetimi yetenekleri açısından yetersiz kalmaktadır. Bu tür trafik yönetimi yetenekleri, güçlü ağ dinamikleri altında verimli ve uyarlanabilir spektrum yönetimi/atama çerçeveleri gerektirir. Son optik yenilikler, Elastik Optik Ağlar (EON'lar) olarak adlandırılan yeni bir umut verici optik ağ türünü mümkün kılmıştır. EON'ların sağladığı faydalalar arasında kaynak optimizasyonu, trafik mühendisliği, daha yüksek hizmet kullanılabilirliği, yeni

iş modeli fırsatları, arızalara karşı artan esneklik, yük dengeleme, isteğe bağlı provizyon vb. yer almaktadır (Gerstel vd., 2012). EON'nin bant genişliği toplama, segmentasyon ve veri hızlarının esnek değişimi (Chatterjee vd., 2018) ile ilgili benzersiz özellikleri, Yazılım Tanımlı Optik Ağ - YTOA (Software Defined Optical Network - SDON) aracılığıyla YTA paradigmının tamamlayıcı doğal özellikleriyle yakından bağlantılıdır ve bunlar tarafından etkinleştirilir (Bhaumik vd., 2014).

1.1 Motivasyon

YTA, kontrol ve veri düzlemlerinin ayrıldığı, ağ zekasının ve durumunun mantıksal olarak merkezileştirildiği ve ağ altyapısının uygulamalardan ayrıstiği bir mimaridir. YTA mimarisi ile akış başına yönlendirme, geleneksel mimarilere kıyasla daha ölçeklenebilir, daha basit ve daha az zaman alır hale gelir (Karakus & Durresi, 2017b). OpenFlow, ağ operatörlerinin veri düzlemindeki çeşitli izole akışları yöneten yönlendirme tabloları oluşturmak için denetleyici içinde çeşitli yönlendirme algoritmaları (tipik en kısa yol yerine) kullanmasını sağlar.

Bunun yanı sıra, cihazların kontrol ve yönlendirme işlevlerinin birbirinden ayrılması sayesinde akışların dinamik olarak yönlendirilmesi denetleyiciler tarafından mümkün hale gelmektedir. Bu özellikler hem akış başına hem de dinamik yönlendirme sayesinde ağ yöneticilerinin ağları için daha fazla QoS özellikle yönlendirme sistemleri geliştirebilmelerine olanak sağlamaktadır. Üstelik YTA, ağ yöneticilerinin ağ dinamiklerini izlemelerine ve akış kuralı başına, paket başına, bağlantı noktası başına, tablo başına, sıra başına ve sayaç başına gibi çok düşük seviyelere sahip sayaçlar aracılığıyla güncel ağ durumu istatistiklerini görüntüleyebilmelerini olanak tanır.

Diğer taraftan, son yıllarda popülerliği ve kullanım alanları hızla artan Blokzincir teknolojisi (Yaga vd., 2018), sahip olduğu veri güvenliği, merkeziyetsizlik, şeffaflık gibi özellikleri sayesinde ağ operatörlerine İSS'ler seviyesinde QoS tabanlı yönlendirme deneyimlerini geliştirmeleri için etkin özellikler sunma potansiyeline sahiptir. Blokzincir, uzlaşma (consensus) protokollerini aracılığıyla Blokzincir işlemlerini (transaction) yönetmek için dağıtık bir ortam sağlar. Blokzincir, dağıtık ortam ve merkeziyetsiz doğası sayesinde İSS'lerin YTA ağlarındaki Broker kavramı ve SDX projesi (Karakus & Durresi, 2017b) gibi merkezi yapıarda özel bilgilerini üçüncü taraflarla paylaşma zorunluluğundan kaynaklanan güvenlik ve gizlilik konusundaki tereddütlerini gidermede yardımcı olabilir. Ek olarak, İSS'ler arası yönlendirme için QoS'i herhangi bir dış varlığa bağımlı olmadan dağıtık bir şekilde sağlayabilir. Ayrıca Blokzincirin şeffaflığı sayesinde tüm katılımcılar ağ üzerindeki Blokzincir işlem

faaliyetlerine erişebilir, onaylayabilir ve görüntüleyebilir. Bu özellik, hizmet ve kaynak sağlayıcıların bir hizmet akışı için ucta bir yol oluştururken, QoS ile ilgili verileri okumak için Blokzincir işlemlerini izleyebilmesi ve tarayabilmesi için ortak çalışmaya dayalı ağ ortamlarında Blokzincir özellikle çerçevelere yardımcı olabilir. Bununla birlikte, Blokzincirin dışarıdan sızmaya karşı koruma özelliği, QoS ile ilgili verilerin bütünlüğünü sağlamak ve değiştirilemez blokzincir işlem defterleri kullanarak, büyük ölçekteki güvensiz ortamlarda çatışmalar olması durumunda İSS'ler arasındaki anlaşmazlıklar arasında uzlaşma (consensus) sağlamak için yararlı bir çözüm olabilir.

Son zamanlarda, anahtarlar üzerinde tutulan akış kurallarının sayısını azaltarak YTA'nın ölçeklenebilirliğini artırma stratejisi, yani denetleyici ve veri katmanı düğümleri arasındaki bilgi değişimini azaltma, potansiyel bir strateji olarak kabul edilmiştir (Ventre vd., 2021). Bununla ilgili olarak, Kaynak-Yönlendirme (Source Routing - SR), kaynaktan hedefe kadar olan tüm rotayı belirleyen bir yöntemdir. YTA ağlarının merkezi yönetimi, denetleyicinin ağını küresel topolojisini izlemesine olanak tanımaktadır. YTA tabanlı hedef yönlendirme gibi, kaynak-anahtar (source-switch) tabanlı rota kapsülleme, akış bilgisini paket başlığı içinde saklamayı amaçlamaktadır. Bu, yol (path) boyunca her anahtarın yapması gerekenin sadece doğru çıkış portlarını okuyup paketleri göndermek olduğu anlamına gelir. Böylece, trafik yönetimi sağlamak için, yol oluşturma sırasında denetleyiciden herhangi bir etkileşim gerektirmeden tüm akışlar tarafından sınırlı ve sonlu miktarda akış kuralı kullanılabilir. Önceki paragraflarda bahsedilen sorumlara (yani, yol kurulum zamanı gecikmesi ve artan denetleyici mesaj işleme) potansiyel bir çözüm, yönlendirme fonksiyonlarında yol bilgilerinin paket başlıklarına yerleştirilmesini yapan bir Kaynak-Yönlendirme yaklaşımının kullanılması olabilir (Abujoda vd., 2016). Bu, sınırlı sayıda akış-bağımsız yönlendirme kuralını ağ cihazlarına koymayı mümkün kılar. Bu durum da yönlendirme kaynaklı iş yükünü önemli ölçüde azaltır ve denetleyicilerin veri katmanı cihazlarına akış girişlerini eklemek için yükünü azaltarak kontrol düzlemi ölçeklenebilirliğini artırır.

Ağlar arası QoS sağlananın derin ölçüği ve karmaşıklığı, alanlar arası bir politika ve güvence orkestrasyonunun daha yüksek düzeyde ucta görünürlüğü sağlayacağı daha katı çalışma koşullarını gerektirmektedir. İş ihtiyaçlarının rasyonel, daha akıcı, verimli bir şekilde kontrol edilmesi ve doğrulanması, çoklu ağlar ve çoklu bulut ortamları arasında bağlayıcı ve güvenli bir politika otomasyon altyapısı aracılığıyla sağlanmalıdır. Bu konjonktürde, proje çalışmalarının bir parçası olarak, doğruluk, şeffaflık, hız ve güvenlik gibi Akıllı Kontrat özelliklerinden yararlanarak akıllı, hızlı ve otonom hizmet sunumunu mümkün kılacak yeni nesil QoS tabanlı bir trafik yönetim modeli için blokzincir akıllı kontratları projede geliştirilen Blokzincir özellikle YTA çerçevesine entegre edilmiştir. Blokzincir, birden fazla YTA

arasında alanlar arası politika otomasyonu ve hizmet seviyesi anlaşması güvence orkestrasyonları sağlamak için akıllı kontratlar ile entegre edilebilir. Blokzincir üzerindeki akıllı kontratlar, sözleşmeye dayalı anlaşmaların, uygulama için merkezi bir varlık veya şeffaf, geri döndürülemez ve izlenebilir işlemelere sahip yasal bir alt sistem olmadan karşılıklı olarak güvensiz varlıklar arasında yönetilmesine olanak tanır. Ayrıca blokzincir, bir hizmet talebi için İSS'ler arası ekosistemde bir uçtan uca yolu oluştururken akıllı kontratların verileri değişmez ve güvenli bir şekilde depolaması ve bu bilgileri QoS ile ilgili alanları içeren bloklara kaydetmesi için ideal bir ortam sağlar. Dahası, akıllı kontratların hız, otomasyon ve güven gibi özellikleri çok taraflı ve sofistik politikaları mümkün kılar.

1.2 YTA ve Blokzincir Teknolojileri

Her ne kadar YTA ve Blokzincir teknolojileri büyük avantajlara sahip olsa da her birinin yaygın olarak uygulanması; entegrasyon, birlikte çalışabilirlik, uyarlanabilirlik, hibridizasyon, çeşitlendirilmiş veri hacmi/türleri gibi zorlukları da içermektedir. Bu bölüm, YTA ve Blokzincir teknolojileri hakkında genel bir teknik bakış vermektedir.

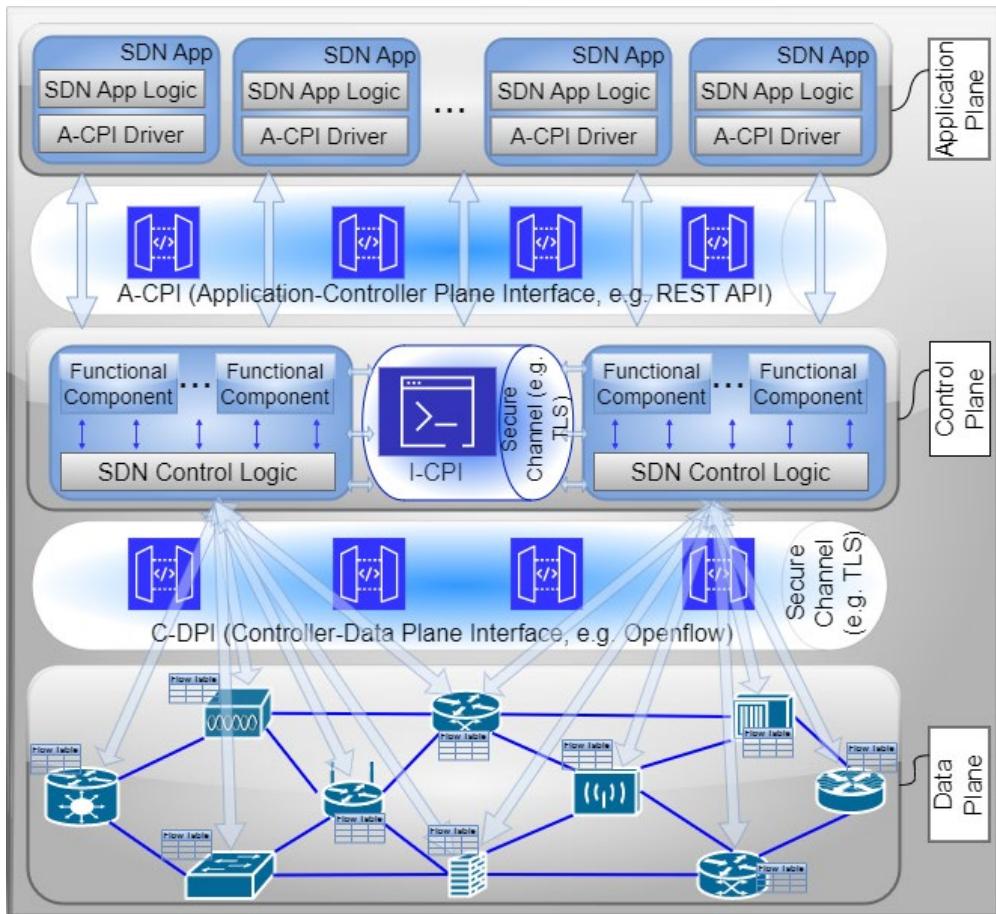
1.2.1 Yazılım tanımlı ağlar (YTA)

YTA mimarisi temel olarak üç bölümde tanımlanır: Şekil 1 de gösterildiği gibi veri düzlemi, kontrol düzlemi ve uygulama düzlemi. YTA, kontrol düzlemindeki ve veri düzlemindeki cihazlar arasında ki iletişim için protokollere sahiptir. OpenFlow protokolü (McKeown vd., 2008), kontrol ve veri düzlemleri arasındaki iletişim için kullanılan en yaygın standart protokoldür. YTA mimarisi ağ işlemlerini basitleştirir ve ağ yönetimi maliyetini azaltır. Bu avantajlar programlanabilir, merkezleştirilmiş, satıcıdan bağımsız ve uyarlanabilir özellikleriyle sağlanır.

VERİ DÜZLEMİ: Veri düzlemi, ağ cihazlarından (örn: yönlendiriciler, fiziksel/sanal anahtarlar ve erişim noktaları) oluşur. Paket iletme, veri katmanının temel fonksiyonudur. Bir YTA denetleyicisi, Denetleyici-Veri Düzlemi Arayüzlerini (C-DPI'ler) kullanarak bu cihazlar ile iletişim kurabilir ve yönetebilir. C-DPI mesajları TLS gibi güvenli bir kanal içerisinde gönderilir. OpenFlow protokolü, denetleyici(ler) ve veri katmanı ekipmanı arasında bağlantı için kullanılan en yaygın standart C-DPI'dir.

KONTROL DÜZLEMİ: YTA kontrol düzlemi, bir veya daha fazla yazılım tabanlı YTA denetleyici cihazını içerebilir. C-DPI'ya ek olarak, denetleyiciler arasında (Ara-Denetleyici Düzlem Arayüzü / I-CPI (Lin vd., 2015)) ve denetleyiciler ile uygulamalar arasında (Uygulama-Denetleyici Düzlem Arayüzü / A-CPI) etkileşime izin veren arayüzler vardır. I-CPI'lerden denetleyiciler arasında veri paylaşmak için yararlanılır. A-CPI, ağ uygulamaları ve

denetleyici(ler) arasında güvenlik, yönetim vb. amaçlarla iletişim kurmayı sağlar. Fonksiyonel bileşenler ve kontrol mantığı (control logic), bir denetleyicinin iki ana bileşenidir.



Şekil 1: Ana düzlemleri ve ara yüzleri ile SDN mimarisi

UYGULAMA DÜZLEMİ: YTA uygulama düzlemi, denetleyici(ler) ile etkileşime giren ve belirli bir ağ oluşturma görevini yerine getirmek için iç karar verme algoritmalarında ağın soyut bir görünümünü kullanan uygulamalardan oluşur. Bu uygulamalar, REST API gibi açık bir A-CPI ara yüzü üzerinden denetleyicilere bağlanır. Bir YTA uygulaması, bir YTA Uygulama Mantığı (App Logic) ve bir A-CPI Sürücüsünden (Driver) oluşur. Genellikle güvenlik, trafik mühendisliği, yük dengeleme vb. gibi gerçekleştirdikleri ağ hizmetlerine göre gruplandırılırlar.

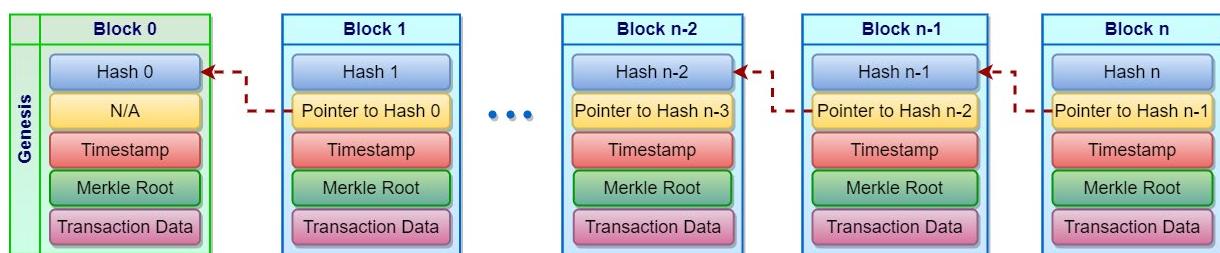
1.2.2 Blokzincir teknolojisi

Blokzincir, genellikle bir sistemdeki işlemlerin küçük boyutlu kayıtlarını depolayan, yalnızca veri ekleme yapan bir veri tabanıdır (Judmayer vd., 2018). Bu kayıtlar, güvenliği ve anonimliği korumak için ayrı ayrı şifrelenir ve blokzincire katılan tüm ağ düğümleri, her kaydın

geçerliliği konusunda anlaşarak, sonuçlar üzerinde konsensüs sağlanması ve ayrıca güven oluşturulmasını sağlar.

Bir blokzincir işlemi, belirli adımlar çerçevesinde dağıtık veri tabanına eklenir. Bunlar: (i) işlemi yürüten kullanıcı, işlemi imzalamak için özel bir anahtar kullanarak bunu eşlerine (peer) yayırlar; (ii) eşler gelen işlemi doğrular ve ardından tüm ağa yayılacak (broadcast) şekilde ileter; (iii) (genellikle) üzerinde anlaşılan bir zaman aralığından sonra düğümler, yeni bir blok oluşturmak/önermek ve bloğu doğrulamak için bir fikir birliği algoritması içeren bir süreç dahil olur; (iv) blok mevcut blokzincire eklenir. Düğümler, madencilik ve fikir birliği sürecindeki faaliyetler için bir ödül alabilir ve blok üreten düğümlerinin seçimi, ağıda kullanılan fikir birliği mekanizmasına bağlı olabilir.

Blokzincirinin eşler arası doğası, güvenilir bir üçüncü tarafa olan ihtiyacı ortadan kaldırın ve dolayısıyla YTA da dahil merkezi sistemlerin tek hata noktası (Single Point of Failure) sorununa bir çözüm sağlayan önemli bir özelliktir. Birden fazla eşit katılımcının olduğu sistemlerde, güven oluşturmak, tüm katılımcıların bazı çözümler üzerinde karşılıklı olarak anlaşabilmesini sağlamaya yönelik bilinen bir sorundur. Blokzincir, güveni sağlamak ve dış otorite ihtiyacını azaltmak için fikir birliği algoritmaları kullanır. Bu algoritmalar, cihazların hesaplama kaynaklarını (örn: CPU) tüketmektektir. Dolayısıyla bu süreçlere katılımı teşvik etmek için ödül mekanizmaları kullanılabilir. Bunun yanı sıra ceza olarak, kötü niyetli davranışları engellemek için bir düğümün herhangi bir prosedürü ihlal etmesi durumunda ödemesi gereken bir maliyet olarak da kullanılabilmektedir.



Şekil 2: Blokzincir ve blok veri yapılarına genel bakış

Blokzincir yapısının genel bir temsili Şekil 2 de gösterilmektedir. Her blok tipik olarak bir sıra numarası, bir özet (hash), önceki bloğun özeti, bir zaman damgası ve bir dizi işlem verisi içerir. Özet işaretçi verileri, bir bloğun zincirdeki konumunu sağlar ve yerel özet, genellikle blokta depolanan işlemlerde yer alan bilgileri doğrulamak için kullanılan bir Merkle ağıacı kökünün özet değeridir.

Çeşitli blokzincir modelleri mevcuttur. Orijinal versiyon, herkesin katılmasına, işlem yapmasına ve madencilik yapmasına olanak tanıyan açık bir platform olan halka açık (public) bir blokzincirdir. Erişim kısıtlaması bulunmadığı ve tüm katılımcılara işlemleri okuma ve yazma yetkisi verildiği için izinsiz (permissionless) olarak da bilinir. Özel (private) blokzincirleri (izinli - permissioned olarak da adlandırılır), özel paylaşımı ve veri alışverişine izin vermek için geliştirilmiştir. Madencilik, seçilen düğümler (örneğin bir kuruluş) tarafından kontrol ediliyor ve erişim belirli varlıklarla sınırlıdır. Hibrit bir yaklaşım, kısmen özel ve kısmen halka açık olan konsorsiyum blokzincirdir. Önceden belirlenmiş bir dizi düğüm, blokların ve fikir birliğinin doğrulanmasından sorumludur. Hangi düğümlerin ağa ait olabileceğine ve hangi düğümlerin madencilik yapabileceğine karar verirler.

1.3 Proje Katkıları

Bu proje; Yazılım Tanımlı Ağ (YTA), Blokzincir ve Makine Öğrenmesi teknolojilerin sağladığı yeteneklerden yararlanarak, kullanıcılarından gelen hizmet kalitesi odaklı bağlantı isteklerinin kurulum süresini ve ağların mesaj alış-verişini azaltarak ağ ölçeklenebilirliğini artıracak İnternet Servis Sağlayıcıları (İSS) arası akıllı ve özgün bir yönlendirme modeli geliştirmiştir. Bu amaç doğrultusunda, bu proje aşağıdaki çıktıları/katkıları gerçekleştirmiştir:

- Projede, Blokzincir teknolojisini YTA ağlarına entegre ederek yeni bir Blokzincir-tabanlı QoS odaklı İSS'ler arası yönlendirme çerçevesi olan *QoSChain* (QC) geliştirilmiştir. *QoSChain*'in özellikleri şu şekilde özetlenebilir: SLA'lar, güvenlik ve gizlilik riskleri gibi konularda çıkar çatışması olan tarafları bir araya getiren bir koordinasyon çerçevesi sunması, merkezi üçüncü taraf kuruluşlara olan ihtiyacı ortadan kaldırması, daha az QoS sinyali mesajı gerektiren yeni bir yol kurma imkanı sağlama, İSS'lerin gizlilik/güvenlik sorunlarını azaltarak iş birliği yapmalarını kolaylaştırması ve YTA ağlarında blokzincir kullanarak İSS'ler arası yönlendirme için potansiyel araştırma alanları açması.
- Projede, ismi *SoRBlock* olan bir iç ağ (intra-network) trafik yönetimi için Port ID'leri ve TTL (PIDTTL) tabanlı bir Kaynak-Yönlendirme şeması geliştirilerek, *QoSChain* çerçevesine entegre edilmiştir. Böylece, *SoRBlock* çerçevesinde, ağlar arası (inter-network) yönlendirme blokzincir teknolojisinden yararlanırken, ağ içi (intra-network) yönlendirme Kaynak-Yönlendirme şemasını kullanır. *SoRBlock* çalışması, Kaynak-Yönlendirme kavramı için ilk olmasa da; blokzincir teknolojisinden ve Kaynak-Yönlendirme şemasından yararlanarak YTA, Blokzincir ve Kaynak-Yönlendirme şemalarının birlikte ağlar içi ve arası yönlendirmede uygulanabilirliğini gösteren ilk çalışmadır.
- Projede, çeşitli yol seçim tekniklerinin *QoSChain*'in genel performansı üzerindeki etkisi kapsamlı bir şekilde ele alınmıştır. İlgili çalışma, İlk Uygun Yol Seçimi (First Feasible Path Selection - FFPS), Rastgele Uygun Yol Seçimi (Random Feasible Path Selection - RFPS)

ve Minimum Düğüm Yol Seçimi (Minimum Hop Path Selection - MHPS) olarak adlandırılan bazı temel yol seçim stratejilerini tanımlayarak; bu stratejilerin akış kurmak için gereken süre, mesaj alışverişi ve işleme sayısı, bant genişliği-düğüm sayısı çarpımı ve seçilen yolların bant genişliği ve gecikmesinden oluşan metrikler bakımından performanslarını karşılaştırmıştır.

- Projede, YTA altyapısını akıllı ve dağıtılmış bir koordinasyon çerçevesi olan *SmartContractChain (SC²)*'i oluşturmak için Blokzincir ve Akıllı Sözleşmeler (Smart Contracts - SCs) ile güçlendirilerek, Yeni Nesil Ağlara yönelik İSS ağları arası QoS özellikli trafik yönetimi geliştirilmiştir.
- Projede, YTOA'larda QoS ile ilgili bir yönlendirme modeli ve blokzincir teknolojisini birleştiren, QoS özellikli bir İSS'ler arası spektrum atama ve yönlendirme çerçevesi olan *SpectrumChain* geliştirilmiştir. *SpectrumChain* çerçevesi, potansiyel çıkar tartışmaları olan tarafları İSS'ler arasında QoS tabanlı yönlendirme için bir araya getiren bir spektrum atama ve koordinasyon çerçevesidir.
- Projede, QoS motivasyonlu ağlar arası trafik yönlendirmesini gerçekleştirmek için Q-learning algoritmasını kullanan bir Pekişirmeli Öğrenme (RL) çerçevesi geliştirilmiştir. Bu yönlendirme çerçevesinde, ağ denetleyicileri birer RL aktörü (agent) gibi çalışırken, blokzincir üzerinde işlemler olarak tutulan parça yol (pathlet) bilgilerini kullanarak ve mevcut overlay ağı üzerindeki duruma (state) göre en uygun parça yol seçimini (action) yaparak ödülü maksimize etmeye çalışmaktadır.

1.4 Sonuç Raporu Organizasyonu

Raporun 2. kısmında proje süresince yapılan çalışmalar ve çıktılarına yönelik literatür anlatılarak, ilgili çalışmalar özeti verilmiştir. 3. kısmında proje süresinde gerçekleştirilen çalışmalar ve proje çıktılarının teknik detayları, mimarileri ve çalışma prensipleri verilmiştir. 4. kısmında ise proje çıktıları ve çalışmalarına ait testler sonucunda elde edilen performans sonuçları ve bulguları tartışılmıştır. 5. bölümde projede gerçekleştirilen çalışmalar sırasında karşılaşılan çeşitli sınırlamalar, zorluklar ve gelecekteki potansiyel çalışma fikirleri verilmiştir. 6. kısmında ise sonuç bölümünü aktarılmıştır.

2. LITERATÜR ÖZETİ

Bu bölümde, proje kapsamında gerçekleştirilen çalışmalar ve konular ile ilgili olarak mevcut literatür aktarılara özeti verilmiştir. Literatürde Blokzincir teknolojisinin uygulamalarını anlatan birçok çalışma bulunmaktadır. Blokzincir, finans, IoT, uç bilişim, tarım, tedarik zincirleri, enerji piyasaları ve sağlık sistemleri gibi çeşitli alanlarda kullanılmaktadır. (Ak & Canberk, 2019;

Kamboj & Pal, 2019; Ramezan & Leung, 2018; Saad vd., 2019; Wang vd., 2018) çalışmalarında yönlendirme veya QoS çerçevelerinde blokzinciri kullanan araştırma çalışmaları bulunmaktadır. Fakat bu çalışmalar, YTA tabanlı İSS'ler için herhangi bir QoS bazlı yönlendirme çerçevesi ortaya koymamaktadır. Arins'in yaptığı çalışma (Arins, 2018), YTA tabanlı İSS'ler arasında gecikmeye duyarlı yönlendirme sağlama yöntemiyle proje kapsamında yapılan QoSChain çalışmamıza yakındır. Çalışmada önerilen mimari, birbirine bağlı her bir İSS çifti arasındaki ikili Gidiş-Dönüş Sürelerini (Round Trip Time - RTT) periyodik olarak ölçmekte ve bunları her bir İSS'nin kendi tuttuğu dağıtılmış merkeziyetsiz bir blokzincir ağında saklama imkânı sunmaktadır. Buna karşın projede önerilen QoSChain çalışması, gecikme parametresine ek olarak bant genişliğini de QoS parametresi olarak dikkate alması bakımından onların çalışmalarından farklılık göstermektedir. Ayrıca, o çalışma yalnızca İSS'ler arasındaki gecikmeyi İSS düzeyinde genel olarak dikkate alırken, QoSChain, İSS'ler içindeki kenar düğümler (edge nodes) arasındaki gecikmeyi daha ince bir ayrıntıda kullanır. Bu, uçtan uca bir yol oluştururken bir hizmet talebinin QoS kısıtlamaları ile ilgili daha fazla doğruluk sağlar.

Bir paketin tam yönlendirme yolu, kaynak (source) tarafından yalnızca Kaynak-Yönlendirme (Source Routing – SR) modeli kullanılarak belirlenebilir. Paket başlığı rota bilgisini içerir. YTA tabanlı veri merkezleri, genelde yol keşfi yerine rota kurulumuna odaklanmaktadır. (Soliman vd., 2014) tarafından tartışılan OpenFlow tabanlı sistemler için yeni bir yönlendirme yöntemi, denetleyici ölçülebilirliğini ve performans zorluklarını vurgulamaktadır. Bu çalışmada, sayı-kodlu Kaynak-Yönlendirme rotalama için denklem türetilmiştir. OpenFlow ve NOX denetleyici uzantıları kullanılarak, StEERING (Zhang vd., 2013) belirli düğümlerin geçmesini sağlayan bir rota oluşturmaya çalışmaktadır. StEERING, rota planlama sorununa bir çözüm sağlamak için Graf Teorisi'ni kullanır. StEERING'in başlıca eksikliği, QoS desteğinin olmamasıdır. En iyi rota bulmak için orta kutu prosedürlerini kullanır. SlickFlow (Ramos vd., 2013) da denetleyicinin hata kurtarma programı, özel başlıklarda rota bilgilerini giriş düğümlerine gönderir.

Veri merkezi ağını sanallaştırmak için, SecondNet (Guo vd., 2010), MPLS port anahtarlaması aracılığıyla Kaynak-Yönlendirme kullanır. SecondNet, daha kısa yollar kullandığı için sanal makinelerin çiftlerini bağlama ile ilişkilendirilen bazı sınırlamalarla karşılaşmaz. (Jyothi vd., 2015)'ün Kaynak-Yönlendirme yapısı, geliştirilmiş esneklik ve performansı hedeflemektedir. Çalışmada, yaprak-omurga veri merkezi topolojilerinde Kaynak-Yönlendirme tabanlı bir yapı tartışılmış ve önerilen yapının mevcut yönlendirme yöntemleri üzerindeki verimlilik avantajlarını tahmin etmiştir. MANET ve Ad-Hoc ağlarında en çok kullanılan Kaynak-Yönlendirme şemalarından birisi Dinamik Kaynak Yönlendirme (Dynamic Source Routing - DSR)'dır (Johnson, Hu, & Maltz, 2007). Bir paketin rota verileri, başlığında

bulunmasından dolayı aracı düğümlerin DSR'de bir yönlendirme tablosu tutmaktan kaçınmasını sağlar. Son olarak, Segment Yönlendirme (Segment Routing) (Clarence Filsfils vd., 2013), IETF tarafından önerilen basitleştirme, trafik mühendisliğinin geliştirilmesi ve hızlı yeniden yönlendirme yetenekleri sağlayan alternatif bir Kaynak-Yönlendirme tabanlı yöntemdir.

Güvenilir iletişim için topolojiye duyarsız çoklu yol kaynak yönlendirme stratejisi ve orkestrasyon çerçevesi olan M-PoLKA (Guimarães vd., 2022), Artık Sayı Sistemi (Residue Number System - RNS) polinom aritmetiğinden çeşitli özelliklerini incelemektedir. Mevcut yönlendirme protokollerı, farklı uygulamaların iletişim özelliklerini dikkate almaz. YTA'nın, kontrol ve veri düzlemlerinin ayrılması nedeniyle WAN'larda ölçeklenebilirlik sorunları yaşaması artan yanıt süresi ve maliyetle sonuçlanır. (Komajwar & Korkmaz, 2021) de yazarlar, önceden hesaplanmış çoklu yollar ve dinamik ağ durum bilgisi (Network State Information - NSI) kullanarak ve gelen akış için en iyi yolu atanmasını sağlayarak ölçeklenebilirlik, performans ve bağlantı-başarısızlık sorunlarını ele alan Kaynak-Yol Yönlendirme Modeli (Source-Path Routing Model - SPRM) çerçevesini önermişlerdir. Ayrıca, bağlantı başarısızlıklarını durumunda birbirlerinin yedeklerini alarak birden fazla yol kullanırlar. (Kumar vd., 2021), Zaman Duyarlı Ağları (Time Sensitive Networks - TSN) başarısızlık olaylarından hızla kurtarmak için Kaynak-Yönlendirme ve stateless bir veri düzlemi kullanan bir YTA tabanlı yaklaşım önermektedir. (Xia vd., 2021) de ki yazarlar, P4 tabanlı bir YTA da çalışma zamanında kontrol-veri düzlemi tutarlığını doğrulamak için SRCV adlı bir mekanizma tanıtmaktadırlar. SRCV, kaynak yönlendirme etiketleri ile aktif prob trafik kullanır, eşleşen akış kural bilgilerini toplar ve sembolik yürütme yoluyla kontrol düzlemi akış kural bilgileri ile karşılaşır. Tablo 1, Kaynak-Yönlendirme konseptini ve YTA'yı inceleyen ilgili çalışmaları özetlemektedir.

Tablo 1: İlgili Bazı Çalışmaların Özeti

ÇALIŞMA	ÖZET
(Soliman vd., 2014)	SDN tabanlı WAN birleşimini ve denetleyici konumlandırma sorunlarını çözmek için Kaynak-Yönlendirme'ye odaklanma
(Zhang vd., 2013)	Herhangi bir orta kutu dizisi üzerinden trafiği dinamik olarak yönlendirmek için bir çerçeve
(Ramos vd., 2013)	Paket başına alternatif yol verileri ekleyerek Kaynak-Yönlendirme tabanlı hızlı arıza kurtarma
(Guo vd., 2010)	Port anahtarlamalı Kaynak-Yönlendirme kullanan veri merkezi ağı sanallaştırma mimarisi
(Jyothi vd., 2015)	Gelecekteki ağ kumaşları için Kaynak-Yönlendirme tabanlı temiz soyutlama ve verimli uygulama
(Johnson, Hu, Maltz, vd., 2007)	IPv4 için Mobil Ad Hoc Ağları için Dinamik Kaynak Yönlendirme (Dynamic Source Routing - DSR)
(Clarence Filsfils vd., 2013)	Gerekli olan Kaynak-Yönlendirme işlevlerini tanımlayan IS-IS protokolü

(Guimarães vd., 2022)	Topolojiye duyarsız çoklu yol kaynak yönlendirme stratejisi ve orkestrasyon çerçevesi
(Komajwar & Korkmaz, 2021)	SDN'de ölçeklenebilirlik, performans ve bağlantı başarısızlık sorunlarını ele almak için SPRM çerçevesi
(Kumar vd., 2021)	Kaynak-Yönlendirme ve stateless veri düzlemi kullanarak ultra hızlı TSN kurtarma için SDN tabanlı bir yaklaşım

QoS bazlı yönlendirme de Blokzincir-YTA entegrasyonunu tamamlamak üzere akıllı kontratlar dahil edildiğinde, literatürde daha az sayıda çalışma bulunmaktadır. (Wang vd., 2018) de yazarlar Ethereum platformu üzerinde çalışan QoS tabanlı hizmet kompozisyonu için akıllı kontrat tabanlı bir algoritma önermektedir.

Tablo 2: YTA, Blokzincir ve akıllı sözleşmeleri kullanan çalışmaların ağ türleri, blokzincir modeli ve akıllı sözleşmelerin kullanımına ilişkin özeti

Çalışma	Ağ Tipi	Blokzincir Modeli	Akıllı Sözleşme
(Wang vd., 2018)	Veri Merkezi Ağı	Genel/İzinsiz	Evet
(Ramezan & Leung, 2018)	IoT Ağları	Genel/İzinsiz	Evet
(Kamboj & Pal, 2019)	IoT Ağları	Genel/İzinsiz	Evet
(Oktian vd., 2019)	WAN	Genel/İzinsiz	Evet
(Zeng vd., 2022)	IoT	Genel/İzinsiz	Evet
(Podili & Kataoka, 2021)	WAN	Genel/İzinsiz	Evet
(Karakus vd., 2023)	WAN	Özel/İzinli	Evet

Tablo 2 de YTA, blokzincir ve akıllı sözleşme kullanan çalışmalarında ağ türleri, blokzincir modeli ve akıllı sözleşmelerin kullanımına ilişkin durumlar özet olarak verilmiştir. (Ramezan & Leung, 2018) deki çalışma, karşılıklı olarak güvensiz IoT cihazları arasında merkezi olmayan blokzincir tabanlı Sözleşmeli Yönlendirme (Contractual Routing - BCR) çerçevesi sunmaktadır. BCR'de, gecikmeye toleranslı bir IoT ağındaki cihazlar, merkezi olmayan bir şekilde bir ağ geçidine veya hedef cihaza giden rotaları bulabilmektedir. Benzer şekilde, (Kamboj & Pal, 2019) bant genişliği ticaretini kolaylaştırmak için akıllı kontratlar ile oyun teorik bir yaklaşımıyla IoT ağları için YTA ve blokzincirden yararlanmaktadır. (Oktian vd., 2019)'daki yazarlar, kullanıcı deneyimini olumsuz etkileyebilecek hizmet kalitesindeki düşüşü önlemeyi amaçlayan İSS müşterileri için hizmet kalitesini yönetmek adına bir sistem oluşturmuşlardır. Sistemlerinde, müşteriler için QoS yönetimini basitleştirmek ve operasyonel giderleri azaltmak için YTA'nın yanı sıra, merkezi olmayan bir şekilde fikir birliğini kolaylaştıran şeffaf ve güvenilir bir ödeme sistemi sağlamak için akıllı sözleşmelerle blokzincir teknolojisi kullanılmaktadır. Uzlaşma protokolü blokzincir sistemleri için çok önemlidir. Ancak, İş Kanıtı (Proof-of-Work - PoW) gibi mevcut olanlar çok fazla enerji tüketir ve sınırlı blokzincir işlemi hacmine sahiptir. Hisse Kanıtı (Proof-of-Stake - PoS) bir alternatif olarak önerilmiştir, fakir ekonomik olarak dezavantajlı

katılımcılar için adil değildir. Zeng ve arkadaşları (Zeng vd., 2022), birden fazla denetleyici arasında güven gerektiren YTA özellikli IoT ağlarında ağlar arası yönlendirme sorununu ele almışlardır. Kötü niyetli denetleyiciler diğerlerini yanlış yolları hesaplamaları için yanıtabilir ve bu da kara delik saldırılarına (black-hole attacks) neden olabilir. Yazarlar, tüm denetleyicilerin blokzincirler ile donatıldığı ve akıllı sözleşmeler yoluyla soyut topolojiler yüklediği güvenli yönlendirmeyi sağlamak için blokzincir tabanlı bir mimari önermektedir.(Podili & Kataoka, 2021), Ethereum Blokzincir ve akıllı sözleşmeler kullanarak uçtan uca QoS'nin başarılı bir şekilde ilettilmesini garanti etmeyi ve ağların QoS uyumluluğunu doğrulamayı amaçlayan TRAQR (TRust Aware End-to-End QoS Routing) çerçevesini önermektedir. TRAQR, bir ağın güvenini, başarılı veya başarısız QoS blokzincir işlemlerine dayalı olarak ölçmek için bir ölçüt olarak güven puanını kullanır. TRAQR, uçtan uca QoS talepleri için ağlar arası QoS yollarını hesaplarken daha yüksek güven puanına sahip ara ağları seçer. Proje çalışmaları sırasında geliştirdiğimiz ve detayları ileriki bölümlerde açıklanan Smartcontractchain (SC2) çalışmamız (Karakus vd., 2023), ağlardaki bağlantıların düşük seviyeli QoS parametrelerini yine QoSChain (Karakus vd., 2021) çerçevesinden gelen bir kavram olan parça yollar (pathlets) aracılığıyla özellikle genel sisteme açık bir şekilde dahil eder ve daha yüksek başarı olasılığına sahip yollar üretmektedir.

Blokzincirin optik ağlara entegrasyonuna gelince, (Alemany vd., 2020)'deki yazarlar, merkezi bir otorite olmadan birden fazla alandan ağ kaynaklarını yönetmek ve paylaşmak için blokzincire duyarlı bir eşler arası optik ağ altyapısı sunmaktadır. (Chen vd., 2020)'de blokzincir, yüksek kaliteli bir IoT iletişim ağını güvenli bir şekilde inşa etmek için optik ağ dilimlemesini tutan kiralama işlemlerini sağlamak için kullanılmaktadır. (Ding vd., 2020)'ün yazarları, sanal optik ağları yerleştirmek ve tüm sanal optik ağlar tarafından iş birliği içinde tutulan blokzincir destekli bir defter kullanarak kurcalamayı engellemek için elastik bir optik ağlarda blokzincir çerçevesi ile güvenli spektrum ticaretini ele almaktadır. (Fichera vd., 2021)'te, ağ olayları ve eylemleri için özlü sorumluluk atfını ve güvenilir hesap verebilirliği kolaylaştmak için dağıtılmış optik ağlarda satıcı çeşitliliği ve birlikte çalışabilirlik desteği önerilmektedir. Blokzincir modeli ile optik ağlar için dağıtık bir kontrol mimarisi (Yang vd., 2019)'te önerilmiştir. Yazarlar, işlem hızı ve kurtarma gecikmesi metriklerini dikkate alarak yüksek verimli hata toleransını desteklemek için blokzincir tabanlı YTOA'yı tanımlamaktadır. (Kou vd., 2017)'daki yazarlar, güvenilir optik iletişim ağları elde etmek için blokzincir sisteme blok yazmak için dinamik liderleri belirleyen bir blokzincir konsensus protokolü tanımlamaktadır. (Yang, 2022)'de, kimlik doğrulama mekanizmasını güvenli bir şekilde tanımlamak ve çok alanlı ağlarda çeşitli izinsiz giriş senaryolarını önlemek için 6G hücresel ağ için blokzincir tabanlı optik ve kablosuz ağlar tanımlanmıştır. Çok alanlı optik ağlar için (Fichera vd., 2019)'de bit hata oranı ve optik sinyal-

gürültü oranı açısından iletim kalitesini onaylamak için blokzincirin uygulanmasının üst düzey bir açıklaması verilmiştir. (Alemany vd., 2022)'daki başka bir yaklaşım, blokzincir kullanarak çok alanlı bir ortamda optik ağ kaynaklarının paylaşımını koordine etmek için ağ dilimleme yöneticilerinin eşler arası bir ağını önermektedir. Bu çalışmaların hiçbir İSS'ler arası YTOA'larda QoS tabanlı uçtan uca yol kurulumu için yönlendirme ve spektrum atamasını araştırmamaktadır.

3. YÖNTEM

Bu bölüm, proje de gerçekleştirilen farklı çalışmaların teknik detaylarını farklı alt başlıklar olarak aktarmaktadır.

3.1 QoSChain (QC): Yazılım Tanımlı Ağlarda Blokzincir ile İnternet Servis Sağlayıcıları (İSS) Arası QoS Sağlama

Bu kısımda, proje çalışmaları kapsamında geliştirilen Blokzincir-etkin QoS-tabanlı İSS'ler arası yönlendirme çerçevesi olan *QoSChain (QC)* çerçevesinin teknik detayları aktarılmıştır. Proje çalışmalarında, İSS'ler arası uçtan uca QoS farkındalıklı yönlendirme problemi aşağıdaki şekilde tanımlanmıştır.

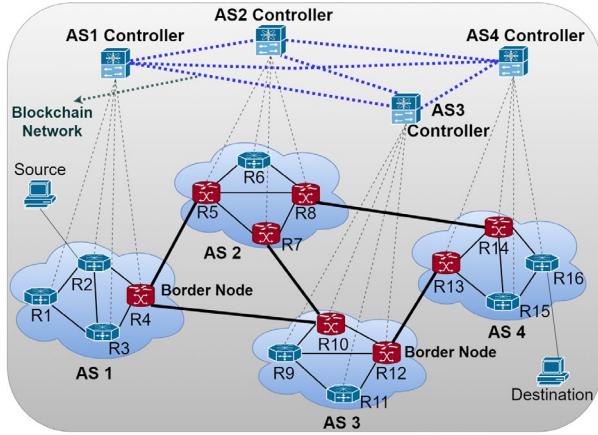
3.1.1 Sistem modeli

3.1.1.1 İSS'ler arası QoS tabanlı yol problemi (IA-QbP)

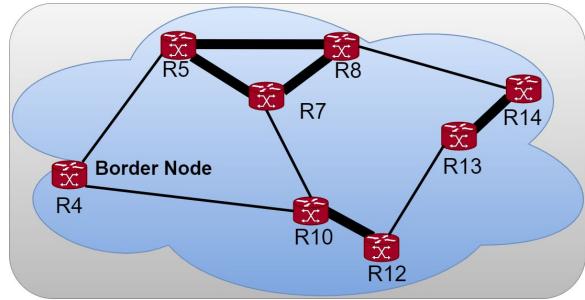
Şekil 3 de gösterildiği gibi $N_i(V_i, E_i)$ şeklinde ağ topolojilerine sahip bir İSS kümesi verildiğinde, V_i , E_i ve $*V_i$ kümeleri sırasıyla N_i de ki tüm düğümlerin, bağlantıların ve sınır düğümlerini göstersin (burada $*V_i \subseteq V_i (1 \leq i \leq i)$). Şekil 4 de İSS'lerden soyutlanmış overlay ağı topolojisi $N(V, E)$ ile gösterilmiştir. Burada $V = \bigcup_{i=1}^n *V_i$ sırasıyla İSS'lerin tüm sınır düğümlerinin ve bu sınır düğümlerin arasındaki (mantıksal) bağlantıların kümeleridir. QoS kısıtlamalarının (örneğin gecikme, bant genişliği vb.) sayısı m ile gösterildiği ve her bir $u, v \in V$, $(u, v) \in E$ bağlantısının m adet negatif olmayan QoS ağırlıkları şartını sağlayan $(w_i(u, v), i = 1, \dots, m)$, m boyutlu boyutlu bir bağlantı-ağırlık vektörüne sahip olduğunu var sayalım. Bu durumda İSS'ler arası QoS tabanlı yol problemi (IA-QbP) aşağıdaki gibi çok kısıtlı bir yol problemi olarak ifade edilebilir:

Tanım 1: İSS'ler Arası QoS Tabanlı Yol Problemi (IA-QbP): $N_i(V_i, E_i)$ Vektöründen alınmış bir $N(V, E)$ overlay ağını ele alalım. Her bir bağlantı $(u, v) \in E$, QoS sisteminin ağırlıklarından alınan $w_i(u, v) \geq 0$, $(i = 1, \dots, m)$ bir m ağırlığına sahiptir. L_i , $(i = 1, \dots, m)$, kısıtlaması verildiğinde, s kaynak düğümünden d hedef düğüme giden bir P yolu

$(s \in {}^*V_i, d \in {}^*V_j \text{ ve } {}^*V_i \neq {}^*V_j)$ sağlayan vardır, öyle ki $i = 1, \dots, m$ için $w_i(P) \stackrel{\text{def}}{=} \Sigma_{(u,v) \leq L_i} \text{eşitliği sağlanmalıdır.}$



Şekil 3: 4 YTA İSS'den oluşan temsili bir ağ modeli



Şekil 4: Şekil 3'deki İSS'lardan soyutlanmış bir katman topolojisi

Tüm m kısıtlamalarını karşılayan bir yol, (QoS tabanlı) uygun bir yol olarak adlandırılır. Kısıtlamaları sağlayan grafik $N(V, E)$ içinde birden fazla yol olabilir. Tanım 1'e göre, bu yollardan herhangi biri, $IA\text{-}QbP$ problemine bir çözümüdür.

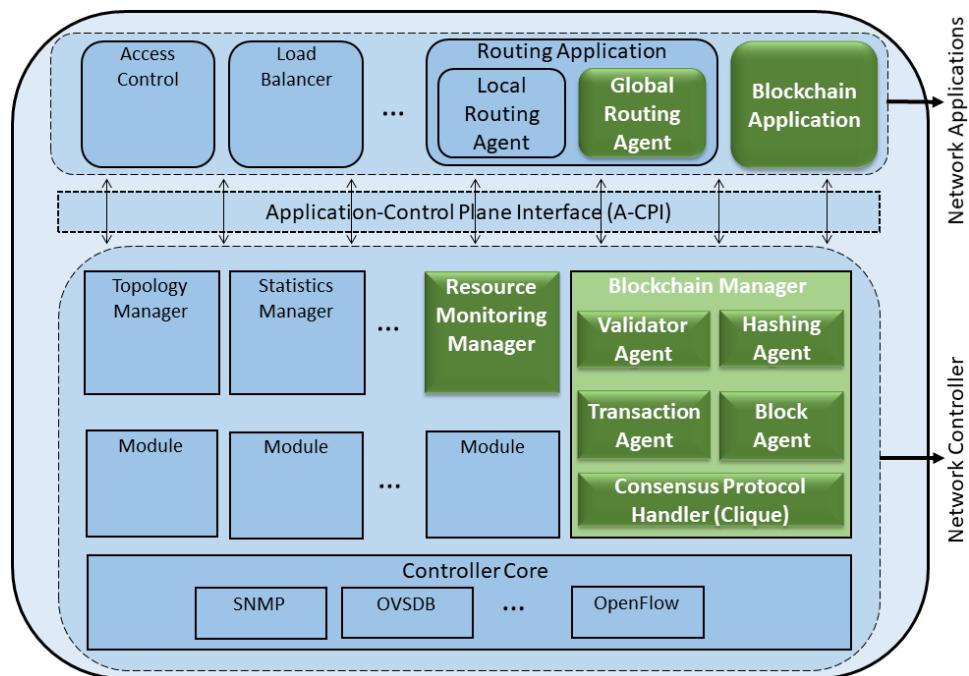
3.1.1.2 Otonom sistemler (Autonomous Systems - ASes)

Bir otonom sistem (Autonomous System – AS), IP ön-ekleri (prefixes) bir İSS'ye atanmış olan coğrafi olarak dağınık iletişim cihazlarını ve ağlarını birbirine bağlar. Her AS'ye Internet Assigned Numbers Authority (IANA) tarafından benzersiz bir Otonom Sistem Numarası (ASN) verilir.

QoSChain çerçevesinde, ağ durumlarını temsil eden blokzincir işlemleri ve blokları saklamak için AS denetleyicileri (yani AS nodeları) arasında özel bir AS ağları vardır. AS'lerin diğer tüm ağlara (fiziksel olarak) bağlanmak zorunda değildir. Şekil 3, denetleyiciler tarafından kontrol edilen 4 YTA AS'den oluşan bir ağ modelini temsil etmektedir. Ağ cihazları ya AS'ler arası bir bağlantı aracıyla farklı bir AS'deki başka bir sınır düğümüne bağlanan bir sınır düğümü (yani kahverengi silindirik objeler) ya da AS'ler arası bağlantısı olmayan bir çekirdek ağ cihazıdır (yani mavi altigen objeler). Siyah kesikli çizgiler, denetleyiciler ve ağ cihazları arasındaki kontrol yollarını (bağlantıları) gösterir. Mavi kesikli çizgiler blokzincir ağındaki denetleyiciler arasındaki mantıksal bağlantıları temsil eder.

3.1.1.3 Blokzincir uyumlu YTA denetleyici

İSS denetleyicileri, blokzincire katılan düğümleri temsil eder. Çalışmada, YTA denetleyicimizi, blokzincir özelliklerine adapte edilmiş yeni denetleyici modülleri ve ağ uygulamaları ile genişlettik. Denetleyici modülleri, ağ durumları olaylarını (topoloji, cihazlar, akışlar, vb.) keşfetmek ve ortaya çıkarmak, denetleyicinin ağ cihazlarıyla iletişimini sağlamak, ağ kaynaklarıyla ilgili istatistikleri toplamak gibi uygulamaların çoğunda ortak kullanıma sahip ağ işlevlerini uygular.



Şekil 5: Blokzincir ile uyumlu bir SDN denetleyicisinin genel bakışı: Yeşil renkli bloklar içinde beyaz metinle gösterilen yeni denetleyici modülleri ve ağ uygulamalarını içerir.

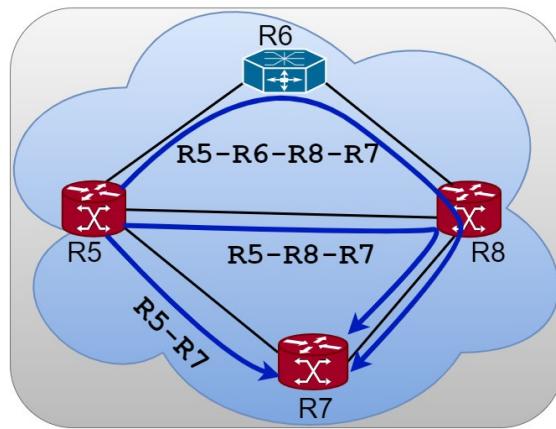
Şekil 5'te YTA denetleyicimizin modülleri, ilişkileri aşağıdaki özetlendiği gibi yeni ve mevcut modüllerle birlikte gösterilmektedir: Ana blokzincir-etkinleştirici modül, Blockchain Manager (BM) ve onun alt bileşenleridir. Bu modül bir İSS denetleyicisinde blokzincir ile ilgili tüm işlevleri gerçekleştirir. Validator Agent, blokzincir kapsamında blok doğrulama kurallarına dayalı olarak diğer denetleyicilerden gelen blokları doğrulamaktan sorumludur. Hashing Agent, blokzincir ağına gönderilecek işlemleri ve blokları hashlemekten sorumludur. Transaction Agent blokzincir işlemlerini oluştururken, Block Agent blokzincir kullanım durumuna ve uzlaşma (concensus) protokolüne bağlı olarak denetleyicide blok oluşumundan sorumludur.

Denetleyici mimarisi farklı uygulama alanlarına da sahiptir. Yönlendirme uygulamasıyla ilgili olarak, Global Routing Agent (GRA), İSS'ler arası bir hizmet isteği denetleyiciye geldiğinde İSS'ler arası yönlendirme işlemini uygular. Bu aracı, gelen hizmet talebi için bir uçtan uca yol bulmak için blokzincir defterindeki yeni işlemlerin giriş ve çıkış alanlarını giriş parametreleri

olarak alır ve Blokzincir Uygulamasını (Blockchain Application / BA) devreye sokar. BA ayrıca blokzincir ağına blok göndermek/almaktan ve BM ile koordineli olarak hizmet ve parça yol (pathlet) taleplerini yönetmekten sorumludur.

3.1.1.4 Parça yol (Pathlet)

Bu çalışmada, bir parça yolu İSS'deki ağ sınır düğüm çiftleri (giriş/çıkış düğümleri) arasındaki benzeri olmayan bir yol olarak tanımladık. Bir parça yolun uç noktaları giriş ve çıkış düğümleri olarak adlandırılır.



Şekil 6: İSS2 ağında R5 ve R7 ağı cihazları arasındaki yollar

Tanım 2: Pathlet: $N_i(V_i, E_i)$ olarak temsil edilen bir topolojiye sahip olan N_i İSS ağı, $P = < v_i^1, \dots, v_i^t >$ yolu ($\forall v_i^j \in P \in V_i$, $v_i^1, v_i^t \in {}^*V_i$ ve v_i^j 'den v_i^{j+1} 'e bir bağlantı olan e_i^l ($\forall v_i^j, v_i^{j+1} \in P, \forall e_i^l \in E_i$) şartlarını sağlıyorsa **parça yol (pathlet)** olarak adlandırılır.

Şekil 6'te gösterildiği üzere, mavi oklu çizgiler ile gösterilen R5-R7, R5-R8-R7 ve R5-R6-R8-R7 (bağlantıların çift yönlü olduğu varsayılarla) yolları Tanım 2'ye göre İSS2'deki R5 ve R7 sınır düğümleri arasındaki parça yollardır. Parça yollar, tüm sınır düğümü çiftleri arasında farklı İSS denetleyicileri tarafından kendi ağları için hesaplanır. Çalışmanın ilerleyen bölgelerinde açıklanacağı üzere, bir blokzincir işlemi parça yol düğümleri ve QoS değerlerinden oluşturulur.

3.1.1.5 Servis İsteği

Yönlendirme, ağlarda alan (domain) içi ve alanlararası bağlantı sağlamak için kullanılan yöntemlerden biridir. Bu bağlamda, QC çerçevesinde, bir hizmet talebi, aynı veya farklı ağlardaki kullanıcılar (host, sunucu) arasında bant genişliği ve gecikme gibi belirli QoS parametreleri ile bağlantı kurulmasını anlamına gelir.

Type	Service ID	Nonce
Source IP		
Destination IP		
Bandwidth		
Delay		
Bandwidth_Max		
Delay_Min		
Hop_Min		

Şekil 7a: S_Req mesajı veri yapısı

Type	Pathlet RequestID	Nonce
Pathlet ID		
Ingress Node		
Egress Node		
AS Number		
Source IP		
Destination IP		
Bandwidth		
Delay		

Şekil 7b: P_Req mesajı veri yapısı

Type	Pathlet Response ID	Nonce
Pathlet_Request ID		
Pathlet ID		
AS Number		
Response		

Şekil 7c: P_Res mesajı veri yapısı

Type	Service Response ID	Nonce
Service_Request ID		
Response		

Şekil 7d: S_Res mesajı veri yapısı

Şekil 7: QoSChain çerçevesindeki mesaj veri yapıları

Şekil 7, QoSChain çerçevesindeki Servis İsteği (S_Req), Yolcu İsteği (P_Req), Yolcu Yanımı (P_Res) ve Servis Yanımı (S_Res) mesajlarının veri yapılarını göstermektedir.

Şekil 7a bir S_Req mesajının veri alanlarını göstermektedir:

- *Type*: Bu alan, iletinin bir mesaj olup olmadığını gösterir. Alan S_Req, bir P_Req, bir P_Res veya S_Res mesajı değerlerini alabilir.
- *Service ID*: Değişmeyen bir hizmet tanımlayıcısı.
- *Nonce*: Mantıksal olarak benzersiz bir Servis ID oluşturmak için Service ID'ye eklenen rastgele sayı.
- *Source IP* ve *Destination IP*: Kaynak ve hedef hostların ip adresleri.
- *Bandwidth* ve *Delay*: Talep edilen bant genişliği ve servis için gerekli olan uçtan uça yol üzerindeki gecikme değeri.
- *Bandwidth_max*, *Delay_min* ve *Hop_min*: Bu parametreler, birden fazla yol olması durumunda kullanılacak yolların önceliğini belirtir.

S_Req mesajı bir kullanıcı bilgisayarında yüklü olan bir uygulama ile kaynak İSS denetleyicisine gönderilir.

Şekil 7b bir P_Req mesajının veri alanlarını göstermektedir:

- *Pathlet Request ID*: Bir parça yol istek tanımlayıcısı.

- *Nonce*: Benzersiz bir *Pathlet Request ID* oluşturmak için Pathlet Request ID e eklenen rastgele sayı.
- *Pathlet ID*: Herhangi İSS'den Servis için talep edilinen parça yolun tanımlayıcı numarası.
- *Ingress Node ve Egress Node*: İSS sistemden talep edilen parça yolun giriş ve çıkış düğümleri.
- *AS Number*: parça yolların talep edildiği otonom sistem cihazının benzersiz tanımlayıcı numarası.
- *Source IP ve Destination IP*: Kaynak ve hedef hostların ip adresleri.
- *Bandwidth ve Delay*: Talep edilen bant genişliği ve servis için gerekli olan uçtan uça yol üzerindeki gecikme değeri.

P_Req mesajları kaynak İSS'taki denetleyicide bulunan BA modülü tarafından İSS'lere servis içi seçilen son yol üzerinden gönderilir.

Şekil 7c bir P_Res mesajının veri alanlarını göstermektedir:

- *Pathlet Response ID*: Bir parça yol cevap tanımlayıcısı.
- *Nonce*: Benzersiz bir *Pathlet Response ID* oluşturmak için Pathlet ResponsesID'ye eklenen rastgele sayı.
- *Pathlet Request ID*: Herhangi İSS'den Servis için talep edilinen parça yolun tanımlayıcı numarası.
- *Pathlet ID*: İSS denetleyicisinde talep servis edilen gönderilen parça yolun *ID'si*.
- *AS Number*: parça yolun talep edildiği otonom sistem cihazının benzersiz tanımlayıcı numarası.
- *Response*: AS'nin talep edilen parça yolu verip, vermeme durumunu belirten alan *Accept* ya da *Reject* değerlerini alabilir.
- *Bandwidth ve Delay*: Talep edilen bant genişliği ve servis için gerekli olan uçtan uça yol üzerindeki gecikme değeri.

P_Res mesajları tüm İSS'ler üzerinde, BA modülü tarafından talep edilen servis için son uçtan uça yol üzerinden Kaynak İSS'ye doğru denetleyiciler tarafından gönderilir.

Şekil 7d bir S_Res mesajının veri alanlarını göstermektedir:

- *Service Response ID*: Bir servis cevap tanımlayıcısı.
- *Nonce*: Benzersiz bir *ServiceResponse ID* oluşturmak için Service Response ID'ye eklenen rastgele sayı.
- *Service Request ID*: S_Req mesajının ID Nonce ve Service ID alanlarının birleştirilmesiyle oluşan değer.

- *Response*: Kaynak İSS denetleyicisi tarafından *P_Res* mesajı değerlendirildikten sonra kaynak İSS'nin talep edilen servisi verip vermeme konusundaki kararı. *Accept* ya da *Reject* değerlerini alabilir.

S_Res mesajları kaynak İSS denetleyicisinde bulunan BA modülünde, servislere ve kullanıcılara gönderilmek üzere oluşturulur.

3.1.2 Blokzincir-etkin QOS-tabanlı İSS'ler arası yönlendirme çerçevesi: QoSChain

3.1.2.1 Blokzincir modeli

Blokzincir düğümleri *QoSChain* çerçevesinde, her bir blokzincir düğümü bir İSS denetleyicisine karşılık gelir ve kendi blokzincir örneğini çalıştırır. Bu nedenle, bir blokzincir düğümü ve İSS denetleyicisi kavramlarını birbirlerinin yerine kullanılmıştır. Bir blokzincir düğümünde iletişimini sağlamak için bir IP adresi ve kriptografik işlemler için bir çift açık (public) ve gizli (private) anahtar bulunur.

Eş Olma (Peering): İSS denetleyicileri, bir eş olma ilişkisi oluşturmak için açık anahtarlarını karşılıklı olarak değiştirirler. Blokzincir düğümleri diğer blokzincir düğümlerini ayırt etmek için açık anahtarlarını AS Numarası (ASN) olan blokzincir düğümleri tanımlayıcıları ile ilişkilendirirler. Blokzincir düğümleri açık anahtarlarını; IP, ASN, sınır düğümlerin listesi gibi verilerle birlikte dijital olarak imzalayarak diğer diğer düğümler ile paylaşırlar ve bunun karşılığında diğer düğümlerden da bu bilgileri bekler.

Blokzincir İşlemi (Transaction): Blokzincir işlemi, bir blokzincir ağı üzerinde tek bir veri aktarımı işlemi veya blokzincirde bu tür bir işlemin kaydına verilen isimdir. Blokzincir işlemi, blokzincirde kaydedilen durumu günceller. Kullanım durumlarına göre farklı veriler içerebilirler. QC çerçevesinde, blokzincir işlemi, düğümler tarafından QoS değerleriyle birlikte parça yollardan oluşturulur. Bundan ötürü, tüm sınır düğüm çiftlerinin farklı parça yolları, İSS denetleyicileri tarafından kendi ağları için hesaplanır. Şekil 8, QC çerçevesinde bir İSS denetleyicisi tarafından oluşturulan bir blokzincir işleminin veri yapısını göstermektedir. Bu blokzincir işlemi aşağıdaki veri alanlarını içerir:

- *Tx ID*: Değişmeyen bir blokzincir işlemi tanımlayıcısı.
- *Signature*: Blokzincir işleminin dijital imzası.
- *ASN*: İSS'nin Benzersiz Otonom Sistem Numarası.
- *Pathlet ID*: Otonom sistemdeki birbirinden farklı parça yollara verilen benzersiz ID.
- *Ingress Node* ve *Egress Node*: İSS sistemden talep edilen parça yolun giriş ve çıkış düğümleri.
- *Max Bandwidth*: İSS parça yolları tarafından sunulan ulaşılabilir maksimum bant genişliği.
- *Min Delay*: İSS parça yolları tarafından sunulan ulaşabilecek minimum gecikme.

Tx ID	Signature	ASN	Pathlet ID	Ingress Node	Egress Node	Max Bandwidth	Min Delay
-------	-----------	-----	------------	--------------	-------------	---------------	-----------

Şekil 8: QC çerçevesinde parça yolların blokzincir işlemi veri yapısı

QoSChain çerçevesinde düğümler, gelen blokzincir işlemlerini bir dizi kurallara uyup uymadığını denetler. Bu kurallar (i) blokzincir işleminin dijital olarak imzalandığını, (ii) blokzincir işleminin veri yapısındaki hiçbir veri alanının boş olmadığını, (iii) bir blokzincir işleminin QoS ile ilgili veri alanlarının blokzincir işleminin veri yapısında pozitif değerler olduğunu, (iv) blokzincir işlemi veri yapısının ASN alanında geçerli bir İSSN'nin mevcut olduğunu ve (v) blokzincir işlemi giriş ve çıkış düğümlerinin ID'sinin blokzincir işlemi oluşturan İSS'ye ait olduğunu doğrular.

Tablo 3: Şekil 3'da İSS2'de R5 ve R7 arasındaki parça yolları için İSS2

Tx ID	Signature	ASN	Pathlet ID	Ingress Node	Egress Node	Max Bandwidth	Min Delay
AS2_1	0xa54be...	AS2	R5_R7_1	R5	R7	15	8
AS2_2	0x19a7d...	AS2	R5_R7_2	R5	R7	5	12
AS2_3	0xb8f8e...	AS2	R5_R7_3	R5	R7	10	15
AS2_4	0x5c91a...	AS2	R5_R7_2	R5	R7	20	12
AS2_5	0x42ae7...	AS2	R5_R7_3	R5	R7	20	15
AS2_6	0xacd42...	AS2	R5_R4_1	R5	R4	40	30
...

Tablo 3, Şekil 3'de gösterilen İSS2 ağındaki R5 ve R7 sınır cihazları arasındaki parça yollar için İSS2 denetleyicisi tarafından oluşturulan örnek bazı blokzincir işlemlerini göstermektedir. R5-R8 ve R7-R8 gibi diğer sınır cihazları çiftleri için de blokzincir işlemleri aynı şekilde oluşturulur ve blokzincir ağında çoğaltılar. Yeni bir İSS, blokzincir ağına katıldığında, sınır ağı cihazları arasındaki parça yollar için başlangıç blokzincir işlemleri (sırasıyla R5-R7, R5-R8-R7, R5-R6-R8-R7 parça yolları için AS2_1, AS2_2 ve AS2_3) oluşturmaya başlar. Daha sonra bunlar ağa veya uygun blokzincir düğümlerine yayılır. Adba QoS ile ilgili bir durum değişikliği olduğunda: örneğin bir bağlantıda bazı parça yolların durumunu etkileyen bant genişliği güncellemesi, denetleyici ilgili parça yollardaki durum değişikliğini yansıtan güncelleme (yeni) blokzincir işlemi (AS2_2 ve AS2_3 için sırasıyla AS2_4 ve AS2_5, R7 ve R8 arasındaki bağlantıda bant genişliğinin 15 Mbps artlığı varsayılarak) oluşturur. AS2_6 ID'li blokzincir işlemi, sırasıyla İSS2 ve İSS1'de R5'ten R4'e ara bağlantı bağlantısı için oluşturulur. Bir parça yol üzerindeki tüm ağ cihazı listesi yalnızca ilgili blokzincir işlemini yaratan denetleyici tarafından bilinir (örnekte İSS2 denetleyicisi). Bir parça yolun denetleyicisinin blokzincir defterinde birden fazla blokzincir işlemi varsa, denetleyici bir servis isteği için uçtan uca yol

hesaplarken bunlardan en blozkincir işlemlerinin sonucusunu yani en güncel olanını kullanacaktır.

Block: Bir blok, veri yapıları ve imzaları doğrulandıktan sonra doğrulayıcı bir düğüm tarafından belirli bir veri yapısında gruplanan blozkincir işlemlerini içerir. Bir blok, blozkincirin kriptografik güvenliğini sağlayan mekanizmalardan biri olan önceki bloğa bir referans olarak bir hash pointer içerir. QC çerçevesinde, baş düğüm mevcut blok dönemi için bir blok oluşturur. Baş düğüm rolü, her dönemde düğümler listesinden bir sonraki düğüme aktarılır. Baş düğüm tarafından tüm yeni blozkincir işlemleri, blozkincir işlemi havuzundan toplanır ve aşağıdaki prosedürler uygulanır.

- Blozkincir işlemleri, blozkincir işlem doğrulama kurallarına göre incelenir. Geçersiz blozkincir işlemleri reddedilir.
- Baş düğüm blokların kurallara uyup uymadığını izler. Varsa, zaman sınırları gibi blok üretim sınırları gibi parametreler kullanılabilir.
- Geçerli blozkincir işlemleri içeren bir blok oluşturulur ve baş düğüm gizli anahtarını kullanılarak bloğu imzalar.
- Yeni oluşturulan blok, blozkincirdeki diğer düğümlere iletir.

Yukarıdaki adımlar takip edilerek yeni blok üretimi gerçekleşir. Eğer baş düğüm geçersiz blok üretmeye devam ederse blozkincir ağından yasaklanabilir ya da eş düğümler tarafından dışlanabilir.

3.1.2.2 QoSChain çerçevesinin blozkincir temelli tasarım bileşenleri

Çalışmada ki İSS ihtiyaçları ile örtüşen, QC çerçevesini kullanan ve Clique aracılığıyla gerçekleştirilen blozkincir odaklı tasarım temelleri ve bunların QoS tabanlı İSS'ler arası yönlendirmedeki avantajları Tablo 4'de gösterildiği şekilde açıklanmaktadır.

Tablo 4: QoSChain Çerçevesinin Tasarım prensipleri

Blozkincir Temelli Tasarım Özellikleri	Avantajları
Kısa blok üretim süresi	<ul style="list-style-type: none"> • Ağ durumu hakkında yüksek tutarlılıkta veri • Daha fazla blozkincir işlem hacmi
Kolay blok üretimi	<ul style="list-style-type: none"> • Düşük hesaplama maliyeti • Düşük enerji tüketimi
Demokratik blok üretimi	<ul style="list-style-type: none"> • Paylaşılmış blok yükü

Kısa blok üretim süresi: Tam koordine QoSChain çerçevesi açısından doğru ve güvenilir bir QoS tabanlı uçtan uca yol oluştururken, parça yol durum güncellemlerinin hızlı bir şekilde yansıtılması (bu şekilde diğer düğüm hızıca bu veriye erişebilir) kritik öneme sahiptir. Üstelik kısa blok oluşturma süresi, işlem hacminde bir artışa neden olabilir.

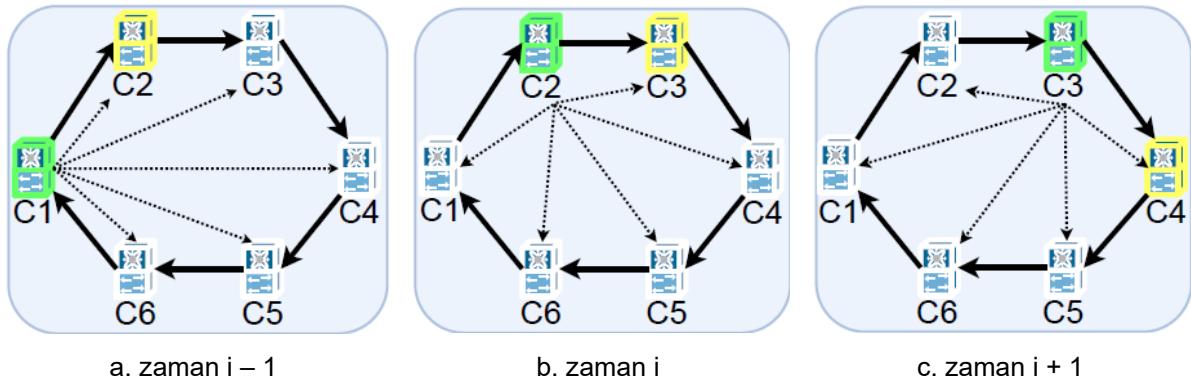
Kolay blok üretimi: Denetleyicilere blok üretimi için yoğun hesaplama gerektiren görevler yüklemek denetleyicilerin performansını önemli ölçüde düşürecektir. Ayrıca, hesaplama maliyeti düşük bir blok üretim mekanizması enerji tüketimini düşürecektir.

Demokratik blok üretimi: Demokratik bir blok üretim mekanizmasının kullanılması blokzincirin yürütülmesinden kaynaklanan yükün eşit bir şekilde paylaşılmasını teşvik etmektedir. Bu da denetleyicilerin aşırı yüklenmesini engeller.

3.1.2.3 Clique uzlaşma protokolü

Çalışmada, Clique (De Angelis, 2018) adı verilen Yetki Kanıtı (Proof-of-Authority / PoA) uzlaşma algoritmasının bir uygulaması kullanılmıştır. Clique uzlaşma mekanizmasının, İş Kanıtı (Proof-of-Work / PoW) gibi donanım / hesaplama kaynakları ve Hisse Kanıtı (Proof-of-Stake / PoS) gibi mevcut bir "pay" gerektiren diğer uzlaşma modellerine göre Tablo 4'de açıklanan tasarım noktalarıyla uyumlu birçok önemli özelliğe sahip olması, QC çerçevesinde bizi kullanmaya teşvik etmiştir.

Clique'in bir avantajı, PoW ve PoS mekanizmalarında sapma gösterebilen blok oluşturma zaman aralığının tahmin edilebilir ve daha kısa olmasıdır. Ayrıca, diğer uzlaşma mekanizmalarına oranla yüksek bir blokzincir işlem oranı sağlar. Bloklar, yetkili ağ düğümleri tarafından belirli aralıklarla bir sıra halinde oluşturulur. Bu da blokzincir işlem onaylarının hızını artırır. Bir diğer avantajı ise PoA uzlaşma yöntemi, PoW uzlaşma yöntemi gibi diğer popüler mekanizmaların aksine, düğümlerin karmaşık matematiksel görevleri çözmek için hesaplama kaynakları ayırmasını gerektirmemesidir. Bu özellik, QC çerçevesindeki denetleyiciler için ön plana çekmektadır çünkü ağa ilgili görevler ve mesajlar, ağların günlük operasyonlarında hesaplama kaynakları açısından blokzincir düğümlerini, yani denetleyicileri zaten zorlamaktadır. Bu nedenle, yüksek performanslı ekipmana ihtiyaç yoktur. Ayrıca Clique, blokzincir işlemleri işlemek, bir blok önermek ve önerilen bloğu her aralık için round-robin şeklinde blokzincir ağına yaynlamak için en fazla $N-(N/2+1)$ birincil ve ikincil blokzincir düğümü belirler.



Şekil 9: Clique'de birincil (yeşil) ve ikincil (sarı) blokzincir düğümlerinin seçim süreci

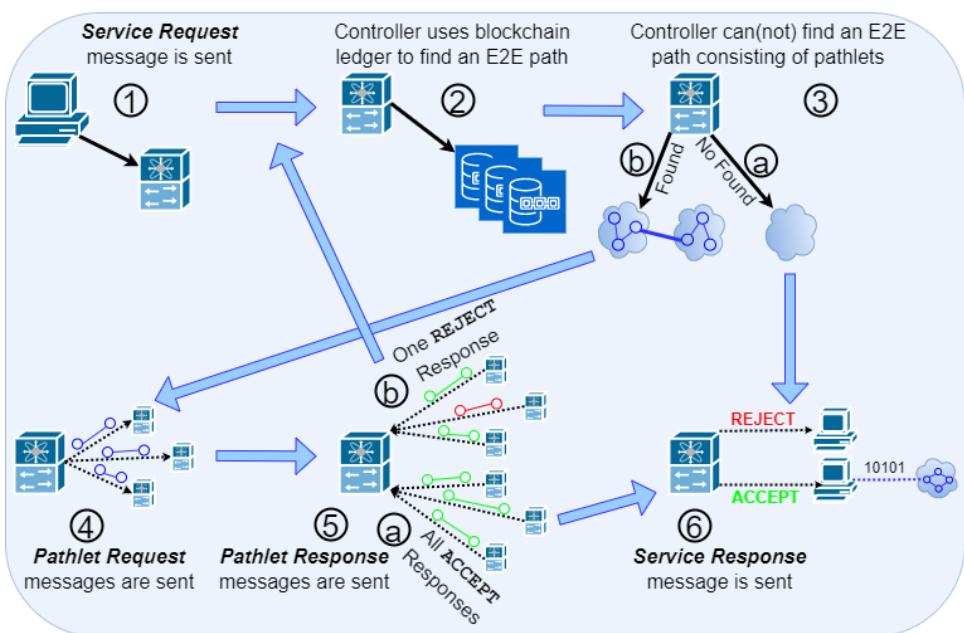
Şekil 9, N = 6 blokzincir düğümü (yani denetleyicileri) durumunda bir bloğu önermek ve ağa yaymak için birincil (yeşil) ve ikincil (sarı) blokzincir düğümlerinin seçilme sürecini üç ardışık adımda göstermektedir. İlkinci düğüm, birinci düğümün bloğu önermemesi durumunda sorumluluğu devralır. Bu sıralı süreç, blok önerme görevini düğümler arasında demokratikleştirir ve blok üretimi ve yayılımından kaynaklanan yükü blokzincir katılımcıları arasında paylaştırır.

3.1.2.4 QoSChain çerçevesinin iş akışı

Bu alt bölümde, QC çerçevesinin nasıl çalıştığı ve bir YTA ağında hizmet taleplerini nasıl ele aldığı Şekil 10'da verilen bir örnekle kısaca açıklanmaktadır. Bir kullanıcının İSS denetleyicisine S_Req mesajı şeklinde QoS tabanlı bir İSS'ler arası hizmet talebi geldiğinde (adım 1), denetleyici blokzincir defterini kullanarak mesajda belirtilen QoS parametrelerini ve önceliklerini dikkate alarak bir uçtan uca yol bulmaya başlar (adım 2). Uçtan uca yol, Şekil 4'de gösterildiği gibi, kaynak-İSS'nin bir sınır düğümünden hedef-İSS'nin bir sınır düğümüne, blokzincirdeki parça yollara (pathlets) yönelik blokzincir işlemlerinden yararlanılarak soyutlanmış bir katman ağı üzerindeki yollardan oluşur.

Kaynak-İSS denetleyicisi, uçtan uca yolunu bulmak için Tablo 1'de gösterildiği gibi blokzincir işlemlerinin Ingress Node, Egress Node, Max Bandwidth ve Min Delay alanlarını kullanır. Kaynak-İSS denetleyicisi S_Req mesajında (adım 3a) belirtilen koşulları karşılayan bir uçtan uca yol bulamazsa, hizmet talebinin sağlanamayacağını belirtmek için kullanıcıya/istemciye REJECT (RET) yanıtını içeren bir S_Res mesajı gönderir. Hizmet talebinin QoS gerekliliklerini karşılayan bir uçtan uca yol bulduğunda (adım 3b), kaynak-İSS denetleyicisi, uçtan uca yol üzerinde bir parçalı yola sahip olan her İSS denetleyicisine (adım 4) P_Req mesajlarını göndererek ilgili parçalı yolun üzerinden istenen QoS değerlerini sağlayıp sağlayamayacaklarını sorar. Bu sorgulamanın amacı, ilgili parçalı yolun ilgili blokzincir

işleminde ilan edilen QoS değerlerini hala sağlayabildiğini teyit etmektir. Blokzincirde ki (blokzincir işlemlerinin) güncellemler ağların parçalı yollarındaki durum güncellemlerini gibi gerçek zamanlı olmadığından, bir parçalı yolun QoS değerleri, İSS-içi trafik ve/veya İSS'ler arası başka bir trafik gibi belirli nedenlerden dolayı ilan edilenlerden farklı olabilir. P_Req mesajlarını aldıktan sonra, ilgili İSS denetleyicileri kaynak-İSS denetleyicisine P_Res mesajlarını geri göndererek cevap verir ve talep edilen QoS parametreleri ile ilgili parçalı yolu sağlayıp sağlayamayacaklarını (ACCEPT veya REJECT yanıtı) belirtir.



Şekil 10: QoSChain çerçevesinde uçtan uca yol bulma süreci

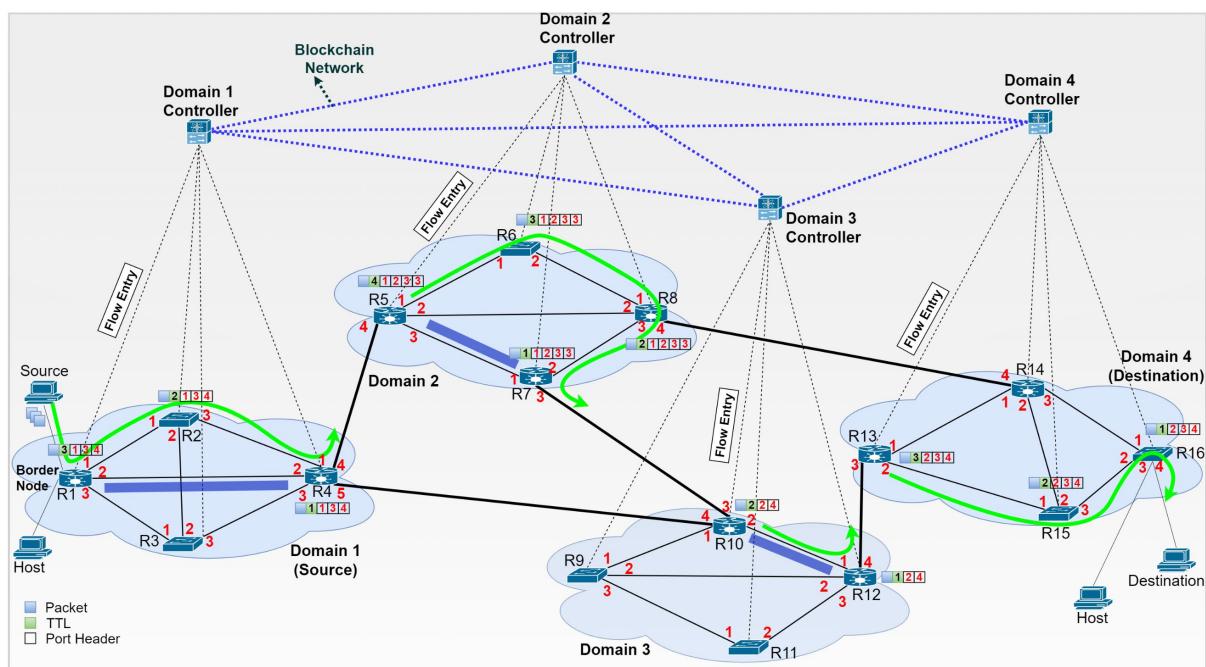
Son olarak, ilgili tüm İSS denetleyicileri kaynak-İSS denetleyicisine P_Res mesajlarında bir ACCEPT (KABUL) yanıtı gönderirse (Adım 5a), kaynak-İSS denetleyicisi kullanıcıya bir KABUL yanıtı içeren bir S_Res mesajı göndererek (Adım 6) hizmet talebinin karşılanabileceğini ve dolayısıyla ağa başlayabileceğini belirtir. İlgili ağa cihazlarının akış tablolarına akış girişlerinin eklenmesi ve ağ kaynaklarının vaat edilen parçalı yol için rezerve edilmesi gibi ağ ile ilgili temel düzenlemeler, ilgili İSS denetleyicileri tarafından S_Req ve P_Req mesajlarındaki verilere göre, hizmetin/parçalı yolun sağlanması kabul etmeleri halinde gerçekleştirilir. Bu nedenle, bir hizmet akışı İSS ağına girdiğinde uçtan uca yolu üzerinden herhangi bir İSS denetleyicisi tarafından tekrar araştırılmaz. Aksine, uçtan uca yolu üzerindeki İSS denetleyicilerinden herhangi biri kaynak İSS denetleyicisine P_Res mesajlarında (Adım 5b) bir RED yanıtı gönderirse, kaynak-İSS denetleyicisi daha önce açıkladığı gibi blokzincir defterini kullanarak aynı özelliklere ve koşullara sahip başka bir uçtan uca yolu bulmaya başlar.

3.2 Yazılım Tanımlı Ağlarda Trafik Yönetimi İçin Blokzincir Destekli Kaynak-Yönlendirmenin (Source Routing – SR) Uygulanması

Bu bölüm, projede geliştirilen Kaynak-Yönlendirme (Source Routing - SR) temelli uçtan uca yönlendirme çerçevesi olan *SoRBlock*'un teknik detaylarını iki kısımda sunmaktadır. İlk olarak, blokzincir tabanlı bir alanlar arası (inter-domain) düzeyinde yönlendirme sunulmaktadır. Daha sonra, uçtan uca bağlantı oluşturmak için YTA tabanlı Kaynak-Yönlendirme yaklaşımını kullanan alan-içi (intra-domain) düzeyinde yönlendirme sunulmaktadır.

3.2.1 Blokzincir tabanlı ağlar arası yönlendirme

Blokzincir destekli alanlar arası yönlendirme ve Kaynak-Yönlendirme tabanlı alan-içi yönlendirme düzenleri ile önerilen trafik yönetimi çerçevesi *SoRBlock* Şekil 11 de sergilenmektedir.



Şekil 11: SoRBlock da 4 SDN alanı için blokzincir destekli alanlar arası yönlendirme ve Kaynak-Yönlendirme tabanlı alan-içi yönlendirme düzenleri

SoRBlock da altyapı düğümleri iki türdedir: (i) Farklı bir alandaki bir sınır cihazına (yuvarlak cihazlar) birleştirme bağlantısı ile bağlanan bir sınır cihazı ve (ii) birbirine bağlı bağlantı olmayan bir çekirdek ağ düğümü (köşegen cihazlar). Siyah kesikli çizgiler, denetleyiciler ve altyapı düğümleri arasındaki bağlantıları temsil eder. Denetleyiciler arasında bir blokzincir ağı da bulunmaktadır. Mavi kesikli çizgiler, denetleyiciler arasındaki blokzincir içindeki bağlantıları gösterir. Kalın mavi çizgiler, alanlardaki kaynak alanı (Şekil 11'te Domain 1) denetleyicisi tarafından hesaplanan uçtan uca yol boyunca yolu oluşturan yolculukları

gösterir. Yeşil çizgiler, alanların denetleyicileri tarafından hesaplanan Kaynak-Yönlendirme tabanlı iç alan yollarını temsil eder. Kırmızı numaralar, ağ düğümlerinin bağlantı noktası kimlik numaralarıdır.

SoRBlock çerçevesinde kullanılan denetleyici, yeni modüllere ve topoloji yöneticisi, istatistik yöneticisi gibi standart modüllere sahiptir. Blockchain Manager (BM) ve alt bileşenleri, temel blokzincir etkinleştirici modülüdür. Bu bileşen, tüm blokzincir ile ilgili işlemleri gerçekleştiren bir alan denetleyicisidir. Validator Agent, blokzincirin blok doğrulama kriterlerine göre diğer denetleyicilerden yeni blokları kanıtlayabilir. Hashing Agent, işlemleri ve blokları blokzincire iletmek için önce karma işlemleri ve blokları gerçekleştirir. Transaction/Block Agent, blokzinciri oluşturan işlemleri/blokları uygulamaktan sorumludur. Consensus Protocol Handler, blokzincir uzlaşma mekanizmasını yürütür. Resource Monitoring Manager (RMM), ağ kaynaklarını, bant genişliği, gecikme ve titreme gibi sürekli izler ve ağ değişikliklerini tespit eder. Herhangi bir güncelleme olduğunda BM modülünü uyarır ve ilgili işlem(ler)i oluşturması için talimat verir. Ayrıca, denetleyici tasarımda yeni uygulamalar bulunmaktadır. Global Routing Agent (GRA), bir denetleyici bir ara alan hizmet isteği aldığıda çapraz alan yönlendirme yeteneklerini gerçekleştirir. GRA, gelen hizmet isteği adına uçtan uca rota belirlemek için işlemlerin giriş ve çıkış alanlarını kullanır. Blokchain Application (BA), blokları iletmek/almak ve hizmet istekleri ile ilgili mesajları yönetmekten sorumludur.

SoRBlock çerçevesinde kullanılan bir parça yol (pathlet) konsepti, bölüm 3.1 de *QoSChain* çerçevesinde verilen model ile aynıdır. Diğer bir ifade ile, parça yol, bir alan içindeki iki sınır düğüm çiftini (giriş ve çıkış düğümleri) birbirine bağlayan bir yol olarak adlandırılır. Bir parça yolun başlangıç ve sonuç düğümleri, içeriye giriş (ingress) ve dışarıya çıkış (egress) düğümleri olarak adlandırılır. Örneğin, Şekil 11'te gösterildiği gibi, R10 ve R12 arasında (çift yönlü bağlantılar varsayırsa) R10-R12, R10-R9-R12 ve R10-R9-R11-R12 yolları Domain 3'teki sınır cihazları arasında birer örnektir.

Her blokzincir düğümü, alan denetleyicisini temsil eder ve blokzincir replikasını, alanlar arası yönlendirme mekanizmasını kullanarak işletir. Alanlar arası yönlendirme bağlamında, blokzincir düğümleri, parça yollardan ve bunlarla ilişkili QoS değerlerinden işlem oluşturur. Bu nedenle, alan denetleyicileri tüm ağ sınır düğüm çiftleri için ayrı parça yollar hesaplar. Bir işlemin veri yapısı aşağıdaki unsurları içerir: Tx ID, İmza (Signature), Alan Numarası (Domain Number), Pathlet ID, İçeriye Giriş Düğümlü (Ingress), Dışarıya Çıkış Düğümlü (Egress), Maksimum Bant Genişliği (Max Bandwith) ve Minimum Gecikme (Min Delay). Bir alan blokzincir ağına eklenirse, kenar ağ düğümleri arasındaki parça yollar için ilk işlemleri üretmeye başlar. Bunlar ardından blokzincirin uzlaşma prosedürüne bağlı olarak ağa veya ilgili blokzincir

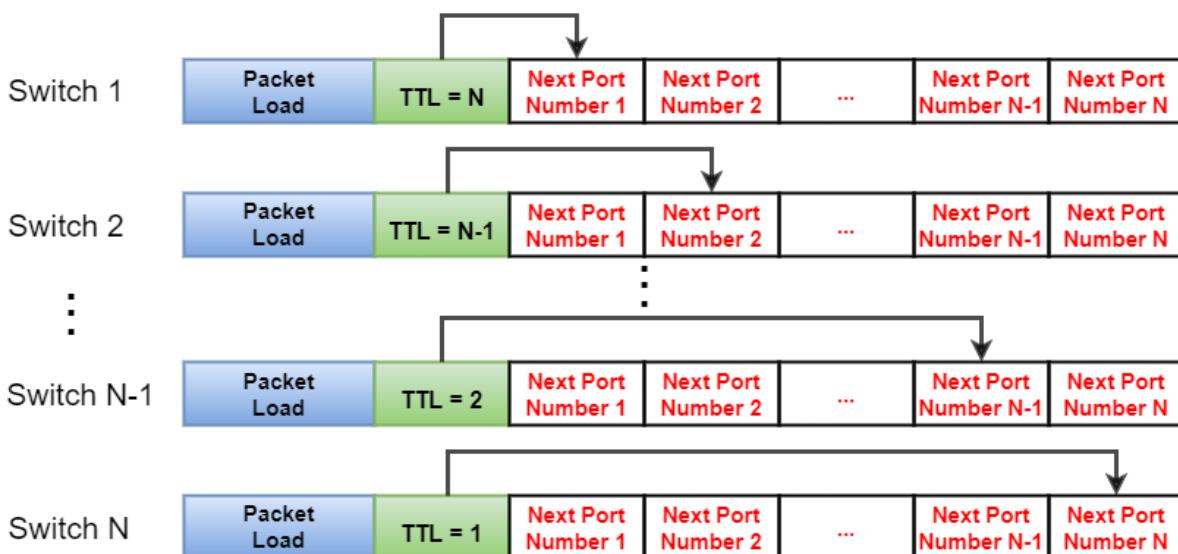
düğümlerine bloklar olarak yayılır. Bir ağdaki QoS'u etkileyen bir ağ değişikliği, örneğin, bir bağlantıdaki bant genişliği ayarı, bir parça yolun durumunu etkilediğinde, denetleyici, parça yolun değişikliğini gösteren yeni bir işlem üretir. Denetleyiciler, hizmet için bir uçtan uca yol bulurken işlem kimliklerini kontrol ederek, aynı kimliğe sahip parça yollar için en güncel işlemi kullanır. Bu işlemin mantığı, en güncel işlemin ağındaki QoS parametreleri açısından ilgili parça yolun mevcut durumunu yansıtmasıdır. Herhangi bir blokzincir uygulamasında olduğu gibi, *SoRBlock*'ta bir blok, iki bölümden oluşur: blok başlığı ve blok gövdesi. Blok başlığı, Blok ID, Previous Block Hash, Merkle Root Hash, Primary-ID ve Timestamp) veri alanlarını içerir. Blok gövdesi, alan denetleyicilerinin hizmet istekleri için alanlar arası uçtan uca yollarını hesaplamak için kullandıkları parça yol işlemlerini tutar.

Alanlar arası yönlendirme, belirli bir istenen hizmetle ilgili mesajlar alındıktan sonra gerçekleştirilir. Bir kullanıcı/müşteri, bir QoS tabanlı alanlar arası hizmet isteğini (kaynak) bir alan denetleyicisine S_Req mesajı aracılığıyla gönderdiğinde, denetleyici, blokzincir defterini kullanarak ve mesajda belirtilen QoS metrikleri ve öncelikleri dikkate alınarak bir alanlar arası uçtan uca yol aramaya başlar. Alanlar arası uçtan uca rota, ağıının bir kenar cihazını (örneğin, kaynak alanı) hedef-alanın bir kenar cihazına birleştirilen parça yollardan oluşur ve bu, blokzincirin parça yol işlemlerini kullanarak soyutlanmış bir ağ katmanı oluşturularak gerçekleştirilir. Kalın mavi parça yollardan oluşan yol, R1-R4-R5-R7-R10-R12-R13, Şekil 11'te bir alanlar arası uçtan uca yol örneğidir. Kaynak alan denetleyicisi, BA S_Req mesajı oluşturduğunda, hizmet isteğinin S_Req mesajında belirtilen gereksinimleri karşılayan bir alanlar arası uçtan uca yol bulamazsa, kullanıcıya Reject cevabı geri gönderir. Kullanıcı/müşteri, BA modülü tarafından oluşturulan S_Res mesajıyla bundan haberdar edilir. Bir uçtan uca rota boyunca herhangi bir alan denetleyicisinden (Şekil 11'te 2, 3 ve 4 numaralı alan denetleyicileri) bir parça yolu olan her bir alan denetleyicisi, BA modülü tarafından oluşturulan P_Req mesajlarını aldıktan sonra belirtilen QoS değerlerini düzenlemek için kaynak alan denetleyicisi tarafından sorgulanır. Kaynak alan denetleyicisi, ilgili alan denetleyicilerinden tüm ilgili alan denetleyicilerinin kaynak alan denetleyicisine Accept cevapları vermesi durumunda kullanıcı/müşteriye Accept yanıtını olan S_Res mesajını gönderir ve hizmet isteğinin karşılanabileceğini ve ağını başlayacağını belirtir. Uçtan uca rota boyunca herhangi bir alan denetleyicisinden Reject yanıtını içeren bir P_Res yanıtı, kaynak alan denetleyicisinin aynı özelliklere ve kriterlere sahip yeni bir alan-içi uçtan uca yol aramasına neden olur.

3.2.2 Kaynak yönlendirme tabanlı alan-içi yönlendirme

Kaynak-Yönlendirme için belirlenen ve kullanılan üç temel strateji bulunmaktadır (Abujoda vd., 2016; Papadopoulos & Papadimitriou, 2019). Kaynak-Yönlendirme'yi

oluşturmanın en temel yaklaşımı, Etiket Sıralama (Label Sequence) ve İşaretçi (Pointer) çözümünü kullanmaktadır. Paket başlıklarında (anahtar portu) etiketlerin dizisi kodlanır ve bir sonraki etikete olan başvuru farklı bir başlık alanında kodlanır. Kaynak/hedef MAC adresi ve IPv6 adresleri, etiketlerin saklanması için iyi adaylardır (ek iletim maliyetleri olmadan). TTL ile Etiket Sıralama yöntemi, bir sonraki etiket için ayrı bir giriş yerine TTL kullanır (önceki yaklaşımın tersine). Son olarak, Anahtar Kimlikleri (Switch IDs) teknigi, benzersiz tanımlayıcı anahtar kimliklerini kullanır ve rotayı bir anahtar kimlikleri dizisi olarak tanımlar. Bu, her ağ cihazında çevredeki cihazların kimliklerini içeren sabit bir eşleştirme tablosu gerektirir. Bir anahtar, zincirdeki kimliğini tanıyalıdıği için bir işaretçiye gerek yoktur. Bu uygulamalar, kendi içsel avantajları ve dezavantajlarına sahiptir.



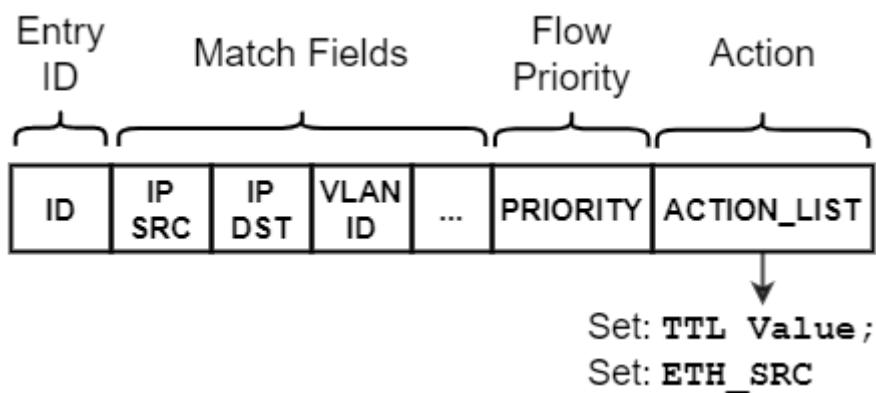
Şekil 12: PIDTTL tabanlı Kaynak-Yönlendirme uygulamasında paket başlıkları

Bu çalışma, etkili başlık alanı kullanımı nedeniyle PIDTTL uygulamasını kullanmaktadır. Kaynak ve/veya hedef MAC adresleri, ilgili veri düzlemi cihazının çıkış portları listesini kodlamak için kullanılır. Özellikle, MAC adresi alanları, her bir düğümün paketi yönlendirmek için kullanacağı çıkış portlarının bir listesiyle doldurulur. TTL alanı, MAC adresindeki bir sonraki port numarasına referans olarak da kullanılır. Şekil 12, paketlerin PIDTTL tabanlı Kaynak-Yönlendirme uygulamasında ağ düğümlerine geldiğinde başlıkta bulunan Next Port Number (Sonraki Port Numarası) (yani düğüm çıkış portları) ve TTL (Time-to-Live - Yaşam Süresi) değerlerini göstermektedir.

- **Yaşam Süresi (Time-to-Live - TTL):** Paketin kaç atlamasının (hop) kaldığını gösteren sekiz bitlik bir başlığıdır. Hedef düğüm, Kaynak-Yönlendirme de yükü işlemek için son atlama olduğunu bilmeliidir. Bu nedenle, TTL bu amaçla oluşturulmuştur ve değeri kaynak düğümde belirlenir ve her bir sonraki atlama ile bir azaltılır.

- **Next Port Number (Sonraki Port Numarası):** Bu, paketin, hesaplanan alan-içi yol boyunca bir düğümdeki sonraki çıkış portunu içeren dört bitlik bir alandır. Düğüm, paket başlığının TTL alanındaki değeri kullanarak ilgili port numarasını okur.

P_Res mesajlarındaki Accept yanıtlarından sonra, Kaynak-Yönlendirme tabanlı alan-içi yönlendirme, alan denetleyicileri tarafından gerçekleştirilir. Bu denetleyiciler, kendi alanlarının yalnızca ilgili giriş sınır düşümlerine hizmet isteği paketleri için bir akış kuralı eklerler. İlgili akış girişi, Kaynak-Yönlendirme için alan içinde paketin geçtiği port numaralarını ve TTL değerini içeren her paket başlığını değiştirir. Örneğin, Şekil 11'teki örnekte, denetleyici 2 (controller 2), kaynak alan (yani, domain 1) denetleyicisine bir P_Res mesajı gönderdikten sonra, alan 2 denetleyicisi (domain 2 controller), talep edilen hizmetin paketlerinin alan 2 (domain 2) boyunca geçmesi için R5'in akış tablosuna bir akış girişi ekler.



Şekil 13: Uçtan uca yolun giriş düşümlerinin eklenen akış girdisi veri yapısı

Şekil 13, uçtan uca yolu üzerindeki alanların giriş ağ düşümlerinin akış tablolarına eklenen akış girişlerinin veri yapısını temsil etmektedir. Alan denetleyici 2 tarafından R5'e eklenen akış girişi, akış girişiyle eşleşen her paketin başlığını, Şekil 11'teki durumda sırasıyla 4 ve $\langle 1, 2, 3, 3 \rangle$ olarak atlama sayısı ve port ID dizisiyle eşleşen bir başlangıç TTL değeri içerecek şekilde değiştirir. TTL değeri, paketin alanında geçtiği her düğüm tarafından MAC adresinde kodlanan sonraki port numarasına işaret edecek şekilde her zamanki gibi azaltılır. Bu işlem her bir alanda kaynak alanı denetleyicisi tarafından hesaplanan uçtan uca alanlar arası yönlendirme yolu üzerinden uygulanır. Paketler uçtan uca yol üzerinden bir sonraki alana geçtiğinde, paketlerin geçtiği alanın giriş düşümü, ilgili alan denetleyicisi tarafından eklenen ilgili akış girişi aracılığıyla paketlerin başlıklarını uygun şekilde değiştirir. Örneğin, denetleyici 3 tarafından R10'a eklenen akış girişi, Şekil 11'teki durumda paketler R7'den R10'a geldiğinde her bir paket başlığını ilk TTL değerini ve port sırasını sırasıyla 2 ve $\langle 2, 4 \rangle$ olarak içerecek şekilde değiştirir.

3.3 Çok Alanlı YTA'da Blokzinciri Tabanlı Yönlendirme Üzerinden Yol Seçim Stratejilerinin Etki Analizi

Bu bölümde, proje kapsamında gerçekleştirilen çoklu YTA'da blokzinciri tabanlı yönlendirme üzerinden yol seçim stratejilerinin etki analizi çalışmasında uygulanan yol seçim tekniklerini açıklanmaktadır. Bir servis isteğinin QoS tabanlı uçtan uca yolunu belirlemek için, QoS gereksinimlerini karşılayarak önerilen blokzincir destekli İSS'ler arası yönlendirme çerçevesi, blokzincirindeki mevcut tüm blokzincir işlemlerini tarar. Bir ajanın yol seçim yaklaşımı, optimum ağ kaynağı kullanımı ve kullanıcı deneyimi kalitesi elde etmek için yol hesaplama algoritması/protokolü kadar önemlidir. Bu amaçla, temel ağ kaynakları ve ölçülebilirlik üzerindeki etkileri araştırmak için (i) İlk Uygun Yol Seçimi (First Feasible Path Selection - FFPS), (ii) Rastgele Uygun Yol Seçimi (Random Feasible Path Selection - RFPS), (iii) Minimum Atlama Yolu Seçimi (Minimum Hop Path Selection - MHPS), (iv) Ağlar Arası Düzeyde BGP-bazlı (Border Gateway Protocol - BGP) FFPS (FFPS_BGP) ve (v) Ağlar Arası Düzeyde BGP-bazlı MHPS (MHPS_BGP) dahil olmak üzere çeşitli uçtan uca yol seçim algoritmalarını incelenmiş ve karşılaştırılmıştır.

İlk Uygun Yol Seçimi (First Feasible Path Selection - FFPS): Blok zincirindeki mevcut tüm işlemleri kullanan bu strateji, Kaynak-Yönlendirme'yi gerçekleştirmek için zamanında hesaplanan ilk olası uçtan uca yolu seçmeyi tercih eder. Eğer $P_{E2E}^{s-d,t} = \{P_i^{s-d,t}, 1 \leq i \leq n\}$ 'i zamanında Kaynak-Yönlendirme'yi karşılayan üç yollar kümesi olarak tanımlarsak, n uygulanabilir yolların sayısıdır ve $P_{E2E}^{s-d,t}$ hiçbir zaman sıfır eşit olmayacağından, bu durumda strateji ilk hesaplanan uygulanabilir $P_1^{s-d,t} \in P_{E2E}^{s-d,t}$ yolunu seçecektir.

Rastgele Uygun Yol Seçimi (Random Feasible Path Selection - RFPS): Bu yöntem, blokzincirde bulunan erişilebilir olan tüm blokzincir işlemleri kullanırken, t zamanında en kısa yol için hesaplanmış olan tüm olası uçtan uca yollar arasından rastgele seçilen yolu tercih eder. $P_{E2E}^{s-d,t} = \{P_i^{s-d,t}, 1 \leq i \leq n\}$ 'i s 'den d 'ye kadar olan ve en kısa yolu ($P_i^{s-d,t}$) zamanında karşılayan üçtan uca yolların kümesi olarak tanımladığımızda, burada (n) uygulanabilir yolların sayısıdır ve $P_{E2E}^{s-d,t}$ hiçbir zaman sıfır eşit olamaz. Bu durumda strateji mevcut tüm uçtan uca yollar arasından ($P_i^{s-d,t} \in P_{E2E}^{s-d,t}$) yolunu seçecektir.

Minimum Atlamalı Yolu Seçimi (Minimum Hop Path Selection-MHPS): Blokzincirde mevcut olan tüm blokzincir işlemlerini kullanan bu seçim yöntemi, kaynaktan hedefe giden tüm olası uçtan uca rotalar arasından en az atlama ile sonuçlanacak yolu belirler. $P_{E2E}^{s-d,t} = \{P_i^{s-d,t}, 1 \leq i \leq n\}$ 'i s 'den d 'ye kadar t zamanında en kısa rotayı sağlayan uçtan uca yolların

kümesi olarak tanımlarsak (n), uygulanabilir yolların sayısına karşılık gelir ve $P_{E2E}^{s-d,t}$ hiçbir zaman sıfıra olamaz. Bu durumda zaman strateji tüm uygulanabilir uçtan uca yollar arasında minimum atlama sayısına sahip uçtan uca yolu yani $\min_{\forall i} L(P_i^{s-d,t})$ seçer. $L(P_i^{s-d,t})$ uçtan uca yolun uzunluğudur (yani atlama sayısı) ve $L(P_i^{s-d,t}) = 1 + \sum_{\forall e_j^{s-d,t} \in P_i^{s-d,t}} 1$, $1 \leq i, j \leq n$ şeklinde tanımlanabilir: Burada $e_j^{s-d,t}$ uçtan uca yol $P_i^{s-d,t}$ üzerindeki bir bağlantıdır.

Ağlar Arası Düzeyde BGP-Bazlı İlk Uygun Yol Seçimi (FFPS_BGP): Bu teknik, ağlar arası düzeyde BGP ile en kısa yolu seçmeyi seçenken (Caesar & Rexford, 2005), blokzincirde ki tüm erişilebilir işlemlerinden yararlanarak en kısa yolu t zamanında karşılamak için hesaplanan uçtan uca yol için ağ içi düzeyden mevcut ilk yolu almayı tercih eder. $I_{E2E}^{s-d,t} = \{I_j^{s-d,t}, 1 \leq j \leq k, j, k \in z^+\}$ ve $P_{E2E}^{s-d,t} = \{P_i^{s-d,t}, 1 \leq i \leq n\}$ uçtan uca en kısa ağlar arası seviyede İSS'ler kümesi olsun (örn. BGP tabanlı en kısa yol ile seçilen İSS'ler) ve s ile d arasındaki t zamanında en kısa yolu sağlayan yollar sırasıyla ($P_i^{s-d,t}$) ve ağlar arası düzeyde BGP protokolü kullanılarak her ağ içi I_j için uygulanabilir yollar olarak $P_{E2E}^{s-d,t} \neq \emptyset$ olsun, strateji daha sonra hesaplanan ilk uygulanabilir yolu seçecektir (her bir $P_1^{s-d,t} \in P_{E2E}^{s-d,t}$ için $I_j \in I_{E2E}^{s-d,t}$).

Ağlar Arası Düzeyde BGP-Bazlı Minimum Atlamalı Yolu Seçimi (MHPS_BGP): Bu seçim stratejisi, en kısa yolun kaynak İSS'den hedef İSS'ye uzanan ağlar arası düzeyde BGP tabanlı en kısa yolu seçenken, blokzincirde ki mevcut tüm blokzincir işlemlerini kullanarak ağ içi düzeyde tüm uygulanabilir yollar arasında minimum atlamaya sahip olan yolu seçer. $I_{E2E}^{s-d,t} = \{I_j^{s-d,t}, 1 \leq j \leq k\}$ ve $P_{E2E}^{s-d,t} = \{P_i^{s-d,t}, 1 \leq i \leq n\}$ uçtan uca en kısa ağlar arası İSS'ler (yani BGP tabanlı en kısa yol ile seçilen İSS'ler) ve s 'den d 'ye t zamanında en kısa yolu sağlayan yollar kümeleri olsun, sırasıyla ($P_i^{s-d,t}$), burada ($P_{E2E}^{s-d,t} \neq \emptyset$), daha sonra strateji uçtan uca yolu seçer (örn, $\min_{\forall i} L(P_i^{s-d,t})$), seçilen her $I_j \in I_{E2E}^{s-d,t}$ için tüm uygulanabilir uçtan uca yollar arasında minimum atlama sayısına sahiptir. $L(P_i^{s-d,t})$ bir uçtan uca yolun uzunluğudur (yani, atlama sayısı) ve $L(P_i^{s-d,t}) = 1 + \sum_{I_p \in I_{E2E}^{s-d,t}} \sum_{\forall e_j^{s-d,t} \in P_i^{s-d,t}} 1$, $1 \leq i, j \leq n$, $1 \leq p \leq k$ ile tanımlanabilir. Bu denklemde $e_j^{s-d,t}$ uçtan uca yol $P_i^{s-d,t}$ üzerindeki bir bağlantıdır.

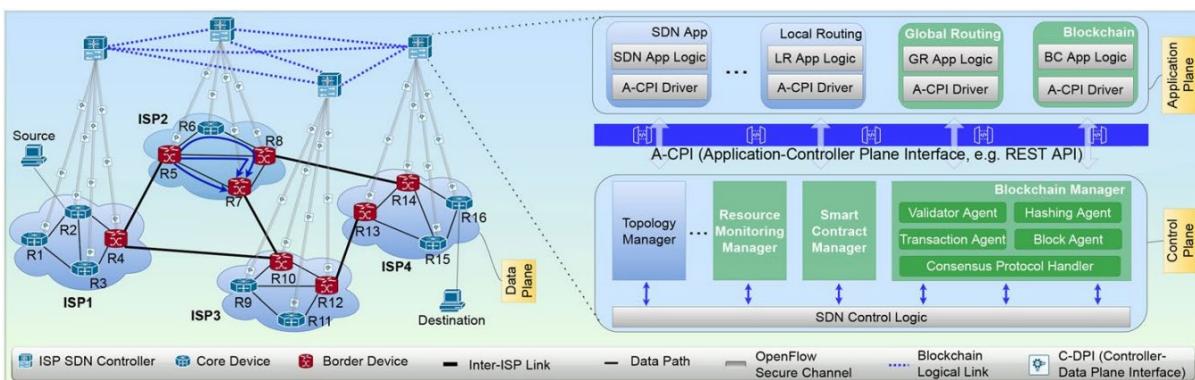
3.4 SmartContractChain (SC²): SDN ve Blokzincir ile Çoklu-İSS QoS Trafik Yönetimi Çerçeveesi

Bu bölüm, akıllı sözleşmelere dayalı bir trafik yönetim çerçevesi olan *SmartContractChain* (SC²) çalışmasının temel varlıklarını, süreçlerini ve işlevsellliğini

açıklamaktadır. 4 adet SDN-bazlı İSS'lerden oluşan, üstünde kontrol düzlemi, veri düzlemi ve bir uygulama düzlemi olan blokzincir destekli SDN ağ mimarisi genel yapısı Şekil 14 de gösterilmektedir. Genel olarak, bir İSS daha spesifik olarak bir otonom sistem veya daha soyut bir terimle bir IP ağları olabilir. Cihazlar ya farklı bir İSS'nin sınır düşümüne (silindirik simgeler) bağlanan bir sınır düşümüdür veya ara bağlantıya sahip olmayan bir çekirdek cihazdır (altigen simgeler). SC² çerçevesinde, her blokzincir düşümü bir İSS SDN denetleyicisine karşılık gelir, kendi blokzinciri örneğini çalıştırır ve diğer blokzincir düşümleri ile tam ağ oluşturur. Bir blokzincir düşümü, güvenli iletiler için bir erişilebilirlik IP adresi ve bir açık/özel anahtar çifti taşır. YTA denetleyicileri, eşleme (peering) ilişkilerini kurmak için açık anahtarlarını değiştirir.

3.4.1 Blokzincir altyapısı

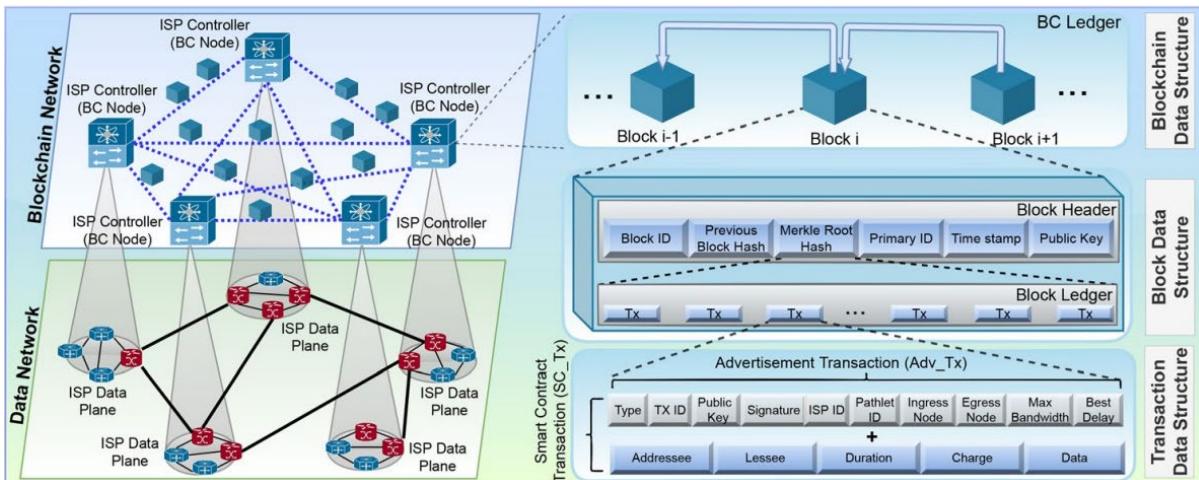
Blokzincirin düşümleri/katılımcıları, uygulamamızda ağ denetleyicileridir. Her denetleyici (yani, blokzincir düşümü), kendi dağıtılmış defter örneğini korur. Kullanıcıya özel blokzincir prensiplerini izleyen kendi özel blokzincir uygulamamız vardır. Blok zincire katılmak ilgili otoritelerden/ajanslardan uygun doğrulama süreçlerini gerektirir. Bu doğrulama süreçleri, İSS'lerin ülkelerinde ilgili kurumlardan, örneğin ABD'de Federal Communications Commission (FCC) veya Internet Assigned Numbers Authority (IANA), kimlik kanıtı sağlamasını gerektirir. Deneylerdeki tüm ağların (yani, İSS'lerin) blokzincire katılmadan önce doğrulandığını varsayıyoruz. SC² çalışmasında kullanılan parça yol modeli önceki bölümlerde açıklanan parça yol yapısı ile aynıdır.



Şekil 14: SC² çerçevesinde denetleyicilerin, özelleştirilmiş akıllı sözleşme yöneticisi, blokzincir yöneticisi ve uygulamaların bir temsili.

3.4.1.1 Blokzincir işlem (transaction) yapısı

Bir blokzincir işlemi (transaction), çeşitli verilerin ve dolayısıyla durum güncellemelerinin bir kaydıdır. Oluşturulduktan sonra, bir blokzincir işlemi, yetkilendirme ve sahiplik kanıtı olarak blokzincir işleminin başlatıcısının imzasıyla dijital olarak imzalanır.



Şekil 15: SmartContractChain (SC²) çerçevesinde kullanılan blokzincir, blok ve blokzincir işleminin (parça yollar için) veri yapıları

SC² çerçevesinde, blokzincir işlemleri, parça yol durumlarına ve Akıllı Kontratların (Smart Contract - SC)'ların İSS'lerdeki parça yol rezervasyonu için gerekli olan QoS değerleri ve veri alanlarına dayanarak blokzincir düğümleri tarafından oluşturulur. Sonuç olarak, giriş-çıkış çiftleri için tüm benzersiz parça yollar, alanlar (domains) için İSS denetleyicileri tarafından hesaplanır. Şekil 15, blokzincir işlemleri için veri yapılarını göstermektedir. Önerilen çerçevede bir blokzincir işlemi iki farklı türde olabilir: Bir İlan Blokzincir İşlemi (Advertisement Transaction - Adv_Tx) ve bir Akıllı Sözleşme Blokzincir İşlemi (Smart Contract Transaction - SC_Tx). Bir Adv_Tx, yolculuk durumunu yansıtır ve şunları içerir: Tür (Type), benzersiz işlem kimliği (Transaction ID - Tx ID), blokzincir işlemini oluşturan İSS denetleyicisinin açık anahtarı (Public Key), blokzincir işleminin dijital imzası (Signature), blokzincir işlemini oluşturanın İSS kimliği (ID), blokzincir işlemi oluşturulan benzersiz parça yol kimliği (Pathlet ID), Giriş (Ingress) ve Çıkış (Egress) düğümleri, mevcut maksimum bant genişliği (Max Bandwith) ve en iyi gecikme (Best Delay). Örneğin, (Adv_Tx, İSS2_3, Abcf..., 0x5ca12d..., İSS2, R5_R7_3, R5, R7, 10, 15) demeti, Şekil 14'teki SC² çerçevesinin blokzincir altyapısındaki İSS2'deki R5-R6-R8-R7 parça yolu için bir Adv_Tx örneğidir. Bir yolculukta bir durum güncellemesi (örneğin, bir bağlantıda bant genişliği artışı, topoloji değişikliği vb.) meydana geldiğinde, ilgili ağ denetleyicisi blokzinciri güncel tutmak için etkilenen parça yollar için güncelleme (yeni) blokzincir işlemlerini oluşturur ve ekler. SC_Tx, bir İSS denetleyicisi tarafından blokzincirde parça yol rezervasyonları için dağıtılan bir SC tarafından tetiklenen bir blokzincir işlemidir. SC_Tx, Adv_Tx'e ek olarak aşağıdaki veri alanlarını içerir: Addressee, SC_Tx tarafından tetiklenen ve Şekil 16'dan Owner alanına karşılık gelen SC'nin blokzincir adresi, Lessee, Lessor-ISP'den (LsR-ISP) bir parça yolu rezerve eden Lessee-ISP'nin (LsE-ISP) denetleyicisinin blokzincir adresi, Duration, parça yol rezervasyon süresi, Charge, parça yol rezervasyon maliyeti, Data ve parça yol üzerindeki

ağ cihazlarının akış tablolarındaki akış kuralı girişlerini tanımlamak ve yüklemek için çeşitli akışa ilişkin bilgiler (örneğin, kaynak/hedef IP'leri, Port Numaraları).

3.4.1.2 Blok modeli

Blok Yapısı: Bir blok, imzaların doğrulanmasından sonra doğrulayıcı bir düğüm tarafından belirli bir veri yapısında birleştirilen bir dizi blozkincir işleminden oluşur. Bir uzlaşma üzerine, blozkincire bir blok eklenir. Bir blok, blozkincirin kriptografik güvenliğini sağlayan mekanizmalardan biri olan önceki bloğa özet (hash) bir değer içerir.

Şekil 15'te blok ve blozkincir veri yapıları gösterilmektedir. Bir blok, blok başlığını ve blozkincir işlemlerinin blok defterini içerir. Blok başlığında Block ID - en güncel blok(lar)ı ayırt etmek için bloğun benzersiz bir kimliği (artan bir sayı), Previous Block Hash - tutarlılığı sağlamak için blozkincirde ki önceki bloğun hashi, Merkle Root Hash - tüm yapıyı temsil eden tekil bir kök hashi oluştana kadar blozkincir işlemlerinin hashlendiği ve birleştirildiği bir veri yapısı, Primary ID - bir blok oluşturma aralığındaki (dönem) benzersiz birincil blozkincir düğüm kimliği, Timestamp - bloğun yayınlandığı zaman ve Public Key – İSS (yani blozkincir düğümü) denetleyicisinin açık anahtarı bloğu oluşturur. SC²'de yeni bir blok mevcut dönemin birincil düğümü tarafından oluşturulur. Birincil sorumluluğu düğüm listesinden aktarmak için bir round-robin sırası izlenir.

Blok Doğrulama Kuralları: Blok doğrulama kuralları, blozkincir işleminin veri içeriğinden bağımsız olan sözdizimsel ve anlamsal doğruluğa odaklanır. Blokların doğruluğunu doğrulamak için uzlaşma protokolü kuralları kullanılır. İlerideki kurallar doğrulanmadan yeni bir blok eklenmeyecektir: (1) bloğun geçerli bir düğüm aracılığıyla oluşturulduğu doğrulanır, (2) bloğun uygun şekilde oluşturulduğu/imzalandığı kontrol edilir, (3) blozkincir işlemi kurallarına uyulduğu doğrulanır, (4) yukarıdakilerin tümü başarılıysa eklenir, aksi takdirde reddedilir.

3.4.2 Blozkincir destekli YTA denetleyicisi

Şekil 14 ayrıca SC² tarafından eklenen denetleyici bileşenlerini de göstermektedir. Denetleyici mimarisi, topoloji yöneticisi, istatistik yöneticisi gibi geleneksel modüllerin yanı sıra yeni modüllere de sahiptir. Birincil blozkincir bileşeni, aracılılarıyla birlikte Blokchain Manager (BM)'dır. BM, blozkincir ile ilişkili blozkincir faaliyetlerini gerçekleştirir. Validator Agent, blok doğrulama kurallarına göre blok onaylamasını gerçekleştirir. Hashing Agent, blozkincir işlemleri ve bloklar üzerinde karma faaliyetleri yürütür. Transaction Agent, blozkincir işlemleri oluşturken Block Agent, blokları oluşturur. Consensus Protocol Handler blozkincir ağında kullanılan uzlaşma kurallarının uygulanmasını sağlar. Smart Contract Manager- SCM dağıtım, yürütme, bakım ve benzeri akıllı sözleşmelerle ilgili işlerin yürütülmesinden sorumludur.

Aşağıdaki bölümde SCM modülünün çalışma mantığı hakkında daha ayrıntılı bilgi verilmektedir. Resource Monitoring Manager- RMM ağdaki değişiklikleri fark etmek için ağ kaynaklarını sürekli olarak gözlemlemektedir. Herhangi bir değişiklik olması durumunda, BM'yi değişiklikleri yansıtan (yeni) blokzincir işlem(ler)i oluşturması için tetikler. Denetleyici yeni ağ uygulamaları ile de güçlendirilmiştir. Global Routing Agent- GRA, denetleyiciye bir çapraz-İSS hizmet talebi ulaştığında çapraz-İSS trafik yönlendirme görevlerini yerine getirir. Blockchain Application (BA) blokzincir ağına/ağından bloklar yayınlar, kabul eder ve servis talep mesajlarını yönetir.

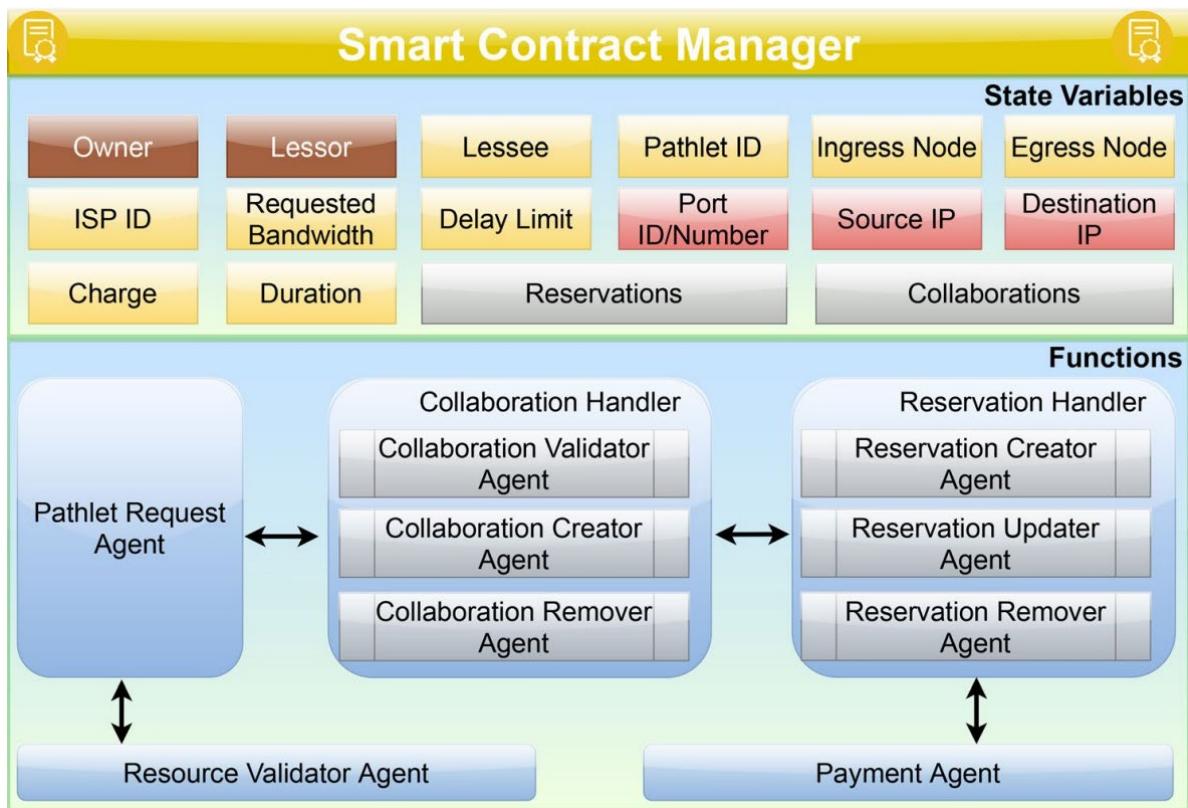
3.4.3 Akıllı sözleşme (smart contract) modeli

SCM, tüm parça yol rezervasyon görevleriyle başa çıkmak için SC²'nin temel yardımcı modülüdür. Bu modül, İSS ağından kaynak rezervasyonları için oluşturulan sözleşmelerin durumlarını izler ve ilgili faaliyetleri yerine getirir. Şekil 16 da SCM durum değişkenleri ve fonksiyonları gösterilmektedir.

3.4.3.1 State (Durum) Değişkenleri

Durum değişkenleri parça yol rezervasyon bilgilerini saklar. Bu değişkenler, ilgili SC tetiklenirken SC_Tx'ten elde edilirler. Sahip değişkeni (owner variable) SC'nin blokzincir adresini saklarken, Kiralayan değişkeni (lessor variable) parça yol ve diğer İSS'lere rezerve eden İSS denetleyicisinin blokzincir adresini saklar. Sarı renkle gösterilen değişkenler, İSS'nin ağından ayrılacak parça yollarla ilgili bilgileri tutar. Bu bilgiler SC_Tx'den alınır ve SC tetiklendiğinde değişkenlere atanırlar. Kiralayan (Lessee), başka bir İSS'den bir parça yol rezerve eden İSS denetleyicisinin blokzincir adresini saklar. Pathlet ID, bir İSS'den rezerve edilen parça yolun ID'sini saklar. Giriş Düğübü (Ingress Node) ve Çıkış Düğübü (Egress Node), sırasıyla ayrılacak parça yolun başlangıç ve bitiş cihazlarını saklar. İSS ID, LsE-ISP'nin benzersiz ID'sini saklar. Talep Edilen Bant Genişliği ve Gecikme Limiti, LsE-ISP tarafından rezerve edilecek parça yol talep edilen bant genişliği miktarını ve talep edilen gecikme kaynaklarının üst sınırını sırasıyla bura saklanır. Ayrıca, Süre parça yol rezervasyonunun beklenen uzunluğudur, Ücret ise maliyetini saklar. Pembe renkle gösterilen değişkenler, İSS denetleyicileri tarafından ayrılmış parça yolu kurmak için veri katmanı cihazlarına yüklemek üzere hazırlanan ilgili akış kuralı girişlerini akışla ilgili verilerdir. Bir parça yolun çıkış düğümünün Port ID/Number rezervasyon için kullanılır. Ayrıca, Kaynak IP (Source IP) ve Hedef IP (Destination IP) sırasıyla servis talebinin kaynak ve hedef noktasıdır. Son olarak, Rezervasyonlar (Reservations) parça yollar için devam eden rezervasyonları saklarken,

Ortaklıklar (Collaborations) İSS'nin parça yollarını rezerve edebileceğи diğer İSS'lerin listesini tutar.



Şekil 16: İSS'lerdeki blokzincir destekli SDN denetleyicilerinin Akıllı Sözleşme Yöneticisi (Smart Contract Manager - SCM) modülünün yapısı

3.4.3.2 Fonksiyonlar

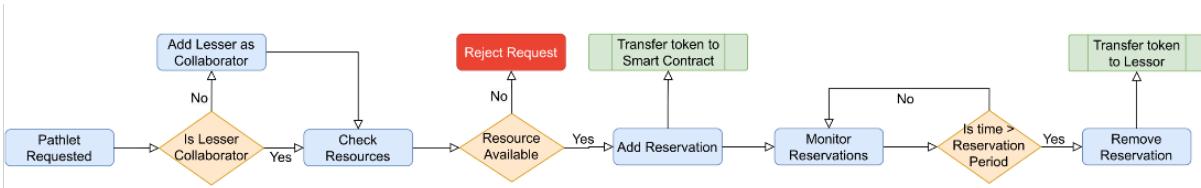
SCM'de Pathlet Talep Aracı (Pathlet Request Agent-PRA), SC'yi başka bir İSS'den bir parça yol rezerve etmesi için tetiklerken SC_Tx'te sağlanan parça yol ile ilgili verileri girdi olarak alır. Bu modül sözleşmeyi yaratan İSS'den bir parça yol rezerve etmek isteyen İSS (denetleyici) tarafından çağrırlı. İlk olarak, PRA parça yol blokzincir işlemi içeriğini doğrular. Eğer başarılı olursa, istenen QoS kaynaklarının parça yol üzerinden hala yerine getirilebilir olup olmadığını kontrol eder. Eğer istek gerçekleştirilebilir ise, fonksiyon bir rezervasyon nesnesi oluşturur (reservation object) ve bunu Rezervasyonlar değişkenindeki rezervasyonlar listesine ekler ardından ücreti parça yolu rezerve eden İSS'den SC hesabına aktarır. Kaynak Doğrulayıcı Aracı (Resource Validator Agent - RVA), talep edilen QoS kaynaklarının parçalı üzerinden hala karşılanabilir olup olmadığını kontrol eder. Eğer talep karşılanabilir değilse, rezervasyon işlemi sonlandırılır. Bu tür bir doğrulamanın amacı, ilgili yolun, diğer parça yollarla ilgili en son blokzincir işleminde bildirilen QoS değerlerini hala sunabilmesini sağlamaktır. Blokzincirdeki (işlem) değişiklikleri, ağ yollarından durum değişiklikleri kadar anlık olmadığından, ağ

dinamikleri bir parça yolun QoS değerlerinin bildirilenlerden farklımasına neden olabilir. Rezervasyon İşleyicisi (Reservation Handler - RH), üç alt bileşen aracılığıyla bir İSS'deki parça yol rezervasyonlarını tutar. Rezervasyon Oluşturma Aracı (The Reservation Creator Agent - RCA) bir rezervasyon nesnesi oluşturur ve bunu Rezervasyonlar durum değişkeninde tutulan rezervasyonlar listesine ekler. Rezervasyon nesnesi, LsE-ISP'nin blokzinciri adresini ve İSS ID, parça yol bilgilerini (ID, Ingres ve Engress node), parça yol için talep edilen süre, bant genişliği ve gecikme değerlerini, hizmet için kaynak ve hedef IP'leri, parça yol için ödenecek ücreti ve nesneyi listeye ekleme süresini içerir. Nesne listeye eklendikten sonra, ücret LsE-ISP'den SC adresine aktarılır. Rezervasyon Güncelleyici Aracı (Reservation Updater Agent-RUA), Rezervasyonlar durum değişkenindeki rezervasyonlar listesinden bu rezervasyon nesnesini günceller. Bu güncellemeler, mevcut bir rezervasyonun süresinin uzatılması, parça yol üzerinde önceden kararlaştırılmış QoS değerlerinin güncellenmesi gibi çeşitli nedenlerden kaynaklanabilir.

Rezervasyon Kaldırma Aracı (Reservation Remover Agent - RRA) rezervasyon nesnesini Rezervasyonlar durum değişkenindeki rezervasyon listesinden kaldırır. Bu kaldırma işlemi, rezervasyon süresinin sona ermesi veya LsR-ISP'nin rezervasyon için tam ödeme koşulu uyarınca karar vermesinden kaynaklanabilir. Ortalık Yöneticisi (Collaboration Handler - CH), bir İSS'nin diğer İSS'lerle parça yol rezervasyonu amacıyla yaptığı ortaklıklarını sürdürmekten sorumludur. Bu yönetici ortalık ile ilgili görevleri üç alt bileşen kullanarak gerçekleştirir. Ortaklık Doğrulama Aracı (Collaboration Validator Agent - CVA), bir parça yolu rezerve eden bir İSS'nin İSS'nin ortakları listesinde olduğunu doğrular. Ortaklık Oluşturma Aracı (Collaboration Creator Agent - CCA), bir İSS'nin ağından bir parça yolu rezerve etmesine izin verdiği ortak İSS'leri Collaborations durum değişkeninde tutulan ortaklar listesine ekler. Bir İSS'nin bir parça yol rezerve edebilmesi için başka bir İSS'nin ortaklar listesinde olması gereklidir. Ortaklık Kaldırma Aracı (Collaboration Remover Agent-CRA), ağ politikalarına bağlı olarak bir ortak İSS'yi İSS'nin ortaklar listesinden kaldırır. Son olarak, Ödeme Aracısının (Ödeme Aracı-PA) ana görevi, parça yolların rezervasyonu karşılığında LsR-ISP'ye ödeme yapmaktadır. Rezervasyon listesindeki bir rezervasyonun süresi dolduğunda, ilgili rezervasyonun ücretini SC'den LsR-ISP'ye aktarır ve karşılığında rezervasyonu, rezervasyon listesinden kaldırır.

3.4.4 SmartContractChain (SC²) çerçevesinin iş akışı

Şekil 17'de SC² çerçevesinin yol seçimi, rezervasyon ve kaynak tahsis için durum geçişleri diyagramı gösterilmektedir. Şekil 18, servis taleplerini yerine getirmek için SC²nin trafik yönetimi faaliyetlerine dahil olan süreçleri ve aktörleri göstermektedir.



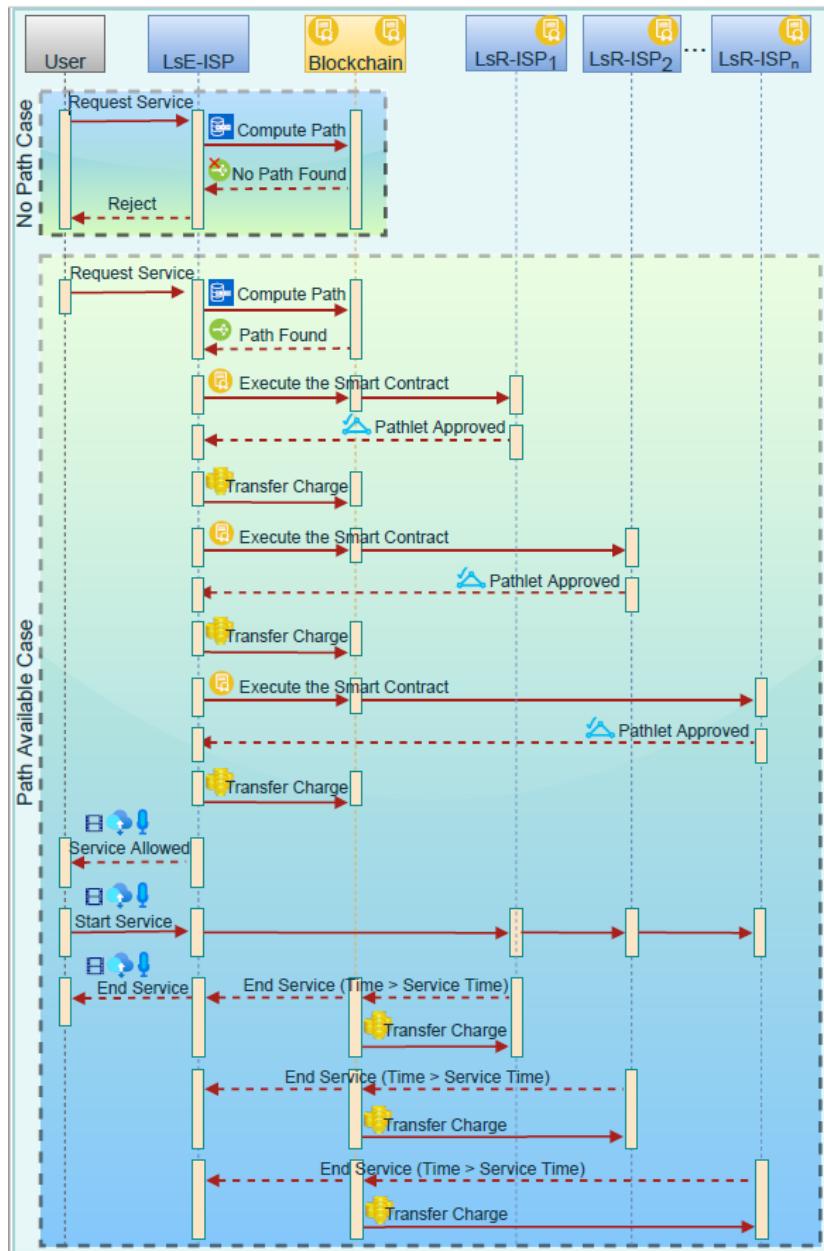
Şekil 17: SC² Durum Geçiş Diyagramı

Bir servis talebi iki olası sonuç verebilir: Yol Yok (No Path) ve Yol Mevcut (Path Available) durumları. İlk durum hizmet reddiyle sonuçlanırken, ikinci durum çoklu İSS ortamında servis teklifiyle sonuçlanır. Bir kullanıcı abone olduğu İSS'nin denetleyicisine (kaynak-İSS/LsE-ISP denetleyicisi) QoS tabanlı bir İSS'ler arası servis talebi gönderir. Kaynak-İSS (yani LsE-ISP) denetleyicisi, kullanıcının hizmet talebinde belirtilen QoS parametrelerine göre blokzincir defterinde bulunan Adv_Tx işlemlerinden yararlanarak tam bir yol hesaplamaya başlar. Tam yol, kendi ağının bir sınır düğümünden hedef-İSS'nin bir sınır düğümüne, blokzincirden gelen parça yol blokzincir işlemlerinden yararlanılarak soyutlanmış bir katman ağı üzerindeki yollardan oluşur. Denetleyiciler, servis talebi için tam yolu hesaplarken en güncel blokzincir işlemlerden yararlanır.

Yol Yok (No Path) Durumu: Kaynak-İSS/LsE-ISP denetleyicisi servis talebinde belirtilen QoS parametrelerini karşılayan tam bir yol bulamazsa, LsE-ISP denetleyicisi servis talebini reddeder.

Yol Mevcut (Path Available) Durumu: Diğer İSS'lerden gelen birden fazla parça yoldan oluşan bir uçtan uca yol bulunursa, LsE-ISP denetleyicisi, hesaplanan uçtan uca yol üzerinde parça yol rezervasyonu için PRA'ları çağrıarak İSS'lerin (yani LsR-ISP'lerin) ilgili SC'lerini tetiklemek için SC_Tx'ler oluşturur. Tetiklenen SC ile hesaplanan uçtan uca yol üzerindeki bir LsR-ISP, CH'nin CVA'sını kullanarak Collaborations değişkeninde tutulan LsE-ISP'nin ortaklar listesinde (örnek bir ortak listesi için bkz. Şekil 19) olup olmadığını kontrol eder. Eğer LsE-ISP listede yer alıysa, servis kurulum süreci devam eder. Eğer listede yer almıyorsa, LsR-ISP ağ politikalarına bağlı olarak LsE-ISP'yi CCA ile listeye ekleyebilir ve servis kurulum süreci buna göre ilerler. Ortaklık doğrulaması tamamlandıktan sonra RVA, ilgili parça yolun daha önce açıklanlığı gibi güncel olmaması durumunu önlemek için talep edilen QoS değerlerini hala sağlayıp sağlayamayacağını doğrular. Eğer Talep edilen kaynaklar mevcut değilse, parça yol talebi reddedilir. Eğer kaynaklar hala mevcutsa, LsR-ISP, RH'nin RCA'sı tarafından bir rezervasyon (nesne) oluşturulur ve Reservations değişkeninde tutulan rezervasyonlar listesine ekler (örnek bir rezervasyon listesi için bkz. Şekil 19a). Rezervasyon eklendikten sonra, rezervasyon ücreti blokzincir üzerinden LsE-ISP'nin cüzdanından LsR-ISP'nin SC cüzdanına

aktarılır. Yukarıdaki işlemlerin, her biri için bir SC_Tx oluşturup ekledikten sonra LsE-ISP tarafından hesaplanan uçtan uca yol üzerinde tüm LsR-ISP'ler tarafından gerçekleştirilir.



Şekil 18: Bir servis talebi senaryosu ele alınırken SC²'nin işleyişinde yer alan süreçler ve aktörlerin çalışma modeli

LsR-ISP'lerden hiçbir parça yol talebini reddetmezse, hizmetin tam yol üzerinden, rezervasyonun başlamasına izin verilir. SCM, İSS'sinin Reservations değişkeninde bulunan tüm parça yollar için yapılan bütün rezervasyonları izler. Rezervasyon süresi sona erdiğinde (Süre-Duration- alanından), RRA bu rezervasyonu PA rezervasyon listesinden kaldırır ve ilgili rezervasyon için ücreti SC'nin cüzdanına aktarır.

Lessee	Pathlet ID	Ingress Node	Egress Node	ISP ID	Requested Bandwidth	Delay Limit	Port ID	Source IP	Destination IP	Charge	Duration	Time Added
0x1a3bb...	R5_R7_1	R5	R7	1	12	10	1	1.1.1.1	4.4.4.4	4	15	2021-05-07-16:19
0x3c4d5...	R5_R7_1	R5	R7	3	8	10	1	3.3.3.3	1.1.1.1	2	10	2021-05-07-15:38
0x4a22b...	R7_R8_2	R7	R8	4	25	15	1	4.4.4.4	3.3.3.3	5	20	2021-05-07-14:23
0x1a3bb...	R5_R8_1	R5	R8	1	16	10	1	1.1.1.1	4.4.4.4	3	15	2021-05-07-15:49
...

(a) Rezervasyon Listesi

Lessee (Address)	ISP ID
0x1a3bb	1
0x3c4d5	3
0x4a22b	4
...	...

(b) Ortakların Listesi

Şekil 19: Bir akıllı sözleşmenin sırasıyla Reservations ve Collaborations değişkenlerinde tutulan örnek bir rezervasyon ve ortaklar listesi

Benzer şekilde, uçtan uca yol üzerinden İSS'lerden gelen parça yol rezervasyonları, mevcut zaman rezervasyon süresini geçtiğinde kaldırılır ve ilgili ücretler PA tarafından SC'lerin cüzdanlarından İSS'lerin cüzdanlarına aktarılır. Veri katmanı cihazlarının akış tablolarına uçtan uca parça yollar üzerinden akış kurallarının yüklenmesi gibi ağ ile ilgili kurulumlar, parça yol rezervasyon taleplerini kabul etmeleri halinde SC_Tx'lerde sağlanan akış ile ilgili bilgiler uyarınca ilgili LsR-ISPs denetleyicileri tarafından yönetilir.

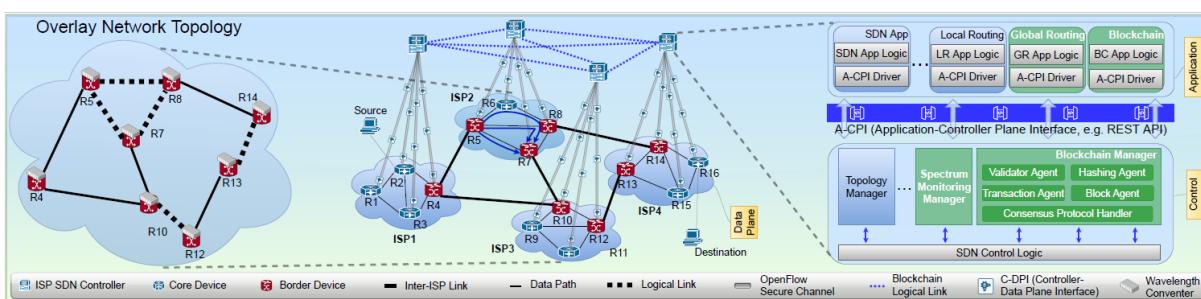
3.5 Yazılım Tanımlı Optik Ağlar (YTOA)'da Blokzincir ile Geliştirilmiş Çoklu-İSS Spektrum Atama Çerçeve: SpectrumChain

Bu kısımda, proje çalışmaları çerçevesinde geliştirdiğimiz Yazılım Tanımlı Optik Ağlar - YTOA (Software Defined Optical Networks – SDON)'larda QoS tabanlı bir yönlendirme modeli ile blokzincir teknolojisini birleştiren İSS'ler arası spektrum atama ve yönlendirme çerçevesi olan *SpectrumChain* (SC) anlatılmaktadır. Bunun için öncelikle, blokzincir ile güçlendirilmiş SDON'larda önerilen uyaranabilir ve verimli spektrum yönetim çerçevesini kullanacak bir ağ modeli tanımlanmıştır.

3.5.1 Sistem Modeli ve Problem Tanımlamaları

Dört SDON tabanlı İSS'den oluşan ağ konsepti Şekil 20'de gösterilmektedir. Her SDON bazlı İSS, blokzincir ağının ağ durumuyla ilgili tüm işlemlerini ve bloklarını senkronize tutmak için birbirleriyle iletişim kurar. Bu çalışmada, ağ cihazı dalga-boyu dönüştürme yetenekleri açısından üç senaryo incelenmektedir:

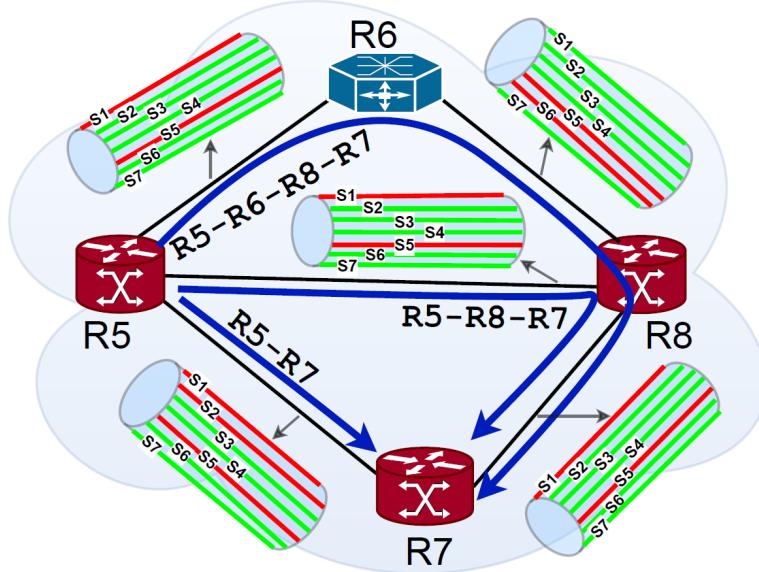
- Hop-by-Hop Dalgaboyu Anahtarlama (Hop-by-Hop Wavelength Switching - HWS): Sadece iç bağlantınlara sahip iç (altıgen mavi nesneler) ve çeşitli İSS'ler aracılığıyla birbirine bağlı bağlantınlara sahip sınır/kenar düğümleri (silindir kırmızı nesneler) dahil olmak üzere tüm ağ cihazları, veri aktarırken dalga boylarını değiştirebilir,
- Sınır Dalgaboyu Anahtarlama (Border Wavelength Switching - BWS): Farklı bitişik ve sürekli spektrumlar (veya alt taşıyıcılar) kullanarak bir İSS'den diğer İSS'ye veri iletilken yalnızca uç ağ cihazları dalga-boyu dönüştürücülerle güçlendirilir,
- Dalgaboyu Anahtarlama Yok (No Wavelength Switching - NWS): Hiçbir cihaz alan-içi veya alanlar arası veri iletilken dalga-boyu dönüşümü yapamaz.



Şekil 20: YTOA İSS ağ modelinin ve onun soyutlanmış yer paylaşımı ağıının bir örneği

Şekil 20'de gösterilen ve *SpectrumChain* sisteminde kullanılan SDON denetleyicisi, (Karakus vd., 2021)'de ayrıntılı olarak açıklanan blozkincir özelliklerine sahip bazı benzer modüllere ve ağ uygulamalarına dayanmaktadır. Blozkincir Yöneticisi (BM), Spektrum İzleme Yöneticisi (SMM), Küresel Yönlendirme Aracı (GRA) ve Blozkincir Uygulaması (BA) tipik denetleyici modüllerinden farklı yeni denetleyici modülleridir. BM modülü, Validator Agent, Hashing Agent, Consensus Protocol Handler, Transaction/Block Agent ve Mining Agent gibi alt bileşenleri aracılığıyla blozkinciri ile ilgili tüm görevleri yerine getirir. Validator agent, diğer denetleyicilerden gelen blokları doğrulamak için blozkinciri bağlamında blok doğrulama kriterlerini kullanır. Bir diğer BM bileşeni, blozkinciri alanında hashing işlemleri ve blokları gerçekleştiren hashing agent; Transaction/Block Agent ise blozkinciri ağını sürdürmek için işlemler veya bloklar üretir. Consensus Protocol Handler, blozkinciri alanında kullanılacak mutabakat mekanizmasını belirler ve Mining Agent, blozkinciri kullanım durumlarına ve uygulanan konsensüs protokolüne bağlı olarak denetleyicide madencilik işlerini yürütür. Spektrum İzleme Yöneticisi (SMM) modülü, SC mimarisindeki SDON denetleyicisine takılan özel bir modüldür. SMM modülü, İSS-içi fiber optik bağlantılar arasında bitişik ve sürekli spektrumların fizibilitesini hesaplamak için ağ kaynaklarını sürekli olarak izler. İSS'ler spektrum tahsislerini değiştirdiklerinde, SMM, blozkincir alanında gerekli işlemleri güncelleten veya oluşturan BM bileşenini uyarır. Denetleyiciye İSS'ler arası bir hizmet talebi geldiğinde GRA

modülü, uçtan uca fiber optik İSS'ler arası bağlantıyi belirlemek için bloklardaki işlemlerin giriş, çıkış (yani sınır düğümleri) ve QoS alanlarını kullanır. BA modülü blok zinciri ağına blok gönderme ve alma işlemlerinin yanı sıra talep mesajlarını da yönetir.



Şekil 21: İSS2 ağında örnek bağlantı spektrumu kullanılabilirliği olarak R5 ve R7 ağ cihazları arasındaki optik parça yolları (R5-R7, R5-R8-R7 ve R5-R6-R8-R7).

Parçalı Optik Yol (Pathlet). Parçalı optik yol, herhangi iki farklı İSS sınır düğümü arasında benzersiz bir yoldur. Bir parça yolun üç noktaları giriş ve çıkış düğümleri olarak adlandırılır (yani, bir İSS'ye giren veya İSS'den çıkan cihazlar). Şekil 21'de gösterildiği gibi, R5-R7, R5-R8-R7 ve R5-R6-R8-R7 fiber optik yolları (bağlantıların çift yönlü olduğu varsayılırsa) İSS2'deki R5 ve R7 sınır ağı düğümleri arasındaki yollardır. Kırmızı ve yeşil spektrumlar, bir optik fiber bağlantısındaki dolu ve kullanılabilir spektrumları temsil etmektedir.

Hizmet Talebi. Bir Hizmet Talebi, aynı veya diğer İSS'lerdeki son kullanıcılar (veya kuruluşlar) arasında belirli QoS parametrelerine sahip trafik ihtiyacını (yani, bitişik ve sürekli bir spektrum talebi) iletir.

SpectrumChain; (i) benzersiz bir hizmet tanımlayıcı numarası, kaynak ve hedef IP adresleri ve uçtan uca optik fiber yolu üzerinden talep edilen spektrum talebini içeren hizmet talep mesajını, (ii) benzersiz bir parça yol talep tanımlayıcı numarası, bir İSS'den parça yolun giriş ve çıkış düğümleri, talep edilen spektrum sayısı, kaynak ve hedef ana bilgisayarların IP adreslerini içeren parça yol talep mesajını, (iii) parça yol yanıt tanımlayıcı numarasına ve İSS denetleyicinin talep edilen parça yolu sağlama kararına (Kabul veya Ret) sahip parça yol yanıt mesajını, (iv) benzersiz bir hizmet tanımlayıcı numarasına ve kaynak-İSS denetleyicinin gelen parça yol yanıt mesajını değerlendirdikten sonra talep edilen hizmeti sağlama kararına sahip hizmet yanıt mesajı kullanmaktadır.

İSS'ler arası QoS tabanlı Optik Yönlendirme. Şekil 20'de gösterilen ağ topolojisine benzer

şekilde $1 \leq i \leq n$ olan $N_i(V_i, E_i)$ grafi olarak gösterilen n optik ağ kümesi göz önüne alındığında, $V = \{V_1, V_2, \dots, V_n\}$ ve $E = \{E_1, E_2, \dots, E_n\}$ sırasıyla tüm ağlardaki tüm düğümlerin ve bağlantılarının kümeleri olsun. $V_i = \{v_{i,1}, v_{i,2}, \dots, v_{i,v}\}$, $E_i = \{e_{i,1}, e_{i,2}, \dots, e_{i,m}\}$ ve *V_i , ${}^*V_i \subseteq V_i$ olan N_i ağındaki sırasıyla tüm düğümlerin, bağlantılarının ve sınır düğümlerinin kümelerini gösterir. $P_i = \{P_i^1, P_i^2, \dots, P_i^j\}$ N_i 'deki tüm farklı yol ağlarının kümesi olsun; burada P_i^j , E_i 'den gelen bağlantı serilerinden oluşur ve $P_i^j = \{e_{i,1}^j, e_{i,2}^j, \dots, e_{i,m}^j\}$ şeklinde gösterilir. Bir i ağında $m e_i^j$ olarak temsil edilen j yolu üzerindeki herhangi bir m bağlantısının k spektrumu vardır ve $m e_i^j = \{m s_i^j, m s_i^j, \dots, m s_i^j\}$ ile temsil edilir; bu da $m S_i^j = \{m s_i^j, m s_i^j, \dots, m s_i^j\}$ ile temsil edilen i ağındaki j yolu üzerindeki m bağlantısı üzerindeki tüm spektrumların kümesine eşittir. Ayrıca, ${}^*S_i^j$, i ağındaki j yolu üzerindeki m bağlantısındaki mevcut spektrumlar kümesini göstergesin. $N({}^*V, {}^*E)$, N_i ağlarından soyutlanmış bir katmanlı ağ topolojisini göstergesin; burada ${}^*V = \bigcup_{i=1}^n {}^*V_i$, ağların tüm sınır düğümlerinin kümesidir ve *E , bu sınır düğümleri arasındaki (mantıksal) bağlantılar kümesidir.

Varsayımlar 1: İki $m e \in E$ ve $n e \in E$ bağlantı verildiğinde, $|m S| = |n S|$ burada $m S$ ve $n S$ sırasıyla $m e$ ve $n e$ bağlantılarındaki spektrum kümeleridir ve $|\cdot|$ kümeye kardinalitesini ifade eder.

Tanım 3: Spektrum Bitişikliği. Bir i ağındaki bir j yolu üzerindeki bir m bağlantısında iki spektrum verildiğinde, yani $x s_i^j$ ve $y s_i^j$, burada $1 \leq x, y \leq k$, $x - y = 1$ ise, $x s_i^j$ ve $y s_i^j$ bitişik spektrumlardır.

Tanım 4: Bitişik Spektrum Kümesi. Bir i ağında bir j yolu üzerindeki bir m bağlantısında bir spektrum kümesi verildiğinde, yani $m S_i^j = \{m s_i^j, m s_i^j, \dots, m s_i^j\}$ burada $1 \leq a \leq k$, eğer tüm spektrumlar bitişikse, $m S_i^j$ bir ardışık spektrum kümesidir.

Tanım 5: Spektrum Süreklliliği. Bir i ağındaki j yolu üzerinde m ve n komşu bağlantılarında iki spektrum verildiğinde, yani $x s_i^j$ ve $y s_i^j$, $x = y$ ve $m \neq n$ ise süreklilik özelliğine sahip oldukları söylenir.

Bir ağdaki her bağlantının herhangi bir zamanda erişilebilir veya hizmet taleplerine atanmış k spektrum taşıdığını varsayalım. Bir optik ağıda $m \leq k$ olmak üzere, m spektrum talep eden bir hizmet talebine spektrum atanırken, üçlü kısıtın (CCD - spektrum Süreklliliği, spektrum Bitişikliği ve spektrum Ayrıklığı (yani, örtüşmeyen) anlamına gelir) karşılanması gereklidir. İSS'ler arası spektrum kısıtlı yol problemi (IN-ScP) aşağıdaki gibi (çoklu) kısıtlı bir yol problemi olarak karakterize edilebilir.

Tanım 6: ISS'ler arası Spektrum Kısıtlı Yol Problemi (IN-ScP) $N_i(V_i, E_i)$ ağlarından soyutlanmış bir $N(^V, ^E)$, ağı düşünün. Her bir $e \in E$ bağlantı, bir anda mevcut veya tahsis edilmiş k spektrum tarafından belirlenir. CCD kısıtları göz önüne alındığında problem, bir s kaynak düğümünden d hedef düğümüne (burada $s \in ^V_i$, $d \in ^V_j$ ve $^V_i \neq ^V_j$) öyle bir yol bulmaktır ki, yol üzerindeki tüm bağlantılarda $s \leq k$ olmak üzere s spektrum talep eden hizmet talebi için tahsis edilen spektrumlar süreklilik, bitişiklik ve ayrınlığa sahip olsun. CCD kısıtlarını karşılayan bir yol (spektrum kısıtlı) uygulanabilir yol olarak adlandırılır. $N(^V, ^E)$ grafında kısıtlamaları karşılayan birden fazla yol olabilir. Tanım 4'e göre, bu yollardan herhangi biri IN-ScP probleminin bir çözümüdür.

Tanım 7: Bağlantı Spektrum Uzunluğu (LSL). Bir i ağında ${}_m e_i^j$ olarak temsil edilen bir j yolu üzerindeki bir bağlantı göz önüne alındığında, ${}_m e_i^j$ için Bağlantı Spektrum Uzunluğu bağlantıdaki spektrum sayısıdır ve aşağıdaki gibi verilebilir:

$$L({}_m e_i^j) = \sum_{\forall {}_m s_i^j \in {}_m e_i^j} 1$$

Tanım 8: Pathlet Uzunluğu (PaL). i ağında P_i^j olarak temsil edilen bir j parça yolu verildiğinde, Pathlet P_i^j için uzunluk, parça yol üzerindeki bağlantı sayısıdır ve aşağıdaki gibi verilebilir:

$$L(P_i^j) = \sum_{\forall {}_m e_i^j \in P_i^j} 1$$

Tanım 9: Bağlantı Spektrum Matrisi (LSM). Bir i ağında ${}_m e_i^j$ olarak temsil edilen bir j yolu üzerindeki bir bağlantı verildiğinde, ${}_m e_i^j$ için Bağlantı Spektrum Matrisi $1 \times k$ matrisidir ve aşağıdaki gibi temsil edilir:

$${}_m e_i^j M = [{}^1 s_i^j \quad {}^2 s_i^j \quad \dots \quad {}^k s_i^j]$$

$${}^k s_i^j = \begin{cases} 1, & {}^k s_i^j \text{ uygun ise} \\ 0, & {}^k s_i^j \text{ dolu ise} \end{cases}$$

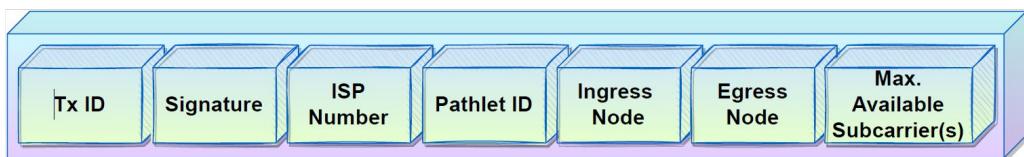
Tanım 10: Pathlet Spektrum Matrisi (PSM). Bir i ağında P_i^j olarak temsil edilen bir parça yolu verildiğinde, P_i^j için Pathlet Spektrum Matrisi bir $m \times k$ marix'tir ve aşağıdaki gibi temsil edilir:

$$M^{P_i^j} = \begin{bmatrix} {}_1e_i^j M \\ {}_2e_i^j M \\ \vdots \\ {}_m e_i^j M \end{bmatrix} = \begin{bmatrix} {}_1s_i^j & {}_2s_i^j & \dots & {}_k s_i^j \\ {}_1s_i^j & {}_2s_i^j & \dots & {}_k s_i^j \\ \vdots & \vdots & \ddots & \vdots \\ {}_m s_i^j & {}_m s_i^j & \dots & {}_m s_i^j \end{bmatrix}$$

3.5.2 SpectrumChain (SC) Mimarisi

Bu bölümde, blokzincir ve optik ağ faaliyetlerinin yürütülmesine yönelik temel kavramları, prosedürleri ve özelliklerini tanımlamak için *SpectrumChain* çerçevesinin mimarisi sunulmaktadır.

Her optik İSS ağ denetleyicisi, *SpectrumChain* mimarisinde kendi blokzincir örneklemesini sürdürmek için bir blokzincir düğümüne başvurur. Criptografik faaliyetler için gereken açık ve özel anahtarlar kümesi bir İSS denetleyicisi (veya blokzincir düğümü) tarafından tutulur. Sonuç olarak, İSS denetleyicileri, ayırt edici blokzincir düğümü kimlikleriyle (yani İSS Numaraları (İSSN)) birlikte açık anahtarlarını değişim tokusu eşleriyle ilişkili kurar. Denetleyicilerden gelen criptografik olarak imzalanmış optik ağ verileri (İSSN, sınır cihazlarının bir listesi vb. gibi) açık anahtarlarını içermelidir ve İSS'ler arası ağdaki denetleyiciler diğer denetleyicilerden tam verileri ister.



Şekil 22: SpectrumChain'de ki transaction (işlem) veri yapısı

SpectrumChain, İSS denetleyicileri aracılığıyla QoS metriklerini içeren optik yollardan kullanım durumlarına göre farklı veriler içeren işlemler üretir. Sonuç olarak, İSS denetleyicileri kendi İSS-içi ağlarının farklı sınır düğümü çiftleri arasındaki bireysel parça yollarını belirler ve bu parça yolları blokzincir defterine ayrı işlemler olarak ekler. SC'deki bir işlemin yapısı Şekil 22'te tanımlanmıştır. İşlemin benzersiz kimliğini, İSS denetleyicisi tarafından imzalanan işlemin criptografik dijital imzasını, benzersiz İSS numarasını, farklı bir yoluun benzersiz kimliğini, bir İSS'deki bir optik yoluun başlangıç ve bitiş düğümünü, bir optik yoldaki maksimum kullanılabilir bitişik ve sürekli spektrum sayısını belirtmek için sırasıyla Tx ID, Signature, İSSN, Pathlet ID, Ingress and Egress Node ve Max Available Subcarrier(s) içerir.

Algorithm 1: Finding Contiguous Spectrums in wavelength converter-enabled infrastructure (FCS)

```

Input:  $M^{P_i^j}$  (PSM for pathlet  $j$  in network  $i$ )
Output: Max number of contiguous spectrums over  $P_i^j$ 

1  $count \leftarrow 0$ ,  $LongestContiguous \leftarrow 0$ 
2  $MaxContiguous \leftarrow L(m^{e_i^j})$ 
3 for  $row \leftarrow 1$  to  $L(P_i^j)$  do
4   for  $column \leftarrow 1$  to  $L(m^{e_i^j})$  do
5     if  $M^{P_i^j}[row][column] == 1$  then
6        $count ++$ 
7       if  $count > LongestContiguous$  then
8          $LongestContiguous = count$ 
9     else
10       $count = 0$ 
11   if  $LongestContiguous < MaxContiguous$  then
12      $MaxContiguous = LongestContiguous$ 
13 return  $MaxContiguous$ 

```

Algoritma 1: Dalga-boyu dönüştürücü-etkin altyapıda bitişik spektrumları bulma

SpectrumChain'de, İSS denetleyicileri (i) işlemin dijital olarak imzalandığından, (ii) maksimum kullanılabilir spektrumun pozitif bir değer olduğundan, (iii) meşru bir İSSN atandığından ve (iv) işlemdeki giriş ve çıkış uç noktalarının işlemi oluşturan İSS'ye ait olduğundan emin olmak için bir dizi kriter uygulayarak işlemleri doğrular.

Tablo 5: Şekil 21'de gösterildiği gibi R5 ve R7 sınır aygıtları arasındaki parça yollar için İSS2 denetleyici tarafından oluşturulan işlemleri

Tx ID	Signature	ISP/N	Pathlet ID	Ingress Node	Egress Node	Max Available Subcarrier(s)
ISP2_1	0000kbxf...	ISP2	R5_R7_1	R5	R7	3
ISP2_2	0000asx34...	ISP2	R5_R7_2	R5	R7	3
ISP2_3	0000fdxr4...	ISP2	R5_R7_3	R5	R7	2
ISP2_4	0000ytx6j...	ISP2	R5_R7_2	R5	R7	1
ISP2_5	0000erfg4...	ISP2	R5_R7_3	R5	R7	2
ISP2_6	0000uprth...	ISP2	R5_R4_1	R5	R7	2
...

Bir örnek vermek gereklirse, Tablo 5, Şekil 21'de gösterildiği gibi İSS2'deki sınır düğümleri R5 ve R7 arasındaki optik yollar için İSS2 denetleyici tarafından yapılan işlemleri göstermektedir. İSS2 denetleyici farklı optik parça yollar için kendi ID'lerini ve tüm cihazların dalga-boyu dönüştürme kabiliyetine sahip olduğu senaryoda, Algoritma 1 ile belirlenen maksimum bitişik spektrum sayısını ve önceki bölümde açıklanan diğer iki senaryoda Algoritma 2 kullanılarak hesaplanan maksimum kullanılabilir bitişik ve sürekli spektrum sayısını kullanarak işlem üretmeye başlar. Hem Algoritma 1 hem de Algoritma 2, PSM kullanarak bir

parça yol üzerindeki tüm fiber optik bağlantılar arasında her bir spektrum indeksinin kullanılabilirliğini bulmak için bir parça yolun PSM'sini alır.

Algorithm 2: Finding Contiguous & Continuous Spectrums in wavelength converter-free infrastructure (FCCS)

Input: $M^{P_i^j}$ (PSM for pathlet j in network i)
Output: Max number of contiguous & continuous spectrums over P_i^j

```

1  $B = \text{ones}(1, L(m e_i^j))$            // B is an  $1 \times L(m e_i^j)$  array of all ones
2  $\text{count} \leftarrow 0$ ,  $\text{MaxContiguousContinuous} \leftarrow 0$ 
3 for  $\text{column} \leftarrow 1$  to  $L(m e_i^j)$  do
4   for  $\text{row} \leftarrow 1$  to  $L(P_i^j)$  do
5      $B[\text{column}] = B[\text{column}] \cdot M^{P_i^j} [\text{row}][\text{column}]$ 
6   if  $B[\text{column}] == 1$  then
7      $\text{count} ++$ 
8     if  $\text{count} > \text{MaxContiguousContinuous}$  then
9        $\text{MaxContiguousContinuous} = \text{count}$ 
10  else
11     $\text{count} = 0$ 
12 return  $\text{MaxContiguousContinuous}$ 

```

Algoritma 2: Dalga-boyu dönüştürücüsüz altyapıda bitişik ve sürekli spektrumları bulma

Algoritma 1, spektrum sürekliliği kısıtlaması cihazların dalga-boyu dönüştürme özellikleriyile başa çıktılarından, bir yol üzerindeki bağlantılarında yalnızca bitişik spektrumların maksimum sayısını hesaplar. Bu nedenle, bir parça yol üzerindeki bağlantılarında mevcut spektrumları aramaya başlar (satır 3 - 4) ve tüm fiber bağlantılarındaki (satır 11 - 12) maksimum mevcut bitişik spektrumları hesaplamak için her bağlantıda (satır 5 - 10) mevcut en uzun bitişik spektrumların (yani, değerleri boş bir spektrum olarak 1'e eşit olan maksimum spektrum sayısı, aksi takdirde 0) kayıtlarını tutar. Algoritma 2, bir parça yol üzerindeki bağlantılarında hem bitişik hem de sürekli spektrumların maksimum sayısını hesaplar. Bu nedenle, sürekliliği sağlamak amacıyla mevcut olup olmadıklarını bulmak için bağlantılarındaki aynı spektrumlar üzerinde arama yapmaya başlar (satır 3 - 5). Ardından, bir yol üzerindeki bağlantılarında hem bitişik hem de sürekli spektrumların maksimum sayısı, bir yol üzerindeki bağlantılarında aynı spektrumları kontrol ederek hesaplanır (satır 6 - 11).

Link	R5-R8	R8-R7
Subcarrier Index	1	
2		
3		
4		
5		
6		
7		

Link	R5-R6	R6-R8	R8-R7
Subcarrier Index	1		
2			
3			
4			
5			
6			
7			

Link	R5-R7
Subcarrier Index	1
2	
3	
4	
5	
6	
7	

Link	R5-R8	R8-R7
Subcarrier Index	1	
2		
3		
4		
5		
6		
7		

Link	R5-R7
Subcarrier Index	1
2	
3	
4	
5	
6	
7	

(a) Available and occupied subcarriers for each pathlet before allocation

(b) Contiguous and continuous occupied subcarriers with $\beta = 2$ in HWS

Link	R5-R8	R8-R7
Subcarrier Index	1	
2		
3		
4		
5		
6		
7		

Link	R5-R6	R6-R8	R8-R7
Subcarrier Index	1		
2			
3			
4			
5			
6			
7			

Link	R5-R7
Subcarrier Index	1
2	
3	
4	
5	
6	
7	

Link	R5-R8	R8-R7
Subcarrier Index	1	
2		
3		
4		
5		
6		
7		

Link	R5-R7
Subcarrier Index	1
2	
3	
4	
5	
6	
7	

(c) Contiguous and continuous occupied subcarriers with $\beta = 2$ in BWS

(d) Contiguous and continuous occupied subcarriers with $\beta = 2$ in NWS

Şekil 23: Şekil 21'de gösterildiği gibi İSS2'nin R5 ve R7 sınır cihazları arasındaki 4 farklı durumda parça yollar üzerinden esnek optik iletişim kaynak tahsisinin bir tasviri: Spektrum tahsisinden önce, HWS senaryosu, BWS senaryosu ve NWS senaryosu

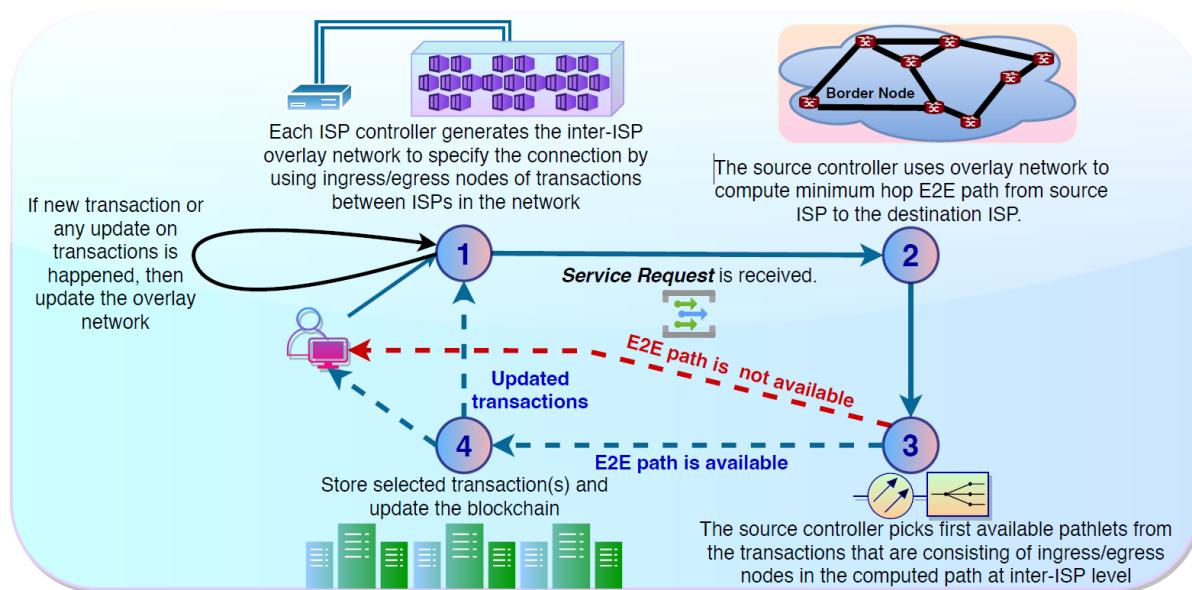
Şekil 23'te sırasıyla kırmızı ve yeşil hücreler kullanılarak İSS2'deki her bir fiber optik bağlantı için dolu ve kullanılmayan spektrumların sayısı gösterilmektedir. Şekil 23a'da yeşil renkli hücreler, İSS2'deki üç cihazlar arasındaki yollar üzerinde o anda kullanılabilir maksimum bitişik spektrumları göstermektedir. İSS2 denetleyicisi, işlemdeki maksimum kullanılabilir spektrumların sayısını hesaplamak için Algoritma 1 aracılığıyla Tablo 5'deki gibi parça yollar üzerindeki spektrumların kullanılabilirliğine dayalı olarak ilk işlemlerini oluşturur.

Şekil 23b - Şekil 23d, Hop-by-Hop Dalgaboyu Değiştirme (HWS), Sınır-Nod-Sadece Dalgaboyu Değiştirme (BWS) ve Dalgaboyu Değiştirme Yok (NWS) senaryolarını kullanırken İSS2_1, İSS2_2 ve İSS2_3 Tx ID'leri ile farklı parça yollarda spektrum talebi ($\beta = 2$ spektrum) ile bir hizmet talebi kullanarak tahsis edilen spektrumları gri renkli hücrelerle göstermektedir. Böylece, İSS2 denetleyicisi, İSS2 4, İSS2 5 ve İSS2 6'da olduğu gibi aynı Pathlet ID ile farklı imzalar kullanarak Tablo 5'deki yeni kullanılmayan spektrum ile işlemleri günceller.

3.5.3 SpectrumChain (SC) Çerçevesinin Yol Bulma Süreci

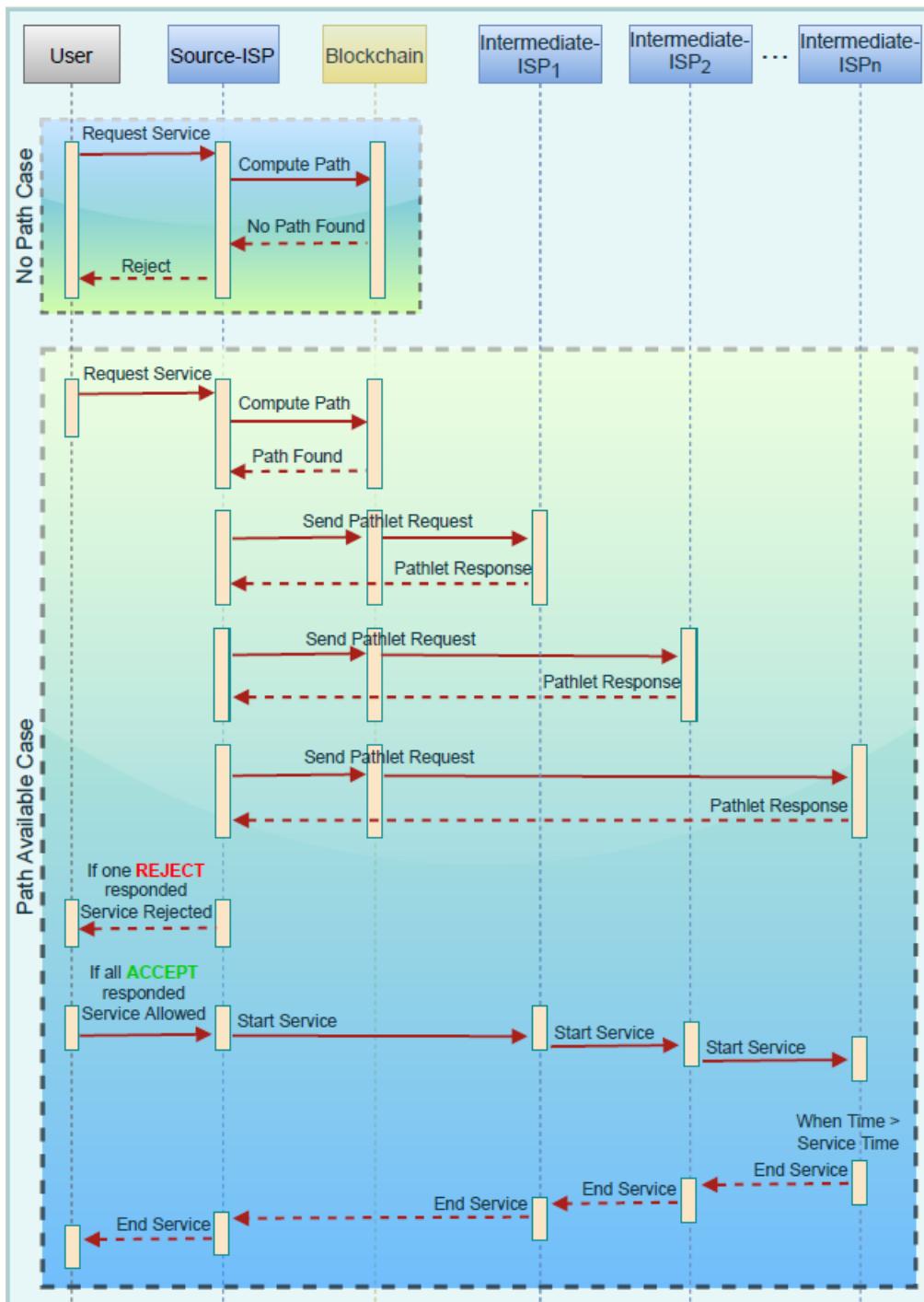
SpectrumChain'de uçtan uca optik yol bulma işleminin sıra diyagramı, bir SDON ağında hizmet isteklerinin nasıl yönetileceğini Şekil 25'de gösterilmektedir. Bir kullanıcı/istemci bir İSS denetleyicisine bir hizmet talebi gönderdiğinde, İSS denetleyicisi, blokzincir defterini kullanarak ağının bir uç cihazından hedef İSS'nin bir sınır düğümüne kadar mesajda belirtilen QoS özelliklerine ve tercihlerine uygun yollardan oluşan bir uçtan uca optik fiber rotasını hesaplamaya başlar. Kaynak-İSS denetleyicisi, Tablo 5'deki örneğe benzer şekilde, yolların yalnızca en güncel işlemlerini kullanarak nihai uçtan uca optik yolunu belirler. Uçtan uca optik

bağlantısı mevcut değilse SC, hizmet talebini reddeder. Kaynak-İSS denetleyicisi, bir uçtan uca optik yolu hizmet isteğiinin QoS kriterlerini yerine getirdiğinde, uçtan uca yoldaki bir yol ile her İSS denetleyicisine parça yol istek mesajını ileter. Her İSS denetleyicisi, QoS kriterlerini yerine getirmek için yol uygulama istek mesajlarını aldıktan sonra yanıt verir ve ilgili QoS parametrelerine bağlı olarak bir hizmet yanıt mesajını (yani Kabul Et veya Reddet) geri gönderir.



Şekil 24: Bir hizmet isteği için İSS'ler arası yönlendirmenin iş akışı

Uçtan uca yol üzerindeki herhangi bir İSS denetleyicisi kaynak-İSS denetleyicisine Reddet yanıtını verirse kaynak denetleyici aynı gereksinimlere sahip başka bir uçtan uca yolu aramaya başlar. Kaynak-İSS ve hedef-İSS denetleyicileri, uçtan uca optik yolu kullanıcından kaynak-İSS sınır düğümüne ve hedef-İSS sınır düğümünden hedef-ana cihaza kadar olan kısmi optik yollar için alan-içi yönlendirmelerini kullanarak hesaplar. Kaynak-İSS sınır düğümünden hedef-İSS sınır düğümüne uçtan uca optik bağlantısı böylece kurulur. Son olarak denetleyiciler, geçerli sürenin hizmet (istek) süresini aşması durumunda, kendi parça yolları için uçtan uca yolu üzerinden hizmeti sonlandırırlar.



Şekil 25: *SpectrumChain*'de uçtan uca optik yol bulma işleminin sıra diyagramı

SpectrumChain mimarisinin blokzincir altyapısı ile birlikte yol seçimi kısmını daha detaylandırmak için, Şekil 24'te bir iş akışı diyagramı verilmiştir. Bir kullanıcı bir hizmet talebini başlattıktan sonra, kaynak İSS denetleyicisi aşağıdakileri yaparak bir uçtan uca yolunu hesaplayacak ve seçecektir: blokzincir defterinde bulunan işlem bilgilerini kullanarak.

3.6 Blokzincir Destekli Çok Aktörlü Pekiştirmeli Öğrenme Algoritması ile QoS Merkezli Uçtan Uca Yol Hesaplanması

Pekiştirmeli Öğrenme (RL), bir aracıya bir ortamda nasıl hareket edeceğini konusunda rehberlik eden bir politikayı öğrenmek için deneme-yanılma yaklaşımını kullanan bir makine öğrenme yöntemidir. RL temsilcisi, ortamındaki her eylemin sonucu olarak yanında bir ödül alır ve kümülatif ödülü optimize etmeyi amaçlar. Durum, aracının aralarında hareket ettiği ve ona göre eyleme geçtiği başka bir RL kavramıdır. RL algoritmaları, dinamik trafik değişikliklerine uyum sağlamak ve etiketli veriler olmadan ağın mevcut durumu için en iyi yönlendirmeye karar vermek üzere eğitilebilir. Hedeflenen uygulamalar için QoS'nin sağlanması sağlanmak amacıyla RL teknikleri, ağ verimini, bant genişliği kullanımını en üst düzeye çıkarmak, gecikmeyi en aza indirmek veya önceden belirlenmiş diğer hedefler açısından ağ performansını iyileştirmeyi amaçlar. Pekiştirmeli Öğrenme (Reinforcement Learning - RL) çerçevesinin arkasındaki ana fikir, RL aktörünü (RL agent) sürekli olarak bazı eylemlerde bulunarak ve çevre sonuçlarından öğrenerek çevreyi öğrenmesi için eğitmektir. İşlem, RL aktörü ödül (reward) adı verilen maksimum ödülü (reward) döndürecek önceden tanımlanmış bir hedefe ulaşana kadar tekrarlanır. Süreç, $(s_0, a_0, r_0), (s_1, a_1, r_1), \dots, (s_k, a_k, r_k)$ şeklinde bir durum (state), eylem (action) ve ödül (reward) dizisi oluşturur.

Uçtan uca İSS'ler arası servis kaliteli odaklı yol probleminin çözümü, çok sayıda alan ve araştırma için gerekli olan iki düğüm arasındaki en yüksek iletim hacmine sahip yola tekabül etmektedir. Projede, YTA mimarisinde Pekiştirmeli Öğrenme (Reinforcement Learning – RL) modeli olan Q-Learning yöntemine dayalı bir yönlendirme yöntemi geliştirilmiştir. Geliştirilen yönlendirme modeli, ödül fonksiyonunu blokzincir üzerinde tutulan parça yolların bağlantı (link) bant genişliği (bandwidth), gecikme (delay) ve güvenilirlik (reliability) göre işlemekte ve RL aracılığıyla kaynak ve hedef arasındaki mevcut maksimum bant genişliğine sahip uçtan uca iletim yolunu belirleyebilmektedir. Geliştirilen modelde ki yaklaşımıza, ağ trafiği değiştiğinde veya ağ tıkanıklığı meydana geldiğinde, mevcut maksimum bant genişliği (bandwidth), gecikme (delay) ve güvenilirlik (reliability) ile en uygun iletim yolunu eğitmek (belirlemek) için ağ bilgilerini verimli bir şekilde kullanmayı hedeflemektedir.

Aşağıda, geliştirilen RL-tabanlı yönlendirme modelinde kullanılan terimler ve bileşenler açıklanmıştır.

Durum (State): Herhangi bir işlem sırasında bir test ortamının durum temsili. Sembolik gösterim: $s_i \rightarrow i$ 'inci durumdaki durum özelliği, s_i toplam durum kümesinin bir alt kümesidir ($s_i \in S$).

Eylem (Action): Hedeflere ulaşmak için durumu değiştiren olay. Sembolik gösterim: $a_i \rightarrow i$ 'inci eylemdeki durum özelliği, a_i toplam eylem kümесinin bir alt kümесidir ($a_i \in A$).

Ödül (Reward): Eylemin takdir edilme (appreciation) düzeyinin sayısal miktarı. Örneğin, pozitif bir sayısal ödül değeri, pozitif bir eylem olarak gösterilebilir. Sembolik gösterim: $r_i \rightarrow i$. eylem de ödül, i . eylem de beklenen ödül, $R_i = E(r_{i+1}|s_i)$.

Durum Geçiş Olasılığı (State Transition Probability): Belirli bir durumdan (state) sonraki duruma geçme olasılığı. Durum geçiş olasılığı, tüm model parametrelerinin bilindiği deterministic (deterministic) model için mevcuttur. Stokastik (stochastic) bir ortam söz konusu olduğunda, parametreler bilinmemektedir. Sembolik gösterim: $P[s_{i+1}|s_i, a_i]$

Azaltma Faktörü (Discount Factor): Gelecekteki ödüllerin 0 ile 1 arasında değişen bugünkü sayısal değeri. Faktör, ödül değerini sonsuz durum-eylem-ödül uzayında sınırlar. Sembolik gösterim: $\gamma \in [0, 1]$.

Geri Dönüş (Return): i . eylemden elde edilen toplam azaltılmış (discounted) ödül. Sembolik gösterim: $G_i = r_{i+1} + \gamma r_{i+2} + \dots = \sum_{k=0}^{+\infty} \gamma^k r_{i+k+1}$, burada γ azaltma bir faktördür.

Değer Fonksiyonu (Value Function): Her durumun ve eylemin faydasını ölçen sayısal bir değer. Sembolik gösterim: $V(s) = E[G_i|s_i = s] \forall s \in S$.

Politika (Policy): Aktörün davranışı veya stratejisi. Politika stokastik veya deterministik olabilir. Sembolik gösterim: deterministik politika, $a = \pi(s)$ $\pi(s): S \rightarrow A$; stokastik politika, $\pi(a|s) = P[a|s] \forall s \in S, \forall a \in A$.

Geliştirilen RL tabanlı yönlendirme modelinde kullanılan Q-learning algoritmasının avantajı, öğrenilen stratejileri kaydetmesi ve en yüksek ödülü elde etmek için uygun eylemlerin alınmasında aktöre rehberlik etmesidir. $Q(s, a)$ terimi, durum ve eylem çiftlerinden oluşan bir matrisi temsil etmektedir. Aktör çevre ile etkileşime girdiğinde, eyleme duruma ve benimsenen stratejiye göre karar verilir ve eylemin yürütülmesi, aktörün bir sonraki durumu seçme olasılığını etkiler. Elde edilen değere göre $Q(s, a)$ matrisi güncellenir. Sürekli yineleme yoluyla, $Q(s, a)$ en yüksek kümülatif ödüle yaklaşır. Çalışmada, QoS motivasyonlu ağlar arası trafik yönlendirmesini ele almak için Q-learning algoritmasını kullanan bir Pekiştirmeli Öğrenme (RL) yaklaşımı kullanılmıştır. Bu bağlamda, herhangi bir genellik kaybı olmaksızın, geliştirilen RL modelinde aktör (YTA denetleyicisi), sonuçta en uygun E2E yolunu arayan YTA denetleyicisidir. Eylemler (actions) uzayı, parça yolların (pathlet) seçimi iken durum (state) uzayı, ağlar arası grafinin düğümlerini ve kenarlarını içerir. Aktör çevreyi keşfettikçe, Q değerleri yinelemeli olarak güncellenir ve her durum-eylem (state-action) çiftini tahmini bir kümülatif ödülle ilişkilendirir. Q-learning güncelleme kuralı aşağıda verilmiştir:

$$Q(s, a) \leftarrow (1 - \alpha) \cdot Q(s, a) + \alpha \cdot (r + \gamma \cdot \max_a Q(s', a))$$

Yukarıda ki formül de; $Q(s, a)$, s durumu ve a eylemi için Q değeri, α öğrenme oranı, r , s ağ durumunda a eylemi gerçekleştirildikten sonra elde edilen ödül, γ azaltma faktörü, s' sonraki durum, $\max_{\alpha} Q(s', a)$ sonraki aşamada en yüksek Q değerine sahip eylem ve α ve γ , $[0, 1]$ aralığındaki gerçek sayılardır. α parametresi, aktör eğitimi sırasında eğitim hızını ve yakınsama süresini etkilemektedir ve γ , r 'nin seçimi sırasında aktörün ağırlığını etkiler. Model de γ 0'a yaklaşlığında, aktör mevcut ödülü daha fazla dikkat ederken γ 1'e yaklaşlığında, aktör gelecekteki ödülü daha fazla dikkat eder. Ayrıca, aktörün ödül değerini bulması için stratejiyi seçmek üzere ε -greedy ismi verilen bir metot kullanıldı. Aktör, ε 'den küçük bir değer seçtiğinde rastgele keşif (exploration) gerçekleştirir. Aktör, $1 - \varepsilon$ 'den büyük veya eşit bir değer seçtiğinde ise, sömürme (exploitation) yoluyla en yüksek ödülü sahip eylemi belirler.

Geliştirdiğimiz modelde önemli bir katkı ve özgünlük, RL de ki öğrenme sürecini geliştirmek için blokzincir işlemlerinin (transactions) süreçce entegrasyonudur. Her YTA denetleyicisi (agent), kendi ağında bulunan parça yollarını (pathlets) QoS ölçümüleriyle birlikte hesaplar ve bunlar için parça yolları (pathlets) temsil eden blokzincir işlemlerini üretir. Bu işlemler, blokzincir düğümlerini temsil eden YTA ağ denetleyicilerinin blokzincir defterlerinde saklanır. Böylece, ağ denetleyicileri, genel çoklu ağ durumunu merkezi olmayan ve şeffaf bir görünümünü elde ederek işlem verilerine erişebilir. RL aktörü görevini gerçekleştiren ağ denetleyicileri, ağ performansına ilişkin geçmiş verileri dikkate alarak bilinçli ve etkili kararlar vermek için bu bilgileri kullanır. Geliştirilen RL tabanlı QoS bazlı yol bulma çerçevesinde, bir YTA ağ denetleyicisini temsil eden bir aktör (agent), en uygun yolu bulmak için blokzincir işlemleri aracılığıyla ağlar arası overlay networkde gezinir. Aktörün karar verme süreci, s durumunda a eylemini seçme olasılığını tanımlayan, $\pi(a|s)$ olarak gösterilen bir politika tarafından yönetilir. Özette, RL algoritması, yani Q -learning, durum-eylem (state-action) çiftleriyle ilişkili Q değerlerini güncellemek için kullanılarak, aktörü en uygun kararları alması için yönlendirmektedir.

Aktör, politika fonksiyonuna dayalı olarak keşif (exploration) ve sömürü (exploitation) arasında karar verir. Keşif-sömürüğ dengesi (exploration-exploitation tradeoff), keşif olasılığı ϵ tarafından kontrol edilir. Rastgele oluşturulmuş bir sayı ϵ 'dan küçükse, aktör rastgele bir parça yol seçerek (eylem) keşif yapar veya maksimum Q değerine sahip eylemi seçerek yararlanır.

$$\text{Keşif (Explore): } \frac{1}{|\mathcal{A}|}$$

$$\text{Sömürü (Exploit): } \operatorname{argmax}_a Q(s, a)$$

Ödül fonksiyonu, QoS ölçümüini karşılayan ve optimize eden parça yolların seçimini teşvik edecek şekilde tasarlanmıştır. Ödül hesaplaması aşağıdaki verilmiştir:

$$\mathcal{R}(s, a, s') = c_D \cdot \frac{1}{D(s, a)} + c_B \cdot \frac{1}{B(s, a)} + c_R \cdot R(s, a) + c_\delta \cdot \delta(s', endNode)$$

burada c_D , c_B , c_R ve c_δ sırasıyla gecikmenin (delay), bant genişliğinin (bandwidth), güvenilirliğin (reliability) ve uç düşüme ulaşmanın önemini temsil eden ağırlık katsayılarıdır. $D(s, a)$, $B(s, a)$ ve $R(s, a)$, s ağ durumunda a eyleminin gerçekleştirilmesi sonucunda oluşan yol üzerinde sırasıyla gecikme, kullanılabilir bant genişliği ve güvenilirlik değerleridir. $\delta(s', endNode)$, s' uç düşüme ulaşırsa 1, aksi halde 0 olan ikili bir fonksiyondur. Burada tanımlanan QoS motivasyonlu ödüllendirme yaklaşımı, ağ koşullarındaki dinamik değişiklikleri ve blokzincir işlemleri yoluyla elde edilen performans verilerini dikkate alarak akıllı bir uyarlanabilir uçtan uca yol keşfi sağlamaktadır.

$\pi(a|s)$ politika fonksiyonu, RL'de çok önemli bir bileşendir ve mevcut durum s 'de belirli bir eylemi seçme olasılığını belirleyerek aktörün davranışını belirler. Politika fonksiyonu aşağıda verilmiştir:

$$\pi(a|s) = \begin{cases} \frac{\epsilon}{|\mathcal{A}|} + (1 - \epsilon) \cdot \text{argmax}_a Q(s, a), & \epsilon \text{ olasılık} \\ \frac{1}{|\mathcal{A}|}, & 1 - \epsilon \text{ olasılık} \end{cases}$$

burada $\pi(a|s)$, belirli bir s durumunda eylem seçme olasılığı; ϵ , keşif ve sömürü arasındaki dengeyi kontrol eden keşif olasılığı; $|\mathcal{A}|$ olası eylemlerin sayısı ($\mathcal{A} = \{a_1, a_2, \dots, a_m\}$); $|\mathcal{S}|$ olası durumların sayısı ($\mathcal{S} = \{s_1, s_2, \dots, s_n\}$); $Q(s, a)$, s durumu ve a eylemi için Q değeri; $\text{argmax}_a Q(s, a)$, s durumundaki Q değerini maksimuma çıkaran eylemdir.

Politika fonksiyonu bir keşif-sömürü stratejisini içermektedir. ϵ olasılığıyla, aktör, tek düzeye (uniform) olasılığa ($\frac{\epsilon}{|\mathcal{A}|}$) sahip rastgele bir pathlet eylemi seçimi ile yeni yolların keşfedilmesine izin vererek araştırma yapar. $1 - \epsilon$ olasılıkla, aktör, maksimum Q değerine ($\text{argmax}_a Q(s, a)$) sahip eylemi seçerek, geçmişte daha yüksek kümülatif ödüller gösteren eylemleri tercih ederek istismar eder. ϵ parametresi keşif derecesini kontrol eder ve azaldıkça, aktör öğrenilen bilgiye dayanarak daha fazla yararlanma eğilimindedir. Bu denge, aktörün hem potansiyel olarak daha iyi eylemleri (pathlet seçimlerini) keşfetmesi hem de bilinen yüksek ödüllü eylemlerden yararlanması açısından çok önemlidir.

Aşağıda, proje de ki ilgili iş paketleri çalışmaları kapsamında geliştirilen Pekiştirmeli Öğrenme (RL) tabanlı ağlar arası uçtan uca QoS bazlı trafik yönlendirme çerçevesinde kullanılan çeşitli algoritmalar verilerek açıklanmıştır.

Algorithm 3 Reinforcement Learning for QoS-based Routing

```

1: function RL_BC_SDN( $N_{BC}$ ,  $|V_{BC}|$ ,  $I$ ,  $\alpha$ ,  $\gamma$ ,  $\epsilon$ , startNode, endNode)
2:    $qValues[ ]_{n \times m} \leftarrow [0]_{n \times m}$   $\triangleright Q(s, a) \leftarrow 0$ 
3:   for episode  $\leftarrow 1$  to  $I$  do
4:      $s \leftarrow$  startNode
5:     while  $s \neq$  endNode do
6:        $s' \leftarrow$  SELECTACTION( $s$ ,  $qValues$ ,  $N_{BC}$ ,  $|V_{BC}|$ )
7:        $r \leftarrow$  CALCULATEREWARD( $s$ ,  $s'$ , endNode)
8:        $C \leftarrow$  GETMAXQVALUE( $s'$ ,  $qValues$ )
9:        $qValues[s'][a] \leftarrow (1 - \alpha) \cdot qValues[s][a] + \alpha \cdot (r + \gamma \cdot \max_{\text{next}} qValue)$ 
10:       $s \leftarrow s'$ 
11:    end while
12:   end for
13: end function

```

Algoritma 3: QoS bazlı yönlendirme için Pekiştirmeli Öğrenme algoritması

Algoritma 3, QoS tabanlı yönlendirme için geliştirdiğimiz takviyeli öğrenme ana algoritmasıdır. Algoritma, girdi olarak; blokzincir ağını temsil eden N_{BC} , blokzincir ağındaki düğüm sayısını temsil eden $|V_{BC}|$, iterasyon sayısını temsil eden I , öğrenme oranını temsil eden α , indirgeme faktörünü temsil eden γ , keşif olasılığını temsil eden ϵ , hizmetin başlangıç düğümünü temsil eden *startNode* ve hizmetin bitiş düğümünü temsil eden *endNode* değerlerini alır. Fonksiyon, her bir durum-eylem çifti için Q değerlerini içeren bir matris olan $qValues[]_{n \times m}$ 'ı başlangıçta sıfırlar (satır 2). Daha sonra, belirli bir sayıda tekrarlanan bölümler (epizodlar) için döngüye girilir (satırlar 3-12). Her bir bölümde, RL aktörü başlangıç düğümüyle başlatılır (satır 4). Aktör, bitiş düğümüne ulaşana kadar döngü içinde kalır (satırlar 5-11). Her adımda, aktörün (denetleyici) mevcut durumunda eylem olarak bir pathlet (parça yol) seçmesi (*SelectAction* fonksiyonu çağrıları), bu eylemin ardından ödül hesaplaması (*CalculateReward* fonksiyonu çağrıları) ve en yüksek Q değerini alarak Q değerlerini güncellemesi (*GetMaxQValue* fonksiyonu çağrıları) gerçekleştirilir (satırlar 6-9). Fonksiyon, tüm tekrarlanan bölümler tamamlandığında sonlanır (satır 13). Bu adımlar, QoS tabanlı yönlendirme için takviyeli öğrenme algoritmasının ana akışını temsil etmektedir. Bu süreçte, SDN denetleyicisi (aktör), ağı dolaşırken her adımda en iyi parça yolu (eylemi) seçerek, belirli bir durumda maksimum toplam ödül elde etmeye çalışır.

Algorithm 4 SelectAction Function

```

1: function SELECTACTION( $s, qValues, N_{BC}, |V_{BC}|$ )
2:   if  $rand() < \epsilon$  then
3:     return GETRANDOMACTION( $s$ )
4:   else
5:     return GETBESTACTION( $s, qValues, N_{BC}, |V_{BC}|$ )
6:   end if
7: end function

```

Algoritma 4: Eylem (action) seçim fonksiyonu

RL aktörü olan ağ denetleyicisi bir sonraki eylemi seçmek için Algoritma 4 de ki *SelectAction* fonksiyonunu kullanır. Bu fonksiyon, mevcut duruma göre aktörün keşif ve sömürü stratejilerini dengeleyen bir karar alır. Fonksiyon da ilk olarak (satır 2), rastgele bir sayı üretilerek bu sayının epsilon değerinden küçük olup olmadığı kontrol edilir. Eğer rastgele sayı epsilon değerinden küçükse, aktör keşif yapma stratejisini uygular ve *GetRandomAction* fonksiyonunu çağırarak rastgele bir pathlet (eylem) seçer (satır 3). Aksi halde, yani sayı epsilon değerinden büyükse, sömürü (Exploitation) stratejisini uygular ve *GetBestAction* fonksiyonunu çağırarak en iyi parça yolu (pathlet) seçer (satır 5). Bu şekilde, *SelectAction* fonksiyonu, RL algoritmasının keşif ve Exploitation arasındaki dengeyi sağlamaşını mümkün kılar. Bu dengenin korunması, aktörün hem ortamı keşfetmesine hem de öğrenilmiş bilgiyi kullanarak en iyi parça yolu seçmesine olanak tanır.

Algorithm 5 GetRandomAction Function

```

1: function GETRANDOMACTION(state)
2:   possibleActions  $\leftarrow$  List( $\mathcal{A}$ , state)
3:   availableActions  $\leftarrow$  { $a \mid \text{possibleActions}[a] = 1$ }
4:   if  $|\text{availableActions}| > 0$  then
5:     return SelectRandom(availableActions)
6:   else
7:     return -1                                 $\triangleright$  No possible actions
8:   end if
9: end function

```

Algoritma 5: Rastgele eylem (action) seçim fonksiyonu

Algoritma 5 ile verdığımız *GetRandomAction* fonksiyonu, mevcut duruma (current state) göre rastgele bir yol seçimi (eylem) seçmek için kullanılır. Fonksiyon, ilk olarak mevcut duruma göre mümkün olan tüm eylemleri (parça yolları) içeren bir liste oluşturur (satır 2). Ardından, bu listeden yalnızca mümkün olan eylemleri (possible actions) içeren bir alt kümeye oluşturulur (satır 3). Eğer mevcut duruma göre mümkün olan eylemlerin sayısı sıfırdan büyükse, yani en az bir mümkün eylem varsa, bu alt kümeden rastgele bir eylem seçilir (satır 4-5). Eğer hiçbir mümkün eylem yoksa, yani mevcut duruma göre hiçbir eylem yapılamıyorsa, -1 değeri döndürülür (satır 6-7). Bu şekilde, *GetRandomAction* fonksiyonu, RL algoritmasının keşif stratejisini uygular ve

aktörün mevcut duruma göre rastgele bir parça yol (eylem) seçmesini sağlar. Bu keşif stratejisi, denetleyicinin (aktörün) ağı daha kapsamlı bir şekilde keşfetmesine ve farklı eylemleri denemesine olanak tanır.

Algorithm 6 GetBestAction Function

```

1: function GETBESTACTION(state, qValues, NBC, |VBC|)
2:   bestAction  $\leftarrow -1$ 
3:   bestValue  $\leftarrow -\infty$ 
4:   for action  $\leftarrow 1$  to |VBC| do
5:     if NBC[state][action] = 1 and qValues[state][action] > bestValue then
6:       bestAction  $\leftarrow$  action
7:       bestValue  $\leftarrow$  qValues[state][action]
8:     end if
9:   end for
10:  return bestAction
11: end function
  
```

Algoritma 6: En iyi eylem (action) seçim fonksiyonu

Algoritma 6 ile verdığımız *GetBestAction* fonksiyonu, mevcut duruma göre en iyi parça yolu seçmek için kullanılır. Fonksiyon, başlangıçta en iyi eylemi ve bu eylemin değerini belirlemek için -1 ve eksi sonsuz değerle başlar (satırlar 2-3). Ardından, mevcut ağ durumuna göre mümkün olan tüm eylemler üzerinde döngü başlatılır (satır 4-9). Her bir eylem için, eğer bu eylem mümkün ise (yani, $N_{BC}[\text{state}][\text{action}] = 1$) ve bu eylemin değeri en yüksek değerden daha büyükse, en iyi eylem ve değeri güncellenir (satırlar 5-8). Sonuç olarak, fonksiyon en iyi eylemi belirler ve bu eylemi döndürür (satır 10). Bu şekilde, *GetBestAction* fonksiyonu, RL algoritmasının en iyi eylemi seçme stratejisini uygular ve aktörün mevcut duruma göre en uygun eylemi belirlemesini sağlar. Bu strateji, aktörün mevcut durumu dikkate alarak en yüksek ödülü sağlayacak eylemi seçmesine olanak tanır.

Algorithm 7 CalculateReward Function with Pathlet Information

```

1: function CALCULATEREWARD(state, action, endNode)
2:   reward  $\leftarrow -1$ 
3:   pathletQoS  $\leftarrow$  GetPathletInfo(state, action)
4:   bandwidthReward  $\leftarrow c_B \times \text{pathletQoS.bandwidth}^{-1}$ 
5:   delayReward  $\leftarrow c_D \times \text{pathletQoS.delay}^{-1}$ 
6:   reliabilityReward  $\leftarrow c_R \times \text{pathletQoS.reliability}$ 
7:   reward  $\leftarrow$  bandwidthReward + delayReward + reliabilityReward
8:   if (state, action)  $\leftarrow$  endNode then
9:     reward  $\leftarrow$  reward +  $c_\delta \times 1$ 
10:  end if
11:  return reward
12: end function
  
```

Algoritma 7: Pathlet bilgileri kullanılarak ödülün hesaplanması

Algoritma 7, blokzincir işlemleri (transactions) olarak tutulan pathlet bilgilerini kullanılarak belirli bir durum ve eylem için RL ödülünü hesaplamaktadır. *CalculateReward* fonksiyonu, başlangıçta ödülü -1 olarak tanımlar (satır 2). Daha sonra, *GetPathletInfo* fonksiyonu kullanılarak durum ve eyleme (seçilen parça yol) göre parça yolun QoS bilgileri alınır (satır 3). Bu bilgilere dayanarak, bant genişliği, gecikme ve güvenilirlik değerleri üzerinden ödüller hesaplanır (satırlar 4-6). Yukarıda verilen ödül fonksiyonunda da görüldüğü üzere, bant genişliği ve gecikme için ters orantılı bir ilişki kullanılırken, güvenilirlik doğrudan bir orantı ile ilişkilendirilir. Son olarak, hesaplanan ödüller toplanır ve toplam ödül belirlenir (satır 7). Eğer durum, eylem son düşüme (endNode) eşitse, bir kısmi ödül eklenir (satırlar 8-10). Sonuç olarak, fonksiyon hesaplanan toplam ödülü döndürür (satır 11). *CalculateReward* fonksiyonu, QoS tabanlı yönlendirmede belirli bir durum ve eylem için uygun ödülün hesaplanmasıını sağlar. Bu sayede, takip edilen yolun kalitesine göre aktörün davranışını şekillendirerek, daha iyi performans elde edilir.

Algorithm 8 GetMaxQValue Function

```

1: function GETMAXQVALUE(nextState, qValues, NBC, |VBC|)
2:   maxQValue ← −∞
3:   for action ← 1 to |VBC| do
4:     if NBC[nextState][action] = 1 and qValues[nextState][action] >
       maxQValue then
5:       maxQValue ← qValues[nextState][action]
6:     end if
7:   end for
8:   return maxQValue
9: end function

```

Algoritma 8: Maksimum Q değerinin hesaplanması

Son olarak, Algoritma 8 de verilen *GetMaxQValue* fonksiyonu, bir sonraki durum için en yüksek Q değerini bulmayı sağlar. Fonksiyon, başlangıçta maxQValue değişkenini eksiz sonsuz olarak tanımlar (satır 2). Daha sonra, tüm parça yol seçimleri (eylemler) üzerinde döngüye girilir ve her eylemin durum ve sonraki duruma bağlı olarak geçerli olup olmadığı kontrol edilir (satırlar 3-7). Eğer eylem geçerli ise ve Q değeri mevcut maxQValue'den büyükse, maxQValue güncellenir (satırlar 4-5). Döngü tamamlandığında, en yüksek Q değeri maxQValue olarak döndürülür (satır 8). *GetMaxQValue* fonksiyonu, belirli bir durum ve sonraki durum için en iyi eylemin Q değerini elde etmeye yönelik bir araç olarak kullanılır. Bu sayede, RL algoritması sonraki adımlar için en iyisini seçebilir ve belirli bir durumda en yüksek ödülü sağlayan parça yolu belirleyebilir.

4. BULGULAR

Bu bölüm, proje de gerçekleştirilen çalışmaların performans değerlendirmelerine yönelik simülasyon/test ortamı sonuçlarını farklı alt başlıklar olarak aktarmaktadır.

4.1 QoSChain: Yazılım Tanımlı Ağlarda Blokzincir ile Otonom Sistemler Arası QoS Sağlama

Bu alt başlıktta, *QoSChain* çerçevesinin performansının YTA ağlarındaki iki yaygın QoS tabanlı yönlendirme stratejisine karşı değerlendirilme sonuçları verilmektedir (Karakus & Durresi, 2017c, 2017a). Bu stratejilerden bir tanesi önceki çalışmamızda (Karakus & Durresi, 2015) önerilen Hiyerarşik Yönlendirme Yaklaşımı (Hierarchical Routing Approach / HRA), diğer ise ağların mevcut operasyonel modunu yansitan Dağıtılmış Yönlendirme Yaklaşımıdır (Distributed Routing Approach / DRA). Çalışmada DRA'nın her zaman Sınır Geçidi Protokolü (Border Gateway Protocol / BGP) rotaları üzerinde çalıştığını varsayılmıştır. Bu nedenle aynı yerel tercih, ağırlık ve yerel kaynaklı veya toplu adreslere sahip iki benzer yol tespit etmesi durumunda, DRA İSS düzeyinde en kısa AS_PATH kriterini kullanılmaktadır. Performans analizi için ise Uçtan Uca Akış Kurulum Süresi (E2E Flow Setup Time - FST), Gönderilen ve İşlenen Mesajlar (Messages Exchanged and Processed - MEP) ve Hizmet Verilen Talepler (Requests Serviced - RS) metrikleri kullanılmıştır. Ayrıca *QoSChain* ve diğer yaklaşımın performansı üzerinde topoloji bazında bir duyarlılık analizi yapmak için İSS düzeyinde Random, NSFNET ve U.S. Backbone ağ topolojileri kullanılmıştır. Simülasyonlarda anahtar sayısı değişikçe Random, NSFNET ve U.S. Backbone ağlarının İSS-içi ayarlarında rastgele oluşturulmuş aynı topolojiler kullanılmıştır.

4.1.1 Deneysel kurulum

YTA topojisini oluşturmak ve simüle etmek ve Tablo 6'te gösterildiği gibi ilgili parametrelerin değerlerini ölçmek için Mininet emülatörü ve Ryu denetleyicisi kullanılmıştır. Çeşitli sayıda İSS denetleyicisi ve farklı boyutlarda blokzincir işlemleri ile özel bir blokzincir ağı oluşturulmuş ve kullanılmıştır. Şekillerde gösterildiği gibi, orijinal parça yol blokzincir işlem sayısına sahip önerilen çerçeve QC, genişletilmiş 250K blokzincir işlemi QC_250K ve 500K blokzincir işlemi QC_500K olarak simülasyonlarda gösterilmektedir. Orijinal parça yol boyutu yukarıda belirtilen blokzincir işlemi sayılarından daha az olduğunda, blokzincirde ki parça yol blokzincir işlemi boyutunu genişletmek için güncelleme blokzincir işlemleri oluşturulmuştur. Simülasyonlarda, ağ kaynağı sınırlamaları nedeniyle hizmet taleplerinin reddedilmemesi için yalnızca bant genişliği parametresi (1 Mbps) olan İSS'ler arası hızmet taleplerini dikkate alarak

fiziksel bağlantıarda yeterli bant genişliği (100 Gbps) sağlandı. Ayrıca, 0.5 bağlantı derecesine sahip ağları rastgele oluşturmak için Erdos-Renyi rastgele grafik modeli (Erdős P. and Rényi, 1964) tarafından tanımlanan rastgele bir topoloji üretici kullanıldı. Ek olarak, %95 istatistiksel anlamlılığa ulaşmak ve aşmak için her test için ortalama 30 çalışma gerçekleştirdik.

Tablo 6: Notasyon Tablosu

Sembol	Tanım
S	$S = \{s^j \mid s^j \text{ bir anahtar}, 1 \leq j \leq n\}$ (Tüm İSS'lerden YTA anahtarlarının kümesi)
C	$C = \{^k c \mid ^k c \text{ bir denetleyici}, 1 \leq k \leq n\}$ (Tüm İSS'lerden YTA denetleyicilerin kümesi)
S_i	$S_i = \{s_i^j \mid s_i^j \text{ bir anahtar}, s^j \in S\}$ (i'inci İSS'de bulunan anahtarların kümesi)
C_i	$C_i = \{^k C_i \mid ^k C_i \text{ bir denetleyici}, ^k c \in C\}$ (i'inci İSS'de bulunan denetleyicilerin kümesi)
fS	$fS = \{f s^j \mid f s^j \text{ bir anahtar}, s^j \in S\}$ (f akışı için olan uçtan uca yolunun üzerindeki anahtarların kümesi)
${}_f^k S$	${}_f^k S = \{{}_f^k s^j \mid {}_f^k s^j \text{ bir anahtar}, {}_f s^j \in fS\}$ (f akışı için olan uçtan uca yolun üzerindeki k 'inci denetleyiciye bağlı anahtarların kümesi)
fC	$fC = \{{}_f^k c \mid {}_f^k c \text{ bir denetleyici}, {}_f c \in C\}$ (f akışı için olan uçtan uca yolun üzerindeki denetleyicilerin kümesi)
fP	f akışının ilk paketi
ρ	OpenFlow packet_in mesajı
θ	OpenFlow flow_mod mesajı
$f\vec{y}$	f akışı için bir kaynak (\rightarrow) ağ cihazı (y)
$f\hat{y}$	f akışı için bir hedef (\leftarrow) ağ cihazı (y)
BR	“Broker” adında bir (süper) denetleyici
B^i	i'inci İSS'de bulunan sınır ağ cihazlarının kümesi
$T(x)_\text{proc}^y$	Bir ağ cihazında (y) bir mesajı (x) işleme süresi
$T(x)_\text{proc}^{y \rightarrow z}$	Bir ağ cihazından (y) bir ağ cihazına (z) bir mesajı (x) yayma süresi
$T(path_L)_\text{comp}^y$	Denetleyicide (y) bir yerel yol (İSS-içi) hesaplama süresi
$T(path_{E2EBC})_\text{comp}^y$	Servis isteği için denetleyicide (y) blokzincir kullanılarak bir uçtan uca yol hesaplama süresi
$T(path_{E2E})_\text{comp}^{BR}$	Servis isteği için BR'de bir uçtan uca yol hesaplama süresi

Simülasyon ayarlarında İSS'lerin, anahtarların ve denetleyicilerin toplam sayısı sırasıyla 6 ila 10, 36 ila 240 ve 6 ila 10 arasında değişmektedir. Formüllerdeki m , DRA ve HRA 'daki denetleyiciler arasında gönderilen çeşitli mesajları (örneğin, uçtan uca küresel yol talebi, ilgili giriş/çıkış düğümü bilgileri, uygulanabilir bir yol bulma ve bir talep için karar verileri, vb.) temsil etmektedir.

4.1.2 Deneysel Sonuçlar

Bu alt bölümde, QoSChain çerçevesinin performansını DRA ve HRA çerçevelerine kıyasla değerlendirmek için FST, MEP ve RS metriklerine göre yapılan testlerin sonuçları sunulmuştur.

4.1.2.1 Uçtan uca akış kurulum süresi (E2E FST)

Uçtan Uca Akış Kurulum Süresi (E2E FST), uçtan uca yol üzerindeki anahtarların akış tablolarında hizmet/akış talebini etkinleştiren ilgili akış kuralı girişlerini yükleme süresini ifade eder. Bu nedenle, FST metriğinin belirlenmesinde zaman alan çeşitli adımlar vardır. Bu metriğin değeri genellikle (i) ağ/topoloji tabanlı gecikmeler (örneğin, denetleyici ve anahtarlar arasındaki RTT (Round-Trip-Time) ve denetleyiciden denetleyiciye mesaj/paket yayılma süresi) ve (ii) anahtar/denetleyici tabanlı gecikmeler (örneğin, anahtar ve/veya denetleyici paket/mesaj işleme süresi ve denetleyicideki yol hesaplama süresi) tarafından belirlenir. Bu gecikmelerden herhangi biri yüksekse, ortaya çıkan akış kurulum gecikmesi de yüksek olur. Bu da hem kontrol hem de veri düzlemi seviyelerinde bir tıkanıklığa ve ağıda daha uzun bir yük devretme süresine neden olabilir. Bu nedenle FST, ölçeklenebilirlik boyutunun iyi bir göstergesi olduğu için YTA ağlarındaki yönlendirme çerçevelerinin performansını ölçmek için önemli metriklerden biridir.

QC çerçevesinde, bir hizmet/akış talebinin FST'si (FST_{QC}), genellik kaybı olmaksızın Denklem (1)'deki gibi tanımlanmıştır.

$$\begin{aligned}
 FST_{QC} = & \lambda(S_Req, S_Req, {}_f \vec{h}, {}_f \vec{s}, {}_f \vec{c}) + T(path_{E2E_{BC}})_{comp}^{\vec{c}} \\
 & + \max_{\{\forall_f^k c \in f C\}} \left\{ \gamma(P_Req, {}_f \vec{c}, {}_f^k c) + \gamma(P_Res, {}_f^k c, {}_f \vec{c}) \right. \\
 & + \max_{\{\forall_f^k s j \in f S\}} \left\{ T(\theta)_{prop}^{{}_f^k c \rightarrow {}_f^k s j} \right\} + T(P_Res)_{proc}^{\vec{c}} \Big\} \\
 & + \lambda(S_Res, S_Res, {}_f \vec{c}, {}_f \vec{s}, {}_f \vec{h})
 \end{aligned} \tag{1}$$

İlgili S_Req ve S_Res mesajlarının \vec{h} , \vec{s} ve \vec{c} arasındaki işlem ve yayılma süreleri Denklem (2) kullanılarak hesaplanır. Denklem (1), Denklem (3)'ü kullanarak P_Req, P_Res'in işlem ve yayılma sürelerinin maksimumunu bulur. Denklem (1)'de verildiği üzere, uçtan uca yol üzerindeki denetleyiciler ve anahtarlar arasında flow_mod mesajlarını da içerir.

$$\begin{aligned}
 \lambda(x_1, x_2, y_1, y_2, y_3) = & T(x_1)_{proc}^{y_1} + T(x_1)_{prop}^{y_1 \rightarrow y_2} + T(x_1)_{proc}^{y_2} \\
 & + T(x_2)_{prop}^{y_2 \rightarrow y_3} + T(x_2)_{proc}^{y_3}
 \end{aligned} \tag{2}$$

$$\gamma(x, y_1, y_2) = T(x)_{prop}^{y_1 \rightarrow y_2} + T(x)_{proc}^{y_2} \tag{3}$$

Benzer şekilde, HRA çerçevesindeki bir hizmet/akış talebinin FST'si (FST_{HRA}) Denklem (4)'teki gibi karakterize edilmiştir.

$$\begin{aligned}
 FST_{HRA} = & \delta(f p, \rho, f \vec{h}, f \vec{s}, f \vec{c}) + T(m)_{prop}^{f \vec{c} \rightarrow BR} \\
 & + \max \{\beta(m, path_L, f \vec{c}), \beta(m, path_L, f \vec{c})\} + T(path_{E2E})_{comp}^{BR} \\
 & + \max_{\{\forall_f^k c \in f C\}} \left\{ \beta(m, path_L, f^k c) + \max_{\{\forall_f^k s^j \in f S\}} \left\{ T(\theta)_{prop}^{f^k c \rightarrow f^k s^j} \right\} + T(m)_{proc}^{BR} \right\}
 \end{aligned} \tag{4}$$

Denklem (4)'te verildiği üzere, temel olarak şunları içerir: (i) Denklem (5)'te formüle edilen f akışının 1. paketi ve $f \vec{h}$, $f \vec{s}$ ve $f \vec{c}$ arasında karşılık gelen packet_in mesajının işlenme ve yayılma süreleri, (ii) Denklem (6)'da formüle edilen; maksimum (a) BR'den $f \vec{c}$ 'ye yerel yol talebi mesajlarının yayılma süresi artı $f \vec{c}$ 'deki yerel yol hesaplama süresi veya (b) Denklem (6) da $f \vec{c}$ için aynı prosedürler, (iii) BR'deki hizmet talebi için uygun bir uçtan uca yolu bulmak için hesaplama süresi ve (iv) uçtan uca yol üzerinde BR, denetleyiciler ve anahtarlar arasında yol talebi/kurulum mesajlarının ve flow_mod mesajlarının maksimum işleme ve yayılma süreleri.

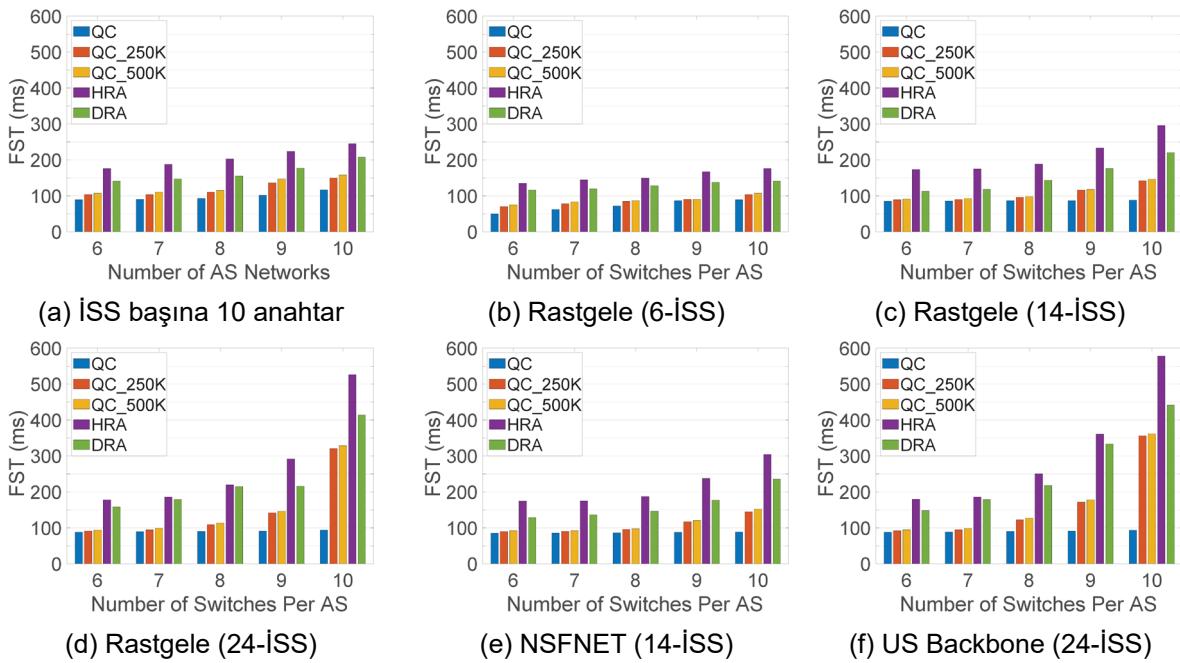
$$\delta(x_1, x_2, y_1, y_2, y_3) = \lambda(x_1, x_2, y_1, y_2, y_3) + T(x_2)_{proc}^{y_2} \tag{5}$$

$$\beta(x, P, y) = T(x)_{prop}^{BR \rightarrow y} + T(P)_{comp}^y + T(x)_{prop}^{y \rightarrow BR} \tag{6}$$

DRA'daki bir hizmet/akış talebinin FST'si (FST_{DRA}) ise Denklem (7)'deki gibi tanımlanmıştır.

$$\begin{aligned}
 FST_{DRA} = & \delta(f p, \rho, f \vec{h}, f \vec{s}, f \vec{c}) + \sum_{\substack{1 \leq k \leq |f C| \\ \forall_f^k c \in f C}} \left(T(path_L)_{comp}^{f^k c} \right. \\
 & \left. + T(m)_{prop}^{f^k c \rightarrow f^{k+1} c} \right) + \max \left\{ \sum_{\substack{\forall_f^k c \in f C \\ |f C| \leq k \leq 1}} \left(T(m)_{proc}^{f^k c} + T(m)_{prop}^{f^k c \rightarrow f^{k-1} c} \right), \right. \\
 & \left. \max_{\{\forall_f^k c \in f C\}} \left\{ \max_{\{\forall_f^k s^j \in f S\}} \left\{ T(\theta)_{prop}^{f^k c \rightarrow f^k s^j} \right\} \right\} \right\}
 \end{aligned} \tag{7}$$

Denklem (7)'de verildiği üzere, (i) Denklem (5)'te formüle edilen f akışının 1. paketinin ve ilgili packet_in mesajının $f \vec{h}$, $f \vec{s}$ ve $f \vec{c}$ arasındaki işlem ve yayılma süreleri, (ii) bir denetleyicideki yerel yol hesaplama süresi ve karar mesajının uçtan uca yolu üzerindeki bir sonraki denetleyiciye yayılma süresi, (iii) uçtan uca yolu üzerindeki denetleyiciler ve anahtarlar arasındaki yol rezervasyon mesajlarının ve flow_mod mesajlarının işlem ve yayılma sürelerinin maksimumundan oluşur. k 'nın bir ID değil, ilgili kümelerdeki bir indeksi temsil ettiğini not ediyoruz. Örneğin, $f^C = \{A, E, B, R\}$ 'nin f akışı için uçtan uca yolu üzerindeki denetleyiciler kümesi olduğunu varsayırsak, $\frac{1}{f}c = A, \frac{2}{f}c = E, \frac{3}{f}c = B, \frac{4}{f}c = R$.



Şekil 26: QoSChain (QC), DRA ve HRA'nın Akış Kurulum Süreleri (FST)

Şekil 26(a), İSS ağı başına 10 anahtar ve farklı hizmet talepleri (yani, farklı kaynak ve hedefler) ile İSS sayılarını değiştirirken FST değerlerini göstermektedir. QC, blokzincir altyapısı tarafından halihazırda tutulan bilgilerden ve daha fazla İSS'den kaynaklanan DRA ve HRA'nın daha yüksek hesaplamaları nedeniyle, DRA ve HRA'dan daha iyi FST değerlerine sahiptir. Simülasyonlarda DRA, FST açısından HRA'dan daha iyi performans göstermektedir. Bunun nedeni Broker, HRA'da uçtan uca yolu üzerinden tüm İSS denetleyicileriyle iletişim kurarken, DRA'da bir uçtan uca yolu oluşturmak için hedef İSS'ye yönelik BGP'nin sonraki-düğüm öznitelik değerleri aracılığıyla BGP tabanlı en kısa yol kullanılmaktadır.

Şekil 26(b)–Şekil 26(f), farklı hizmet talepleri kullanılarak (her talep, kaynak ve hedefin farklı İSS'lerden olduğu farklı kaynak-hedef çiftlerini içerir) 6-, 14-, 24-İSS Random, NSFNET ve U.S. Backbone ağ topolojilerinde anahtar sayısını İSS ağı başına değiştirirken QC, QC_250K, QC_500K, DRA ve HRA durumlarında toplam FST değerlerini göstermektedir. Şekillerde görüldüğü gibi, anahtar sayısı İSS ağına arttıkça QC çerçevesi, FST açısından DRA ve HRA yaklaşımlarını geride bırakmaktadır; çünkü DRA ve HRA yaklaşımlarının akış kurulumlarında daha fazla anahtar-denetleyici ve denetleyici-denetleyici gecikmesi bulunmaktadır. Bu durum, QC yaklaşımının İSS'lerin QoS ile ilgili durumlarını yansıtan mevcut blokzincir işlemlerini kullanarak, DRA ve HRA yaklaşımlarından daha az kontrol mesajı ile yönlendirme yolunu belirleme avantajından kaynaklanmaktadır. İSS ağına düşen farklı yolculuk sayılarındaki artış her zaman lineer olmamaktadır. Bir İSS'teki farklı parça yollarındaki artış, İSS'ye ait uçtan uca yolunu hesaplamak için blokzincirde aranması ve okunması gereken blokzincir işlemlerinin

sayısındaki bir artışa yol açar. QC_250K ve QC_500K'nın Şekil 26(d) ve Şekil 26(f) de ki 9 ve özellikle 10 noktalarındaki dalgalanmalar, yukarıda bahsedilen durumun bir sonucudur ve aynı zamanda topolojideki daha fazla İSS'yi içerir.

4.1.2.2 Gönderilen ve işlenen mesajlar (MEP)

Ağ denetleyicileri, düğüm sayısı arttıkça QoS temelli uçtan uca yolları kurmak için ağ cihazları ve diğer denetleyiciler arasında daha fazla Mesaj Gönderme ve İşleme (MEP) ile başa çıkmalıdır. Bu, bir hizmet talebi adına QoS tabanlı bir uçtan uca yol oluşturmak için gereken herhangi bir mesaj olabilir ve ağ ve iletişim fazlalığı olarak adlandırılabilir. Bu mesaj alışverişî ve işleme prosedürleri, denetleyicileri sınırlı hesaplama kaynaklarına sahip olmaları nedeniyle (yani CPU ve bellek) bir darboğaz haline getirebilir. Bu nedenle, ilgili karar verici denetleyiciler tarafından bir hizmet/akış talebi adına bir uçtan uca yol kurmak için gönderilen MEP sayısını en aza indirmek, denetleyiciler için önemlidir. Bu durum, MEP'i yönlendirme performansını ve YTA ağ sisteminin ölçeklenebilirliğini değerlendirmek için başka bir önemli metrik haline getirir.

QC çerçevesinde bir hizmet/akış talebi adına bir uçtan uca yol kurmak için gerekli olan MEP sayısı (MEP_{QC}), HRA çerçevesinde (MEP_{HRA}) ve DRA çerçevesinde (MEP_{DRA}) sırasıyla Denklemler (8)-(10)'da verilmiştir.

$$MEP_{QC} = 2 \cdot |{}_f C| + \sum_{\forall {}_f^k c \in {}_f C} |{}_f^k S| \quad (8)$$

Denklem (8)'de gösterildiği üzere, MEP_{QC} 'nin başlıca unsurları şunlardır: (i) bir S_Req mesajı, kaynak ana makineden (f^h) kaynak denetleyiciye (f^c) gönderilir, (ii) hizmet talebi için bulunan uçtan uca yol üzerindeki diğer denetleyicilere (f^c) tarafından gönderilen P_Req mesajları, (iii) hizmet talebi için bulunan uçtan uca yol üzerindeki denetleyicilerden P_Req mesajlarına yanıt olarak (f^c)'ye gönderilen P_Res mesajları, (iv) (f^c)'den (f^h)'ye gönderilen bir S_Res mesajı ve (v) denetleyicilerden anahtarlar üzerindeki ağları boyunca gönderilen flow_mod mesajları.

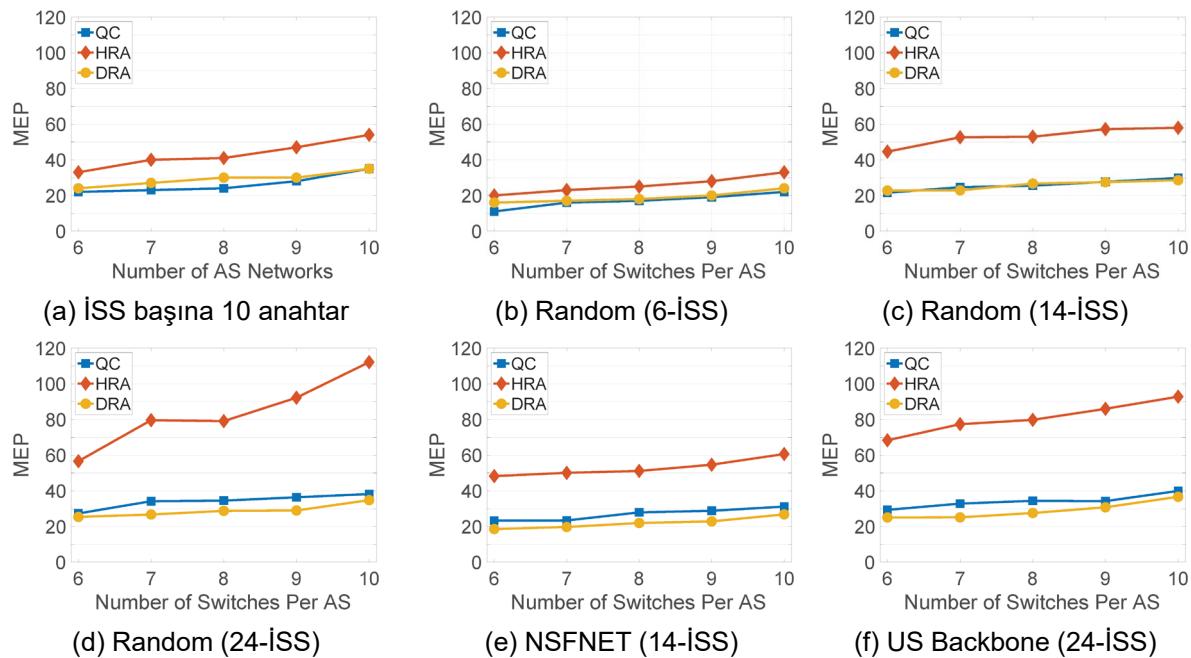
$$MEP_{HRA} = 4 + |B^f \vec{N}| + |B^{f \vec{N}}| + 2 \cdot |{}_f C| + \sum_{\forall {}_f^k c \in {}_f C} |{}_f^k S| \quad (9)$$

$f^{\vec{N}}$ ve $B^{\vec{N}}$ sırasıyla akış f için uçtan uca yolun kaynak ve hedef İSS'leridir. Benzer şekilde, Denklem (9)'da, MEP_{HRA} 'nın başlıca unsurları şunlardır: (i) kaynak anahtar (f^s) tarafından (f^c)'ye gönderilen bir packet_in mesajı, (ii) (f^c)'den BR'ye gönderilen bir uçtan uca yol isteği mesajı, (iii)

BR'den f^{c} 'ye ve hedef denetleyici (f^{d})'ye (her sınır düğümü ile kaynak/hedef anahtar çiftleri arasındaki yerel yolların duyuruları için) iki hesaplama isteği mesajı, (iv) f^{c} ver f^{d} den BR'ye her sınır düğümü ve kaynak/hedef anahtar çiftleri için parça yol duyurularını içeren mesajlar, (v) BR'den uçtan uca yol üzerindeki denetleyicilere (her ağdaki ilgili sınır düğüm çiftlerini içeren) gönderilen mesajlar, (vi) denetleyicilerden uçtan uca yol üzerindeki BR'ye (her ağdaki istenen parça yollar için) gönderilen onay mesajları ve (vii) denetleyicilerden uçtan uca yol üzerindeki ağlarındaki anahtarlara gönderilen flow_mod mesajları.

$$MEP_{DRA} = 1 + 2 \cdot |_f C| + \sum_{\forall_f^k c \in f C} |_f^k S| \quad (10)$$

Denklem (10)'da olduğu gibi MEP_{DRA} , başlıca olarak şunları içerir: (i) bir packet_in mesajı, kaynak anahtar (f^{s}) tarafından f^{d} 'ye gönderilir, (ii) talep edilen yol için uçtan uca yol boyunca denetleyiciler arasında gönderilen karar ve bildirim (yol rezervasyonu için) mesajları ve (iii) denetleyicilerden uçtan uca yol üzerindeki ağlarındaki anahtarlara gönderilen flow_mod mesajları.



Şekil 27: QoSChain (QC), DRA ve HRA'nın Gönderilen ve İşlenen Mesajları (MEP)

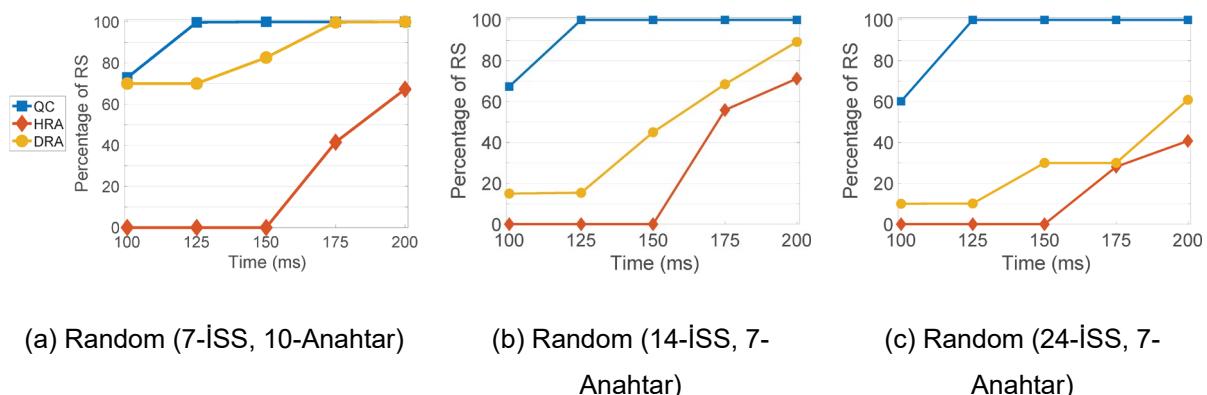
Şekil 27, QC, DRA ve HRA modellerindeki MEP değerlerini göstermektedir. Şekil 27(a)'da İSS sayıları İSS başına sabit anahtar sayısı (10) ile değişmemektedir. Şekil 27(b)–Şekil 27(f)'te ise, İSS başına düşen anahtar sayısı sırasıyla 6-, 14-, 24-İSS Random, NSFNET ve U.S. Backbone ağlarında değişmektedir. Şekil 27(a)'da HRA, tüm İSS durumlarında QC ve

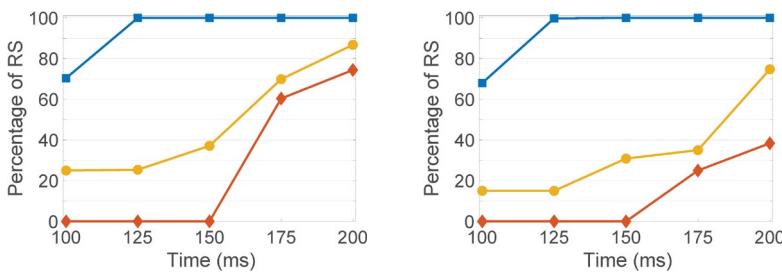
DRA'dan %50 daha fazla MEP gerektirir. Bu, HRA'nın Denklem (9)'da yansıtıldığı gibi İSS'lerde daha fazla sayıda sınır cihazı kullanmasından kaynaklanmaktadır.

Şekil 27(b)–Şekil 27(f), QC ve DRA çerçevelerinin, farklı ağ topolojileri altında İSS başına düşen anahtar sayısını değiştirmeye açısından HRA yaklaşımına göre daha iyi bir performans sergilediğini göstermektedir. Şekillerde, QC ve DRA'nın MEP değerleri, İSS başına düşen anahtar sayısı arttıkça neredeyse aynı değerlere sahiptir. Bu durumun bir nedeni, tüm çerçeveler için aynı talepte (yani, toplam anahtar ve denetleyici sayıları farklı değerlere sahip olabilir) uçtan uca yolun her zaman aynı olmaması olabilir. Diğer bir neden ise DRA'da bir İSS'in hedefe ulaşmak için yalnızca varsayılan BGP sonraki-düğüm İSS'ına sormasıdır, yani İSS düzeyinde en kısa yolu kullanır (sadece sonraki İSS ile iletişim kurar ve mesaj alışverişini yapar). Bu nedenle, bir İSS'nin, yalnızca varış noktasına giden varsayılan BGP sonraki-düğüm İSS'i hariç başka bir İSS ile herhangi bir iletişim ve mesaj alışverişi yapmasına gerek yoktur (daha az iletişim ve mesaj alışverişi). QC'nın MEP'i, DRA'nın kine benzer olsa da; DRA, İSS düzeyinde en kısa BGP yolunu kullanırken QC, Şekil 10'da gösterildiği gibi FST'yi önemli ölçüde azaltır. Ayrıca, Şekil 27(b)–Şekil 27(f)'de, her bir ayarlamada İSS-içi anahtar sayısı büyütükçe uçtan uca yol üzerindeki anahtar sayısını artır, bunun sonucunda MEP değerleri yükselmektedir.

4.1.2.3 Hizmet Verilen Talepler (RS)

QoSChain (QC), HRA ve DRA çerçevelerinde anahtar tablolarındaki her bir hizmet/akış talebi için ağ kontrol örnekleri tahsis etmek, FST'leri nedeniyle QC, HRA ve DRA çerçevelerinde tüm ağ için taleplerin karşılanma sayısını etkileyebilir. FST daha yüksek olduğunda, çerçeveler hizmet talebini daha uzun bir süre içinde kurabilir. Ayrıca, RS metriği FST parametresinin sonuçları için farklı bir bakış açısından doğrulama sağlamaktadır. Bu nedenle, birim zamana düşen RS sayısı, QoS tabanlı uçtan uca yol kurulumu durumunda yönlendirme çerçevelerinin ölçeklenebilirliği için başka bir önemli metriktir.





(d) NSFNET (14-İSS, 7-Anahtar) (e) US Backbone (24-İSS, 7-Anahtar)

Şekil 28: QoSChain (QC), DRA ve HRA'nın Hizmet Verilen Talepleri (RS)

Şekil 28, 7-, 14-, 24-İSS Random, NSFNET ve U.S. Backbone ağları için zaman aralığını (ms cinsinden) değiştirirken 800 hizmet talebinin RS yüzdesini göstermektedir. Zaman arttıkça, QC çerçevesi tüm hizmet taleplerini kolayca yönetebildiği görülmektedir. Ancak genel olarak, DRA çerçevesinin RS yüzdesi İSS yoğunluğu arttıkça azalır. Şekil 26'in FST değerlerinden beklenildiği gibi, Şekil 28, HRA'nın zaman 150 ms olana kadar hizmet taleplerini karşılayamadığını doğrular. Zaman 150 ms'yi aşlığında, HRA, Şekil 28(a)-Şekil 28(e)'de gösterildiği gibi talepleri karşılama işlemeye başlar. Başka bir deyişle, HRA'nın uçtan uca yolları bulması için daha yüksek FST'ye ihtiyacı vardır. HRA ve DRA performansları, daha küçük zaman değerlerine kıyasla birbirine daha yaklaşısa da belirgin bir fark devam eder ve DRA her zaman HRA'yı domine eder. RS metriğinin sonucu olarak, QC çerçeveleri tüm hizmet taleplerini daha kısa sürede kurabilirken, Şekil 28(b)-Şekil 28(e)'de gösterildiği gibi HRA ve DRA çerçeveleri tüm hizmet taleplerini karşılayamaz. Bu, QC çerçevesinin, İSS denetleyicileri tarafından zaten oluşturulmuş blokzincir işlemlerinin avantajını kullanarak İSS ağları üzerinde QoS tabanlı uçtan uca yol bulma yeteneğini içermesinden kaynaklanmaktadır.

4.2 Yazılım Tanımlı Ağlarda Trafik Yönetimi İçin Blokzincir Destekli Kaynak-Yönlendirmenin (Source Routing – SR) Uygulanması

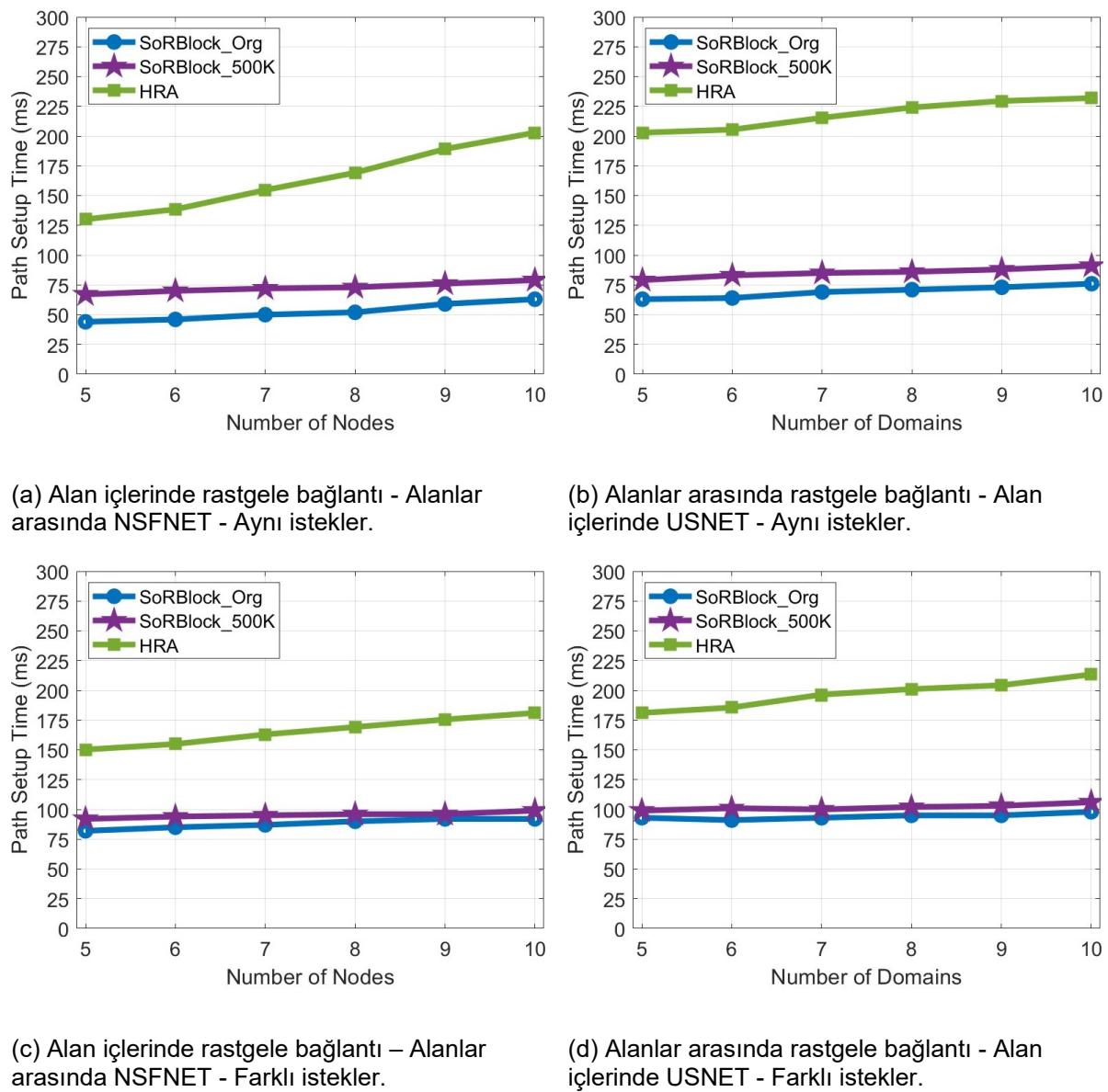
SoRBlock'un performansı, Yol Kurulum Süresi (Path Setup Time - PST) ve İşlenen Denetleyici Mesajları (Controller Messages Processed - CMP) parametrelerinin (Karakus & Durresi, 2015) de sunulan hiyerarşî tabanlı yönlendirme yaklaşımı olan Hiyerarşîk Yönlendirme Yaklaşımı (Hierarchical Routing Approach - HRA) ile analiz edilmesiyle test edilmiştir.

Deneýlerde, YTA ağ cihazlarını oluþturmak ve modellemek ve deneýler sırasında Tablo 2'de listelenen ilgili parametreler için sonuçları ölçmek için Mininet simülatörü kullanılmıştır. Sayısal hesaplamaları gerçekleştirmek için MATLAB yazılımı kullanılmıştır. Java'da oluşturulmuş özel bir blokzincir ağı, farklı ağ elemanı sayıları (yani veri düzlemi düğümleri ve denetleyiciler) ve iki farklı işlem boyutu (SoRBlock_Org olarak orijinal parça yol işlem sayısı ve

SoRBlock_500K olarak 500K işlem sayısını ile kullanılmıştır. Simülasyonlarda topolojilere bağlı olarak 25 ila 100 düğüm ve 5 ila 10 denetleyici arasında değişmektedir. Bir topoloji içindeki kenar düğümleri 2 ile 4 düğüm arasında değişmektedir. Bant genişliği kaynakları (1 Mbps) ile İSS'ler arası hizmet talepleri (yani bir hizmet talebinin başlangıç ve hedef düğümleri farklı etki alanlarındadır) dikkate alınmıştır. Bu üç seçenekin gerçek etkisini değerlendirmek için kaynak sıkıntısı nedeniyle hizmet reddini önlemek amacıyla bağlantılarında yeterli bant genişliği (100 Gbps) sağlanmıştır. Ayrıca, Erdos-Renyi modeli (Erdős P. and Rényi, 1964) altyapı düğümleri ve alan denetleyicileri için 0,5 bağlantı derecesine sahip ağları gelişigüzel oluşturmak için kullanılmıştır. Ayrıca, %95 istatistiksel anlamlılığı aşmak için her deney yirmi kez gerçekleştirilmiş ve her çalışmada aynı ve farklı hizmet talepleri durumları kullanılmıştır. Son olarak, tüm denemeler Oracle VirtualBox'ta Ubuntu 14.04 çalıştırın 12 GB RAM'e sahip bir Intel Core i7-5500 makinede gerçekleştirılmıştır.

4.2.1 Yol Kurulum Süresi (Path Setup Time - PST)

Yol Kurulum Süresi (Path Setup Time - PST), uçtan uca yolu boyunca alan giriş düğümlerinin akış tablolarındaki başlık bilgilerini değiştirmek Kaynak-Yönlendirme'yi etkinleştiren uygun akış kuralı kayıtlarını dağıtmak için gereken süredir. Dolayısıyla, PST ölçümüne birkaç aşama katkıda bulunur. Tipik olarak (i) denetleyici ve altyapı düğümleri arasındaki RTT ve denetleyiciden denetleyiciye mesaj/paket传递 için gereken süre gibi ağ/topoloji ile ilgili gecikme ve (ii) düğüm ve/veya denetleyici paket/mesaj işlem süresi ve denetleyici rota hesaplama süresi gibi düğüm/denetleyici ile ilgili gecikme ile karakterize edilir. Bu gecikmelerden herhangi birinin aşırı olması durumunda, ilgili yol kurulum gecikmesi artar ve düğümlerin akış tablolarında daha uzun bir akış kuralı yükleme, kaldırma veya değiştirme süresine neden olur. Daha sonra hem kontrol hem de veri katmanı seviyelerinde tıkanıklığa ve uzun bir ağ kurtarma süresine yol açabilir. Bu nedenle PST, genel ağ ölçeklenebilirliğine katkıda bulunduğuundan YTA ağlarındaki yönlendirme mimarilerinin verimliliğini değerlendirmek için önemli bir parametredir. PST hesaplamasında $PST_{SoRBlock}$ çerçevesi için Denklem (1), (2) ve (3)'nın uyarlanmış versiyonları ve PST_{HRA} çerçevesi için Denklem (4), (5), (6)'nın uyarlanmış versiyonları kullanılmıştır.



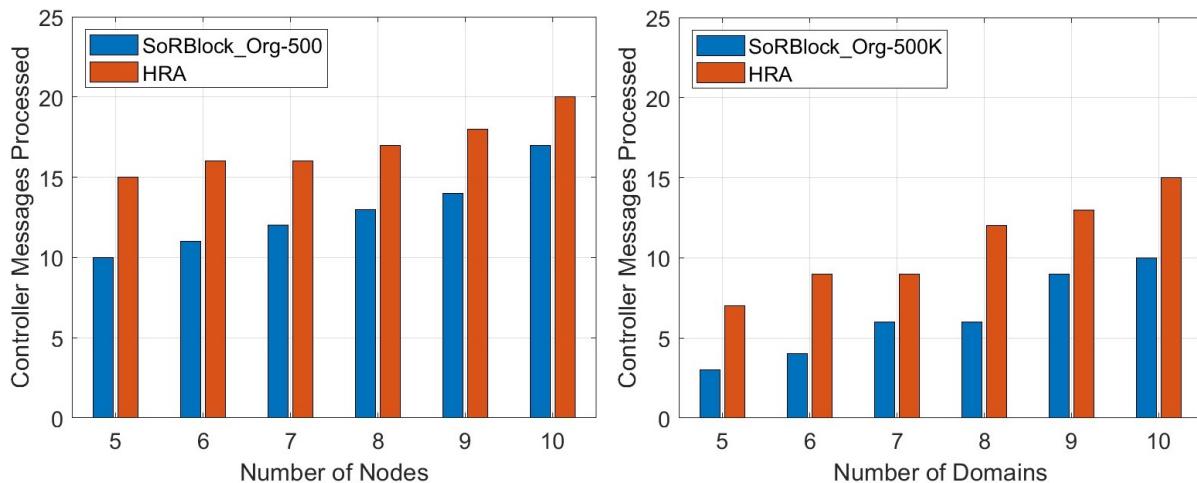
Şekil 29: SoRBlock ve HRA çerçevelerinin PST performansları

Şekil 29a ve Şekil 29c, SoRBlock'ta orijinal (SoRBlock_Org) ve 500K (SoRBlock_500K) işlem boyutları ve alanlardaki altyapı düğümlerinin sayısı değiştirilirken HRA şeması ile toplam yol kurulum sürelerini göstermektedir. Alan-içi düğümler rastgele bağlanırken, alanları arası bağlantılar NSFNET topolojisini taklit eder. Şekil 29a'daki tüm çalışmalar için aynı (yani, tam kaynak ve hedef) hizmet talepleri kullanılırken, Şekil 29c'de farklı hizmet talepleri (yani, farklı kaynak ve hedef) kullanılmıştır. Şekil 6a'da gösterildiği gibi, sunulan Kaynak-Yönlendirme mimarisi hem Şekil 29a hem de Şekil 29c'de HRA yaklaşımının düğüm-denetleyici ve denetleyici-denetleyici gecikmelerini artırması nedeniyle toplam PST'deki artırma oranı bakımından HRA yöntemini geride bırakmaktadır. Bunun nedeni, önerilen modelin uçtan uca rotasını belirlemek için mevcut işlemleri HRA yönteminden daha az kontrol mesajıyla kullanması ve akış kurallarının dağıtımını alanların giriş düğümleriyle sınırlamasıdır.

Ek olarak, Şekil 29b ve Şekil 29d, alan sayısını değiştirirken sırasıyla blokzincir ve HRA stratejilerinde orijinal ve 500K işlem boyutlarını kullanarak SoRBlock'taki genel yol kurulum süresini göstermektedir. Alan-içi düğümler USNET'ekine benzer bir şekilde bağlanırken, alanlar arası bağlantılar rastgele oluşturulmaktadır. Şekil 29b'de her iterasyon için aynı (yani, tam kaynak ve hedef) hizmet talepleri kullanılırken, Şekil 29d'de benzersiz hizmet talepleri (yani, farklı kaynak ve hedef) kullanılmaktadır. Tüm SoRBlock_Org ve SoRBlock_500K düzenlemeleri ve HRA için bulgularındaki alaı sayısı arttıkça genel yol kurulum süresi de artmaktadır. Bu durum, alan sayısının artmasının bir sonucu olarak ortaya çıkmakta ve her bir alandan gelen çok sayıda parça yol ve dolayısıyla işlem yüküne yol açarak yol hesaplama sürelerinin ve gecikmelerin artmasına neden olmaktadır. Şekil 29d'de Kaynak-Yönlendirme tabanlı şemalar Yol Kurulum Süresinde HRA stratejisinin yaklaşık iki katı kadar HRA yöntemini geride bırakırken, Kaynak-Yönlendirme tabanlı şemalar Şekil 29b'deki HRA performansını kolayca dört katına çıkarmaktadır. Öte yandan Şekil 29d, mevcut işlemlerden uçtan uca yolları hesaplanırken SoRBlock şemasının üstün performansını daha da doğrulamaktadır. Bunun nedeni SoRBlock'un yönlendirme yaklaşımının HRA şemasından daha hızlı olmasıdır.

4.2.2 İşlenen Denetleyici Mesajları (Controller Messages Processed - CMP)

Ağın boyutu düğümlere (örneğin uç noktalar, veri düzleme cihazları, denetleyiciler vb.) göre arttıkça, denetleyicilerin artan akış talepleriyle ve birden fazla etki alanında QoS tabanlı bir uçtan uca yolu oluşturmak için ağ düğümleri ve diğerleri arasında değişim tokuş edilen ve işlenen ilgili mesajlarla başa çıkması gerekebilir. Ancak, CPU ve RAM gibi denetleyicilerin kullanabileceği bilgi işlem kaynaklarının kısıtlı olması nedeniyle, bu mesaj alışverişi ve işleme görevleri denetleyicilerin darboğaz oluşturmmasına neden olabilir. Denetleyiciler, hizmet talep paketleri için bir uçtan uca rotası oluşturmak amacıyla değişim tokuş edilen ve işlenen mesaj sayısını en aza indirmelidir. Bu nedenle, İşlenen Denetleyici Mesajları (Controller Messages Processed - CMP), genel ağın ölçeklenebilirliğine katkıda bulunduğuundan, YTA ağlarındaki yönlendirme şemalarının performansını değerlendirmek için bir başka önemli göstergedir. CMP ölçüdü, bu araştırmada bir hizmet talebi paketleri için ilgili karar verici denetleyicilere, yani uçtan uca rotasındaki denetleyicilere ve Broker'a (varsayımsa) bir uçtan uca rotası oluşturmak için işlenen mesajların sayısını gösterir. CMP hesaplamasında $CMP_{SoRBlock}$ çerçevesi için Denklem (8)'in uyarlanmış versiyonları ve CMP_{HRA} çerçevesi için Denklem (9)'un uyarlanmış versiyonları kullanılmıştır.



(a) Alan içinde rastgele bağlantı – Alanlar arasında NSFNET.

(b) Alanlar arasında rastgele bağlantı – Alan içinde USNET.

Şekil 30: SoRBlock ve HRA çerçevelerindeki CMP

Şekil 30, önerilen çerçevede orijinal (SoRBlock_Org) ve 500K (SoRBlock_500K) işlem boyutları ve HRA şeması ile her bir etki alanındaki altyapı düğümlerinin sayısı değiştirilerek işlenen denetleyici mesajlarını göstermektedir. Alanı içi düğümler rastgele bağlanırken, alanlar arası bağlantıları Şekil 30a'daki NSFNET topolojisini taklit etmektedir. Alan-içi düğümler USNET topolojisini taklit ederken, alanlar arası bağlantıları Şekil 30b'de farklı hizmet talepleri (yani, farklı kaynak ve hedefler) altında rastgele bağlanır. Şekil 30a ve Şekil 30b, önerilen Kaynak-Yönlendirme tabanlı çerçevenin çeşitli etki alanı ve düğüm sayılarına göre HRA şemasından daha iyi bir performansa sahip olduğunu göstermektedir. Özellikle, Şekil 30b'de alan sayısı değiştiğinde HRA, önerilen çerçeveye göre yaklaşık %50 daha fazla denetleyici mesajıyla uğraşmaktadır. Bunun nedeni alanlardaki sınır düğümlerinin sayısıdır. İlginç bir şekilde, SoRBlock şemasında işlenen denetleyici mesajlarının sayısı, oran bakımından HRA çerçevesindekine yaklaşmaktadır. Aynı zamanda, Şekil 30a'da gösterildiği gibi alanlardaki düğüm sayısı artmaktadır, çünkü uçtan uca yolunda daha fazla düğüm kullanılmaktadır. Ayrıca, kullanılan rastgele hizmet talebi modeli ve alan-içi düğüm sayısı arttıkça uçtan uca yollarındaki düğüm sayısının artması nedeniyle denetleyici mesajlarının sayısı Şekil 30a'da daha yüksektir.

4.3 Çok Alanlı YTA'da Blokzinciri Tabanlı Yönlendirme Üzerinden Yol Seçim Stratejilerinin Etki Analizi

Bu bölümde, FFPS, RFPS, MHPS, FFPS_BGP ve MHPS_BGP QoS tabanlı uçtan uca yol bulma stratejilerinin, Akış Kurulum Süresi (Flow Setup Time-FST) olarak uçtan uca yolu hesaplamak için gereken süreyi kullanma ve karşılaştırma açısından etkinliğini ve uygulanabilirliğini gösteren simülasyonların sonuçlarını verilmektedir. Bu test sonuçları hazırlanırken QoS tabanlı uçtan uca yolu kurmak için iletişim mesajlarının ek yükü olarak Değişen ve İşlenen Mesaj Sayısı (Messages Exchanged and Processed - MEP), kaynaktan hedef ana bilgisayarlara tüm veri dağıtıımı için gerekli ağ kapasitesi olarak Ağ Kaynağı Tüketimi (Network Resource Consumption - NRC) ve aynı anda ağ kapasitesi ve gecikmenin etki analizi olarak Yol Bileşik Metriği (Composite Metric of a Path - CMP) metrikleri kullanılmıştır. Ağ çeşitlerinin ağlar arası iletişim üzerindeki etkisini değerlendirmek için, NSFNET ve US Backbone ağ topolojilerini İSS'ler arası seviyede uygulanmış ve ağ içi anahtar sayısı [5, 10] aralığında değişken tutulmuştur. Tüm ağ içi topolojiler Erdos-Renyi modeli (Erdős P. and Rényi, 1964) kullanılarak 0,8 derece bağlantı ile oluşturulmuştur. Her bir servis isteği için, [5, 25] aralığında bant genişliği gereksinimi olan bir talep kullanılmıştır. Ağ içi fiziksel bağlantılar [5, 55] aralığında bant genişliği kapasitesi sağlarken, testlerde İSS'ler arası her bir fiziksel bağlantı için bant genişliği kapasitesini desteklemek, ağ kaynağının sınırlamaları nedeniyle hizmet talebinin reddedilmesini göz ardı etmeyi ortadan kaldırmak için 10 Gbps oldukça yeterli olmuştur.

4.3.1 Akış Kurulum Süresi (Flow Setup Time - FST)

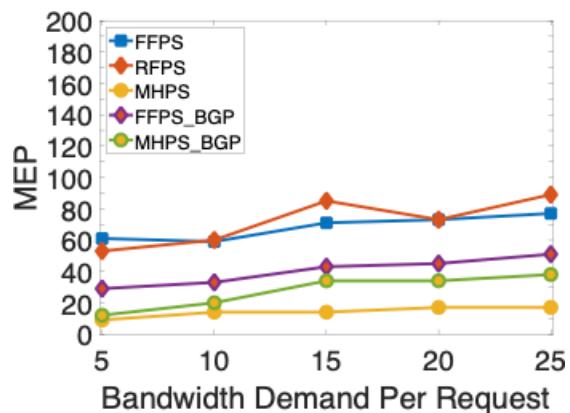
Akış Kurulum Süresi (Flow Setup Time-FST), QoS kullanan uçtan uca yol boyunca rezervasyon durum bilgisinin oluşturulabilmesi için geçmesi gereken süredir. Yayılma gecikmesini, işlem süresini ve yol hesaplamasını dikkate alındıdan, FST metriği Yazılım Tanımlı (YTA) ağlarının yönlendirmesini ve ölçeklenebilirliğini değerlendirmek için kullanılabilecek verimli bir istatistiktir.

Bir servis talebindeki artan bant genişliği talebinin ve ağ başına anahtar sayısının FST üzerindeki etkileri Şekil 31a-Şekil 33a ve Şekil 32a-Şekil 34a'da gösterilmektedir. Önerilen yol seçim stratejileri Şekil 31a'da benzer akış kurma sürelerine sahiptir. Buna karşılık, Şekil 32a'da gösterildiği gibi, daha düşük toplam anahtar sayısı, 9'dan az anahtar içeren ağlar için buna bağlı olarak daha düşük FST değerleri üremektedir. FST, anahtar sayısı arttıkça daha da artar ve bu artışa en önemli katkıyı yapan metrik MHPS olur. Bunun durum, daha fazla kullanılabilir

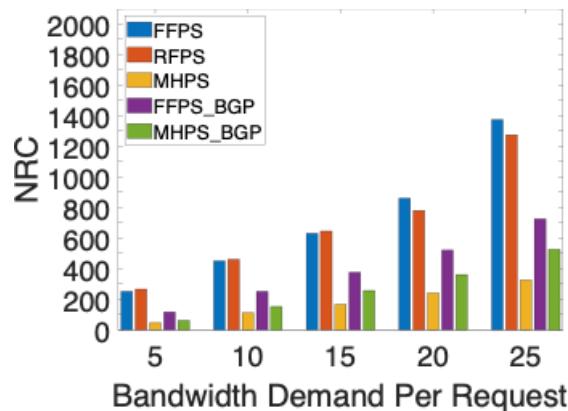
ağ kaynağına sahip blokzincirde daha fazla sayıda mevcut blokzincir işlemi olması nedeniyle en az atlamlı yolu seçmek için çok daha fazla iş yapması gerekeceğinden kaynaklanmaktadır.



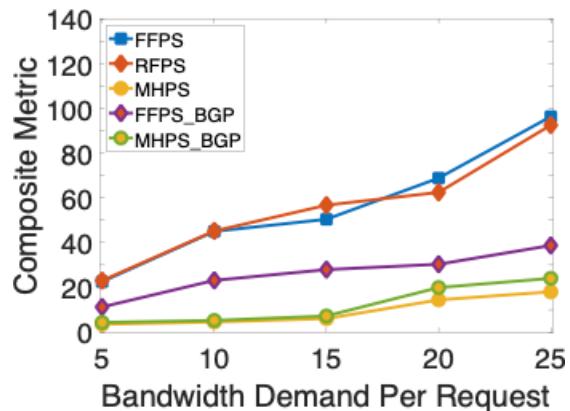
(a) Akış Kurulum Süresi (Flow Setup Time - FST)



(b) İletilen ve İşlenen Mesajlar (Messages Exchanged and Processed - MEP)



(c) Ağ Kaynağı Tüketimi (Ağ Kaynağı Tüketimi - Bant Genişliği)



(d) Bileşik Metrik (Bant Genişliği ve Gecikme)

Şekil 31: Bir Hizmet Talebinde artan bant genişliği talebi (NSFNET İSS Ağı)

Şekil 33a, önerilen MHPS ve MHPS_BGP yaklaşımlarının bir servis talebini karşılamak için daha yüksek FST'ye sahip olduğunu göstermektedir. Zira bu stratejiler, kaynak ana bilgisayardan hedef ana bilgisayara QoS tabanlı uçtan uca yol için kullanılan daha az sayıda atlamayı seçmek için mevcut blokzincir işlemlerini ararken, diğer yaklaşımalar blokzincirindeki mevcut işlemlerden ilk veya rastgele mevcut yolları seçmektektir. Benzer şekilde, anahtar sayısı arttığında, blokzincirde farklı yollara sahip daha fazla sayıda mevcut blokzincir işlemi olacaktır. Şekil 34a'da görüldüğü gibi, MHPS ve MHPS_BGP daha az sayıda atlama ile uçtan uca bir yol bulmak için blokzincirinde daha fazla sayıda işlem aradığı için FFPS, RFPS ve FFPS_BGP

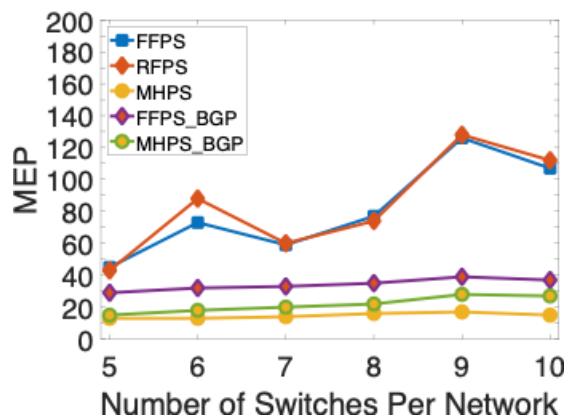
stratejileri minimum atlama yolu seçim stratejilerinden göre daha iyi performans göstermektedir.

4.3.2 İletilen ve İşlenen Mesajlar (Messages Exchanged and Processed – MEP)

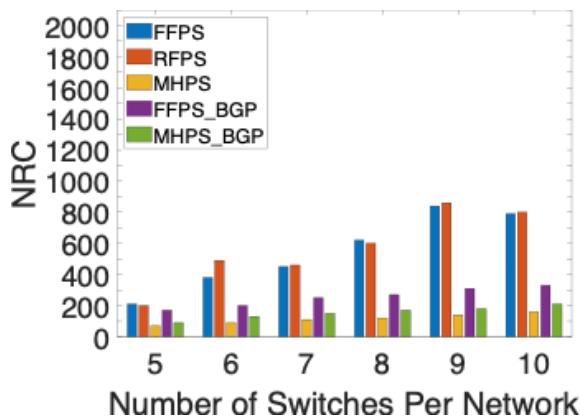
Her bir ağda kullanılan anahtar sayısı bakımından ağ boyutu artarken, ağ denetleyicileri ağ cihazları (ör. ağ cihazları, ana bilgisayarlar, vb.) arasında işlenen ve gönderilen akış taleplerinin ve ilgili mesajların sayılarındaki artışla başa çıkmak zorunda kalabilir. Ayrıca, birçok ağ kapsayan QoS tabanlı uçtan uca bir bağlantı kurmak için denetleyiciler birbirleriyle ve ağda hali hazırda bulunan diğer denetleyicilerle iletişim kurmalıdır.



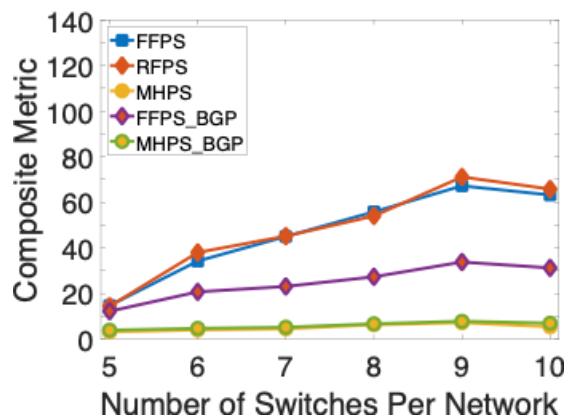
(a) Akış Kurulum Süresi (Flow Setup Time - FST)



(b) İletilen ve İşlenen Mesajlar (Messages Exchanged and Processed - MEP)



(c) Ağ Kaynağı Tüketimi (Ağ Kaynağı Tüketimi - Bant Genişliği)



(d) Bileşik Metrik (Bant Genişliği ve Gecikme)

Şekil 32: Ağ başına anahtar(lar) sayısının artırılması (NSFNET İSS Ağı)

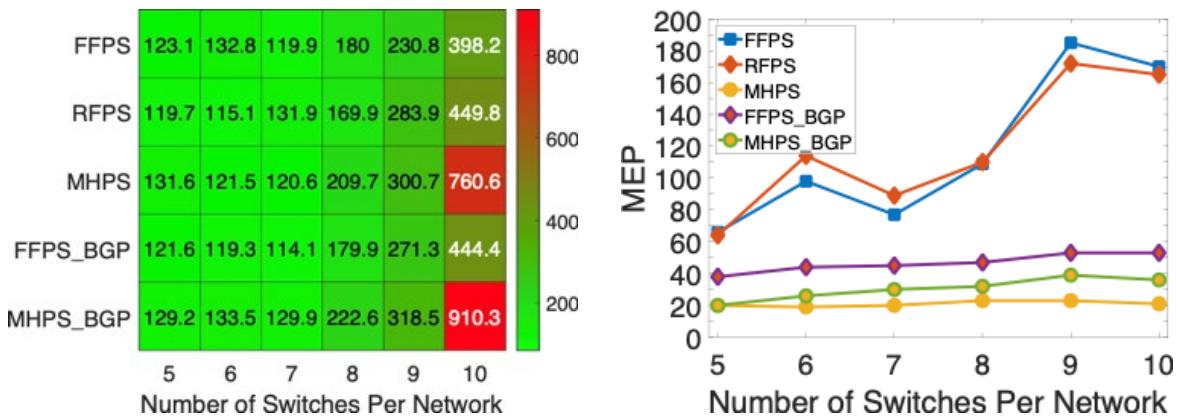
İSS ağlarının denetleyicileri, bu mesaj alışverişesi ve işleme operasyonlarının bir sonucu olarak CPU ve bellek gibi sınırlı bilgisayar ağı kaynakları nedeniyle bir tıkanma noktası haline

gelebilmektedir. Bir servis talebi akışı için QoS tabanlı uçtan uca bir yol sağlamak amacıyla, denetleyiciler işlenmesi ve alıp vermesi gereken toplam mesaj miktarını azaltmak için çaba gösterir. Sonuç olarak, çok alanlı YTA ağlarında yönlendirme çerçevelerinin etkinliğini analiz etmek amacıyla, Değiştirilen ve İşlenen Mesajlar (MEP) bir ağın genel manada ölçülebilirliğine katkıda bulunan bir diğer önemli etkendir.

Şekil 31b-Şekil 33b ve Şekil 32b-Şekil 34b FFPS, RFPS, MHPS, FFPS_BGP ve MHPS_BGP için, FST testlerinde kullanılmışa benzer bir konfigürasyonda, servis talebi başına gereken bant genişliği miktarları ve NSFNET ve US Backbone ağ topolojileri için her bir ağda kullanılan anahtar sayısı ile birlikte, İSS'ler arası seviyede MEP grafiklerini göstermektedir. Şekillerden de görülebileceği gibi, MHPS yaklaşımı MEP ölçütü açısından ve özellikle bant genişliği talebi ve her bir ağdaki anahtar sayısı arttıkça diğer tüm yönlendirme tekniklerinden daha iyi performansa göstermektedir. Bunlarla birlikte, FFPS_BGP ve MHPS_BGP, FFPS ve RFPS stratejilerine göre daha iyi bir MEP performansına sahiptir çünkü FFPS_BGP ve MHPS_BGP yaklaşımı ağ düzeyinde BGP tabanlı en kısa yolu kullanmaktadır. Bununla birlikte, MHPS_BGP stratejisi ağ düzeyinde BGP tabanlı her bir yolun İSS-içi daha az sayıda atlamaya sahip olmasını garanti edemezken, MHPS yaklaşımı mevcut tüm işlemleri kullanarak kaynak ana bilgisayardan hedef ana bilgisayara QoS tabanlı uçtan uca bir yol seçmeye zorlar. Başka bir deyişle, diğer ağlara ek mesajlar göndermeye ihtiyaç duymayan diğer yönlendirme sistemlerinin aksine, MHPS stratejisi uçtan uca bir yol oluşturmak için atlanacak en uygun atlama sayısını belirlerken kaynak kullanılabilirliğini dikkate alan aktif ve belirgin bir yaklaşım benimser. Hem FFPS hem de RFPS, sırasıyla ilk ve rastgele uygulanabilir yolu seçikleri için daha fazla iletişimle sonuçlanır. Bu bağlamda MHPS, rotaların oluşturulması için başlangıçta harcanan daha uzun süreyi, MEP ölçütü açısından daha yüksek bir etkinlik düzeyine ulaşarak telafi etmektedir.

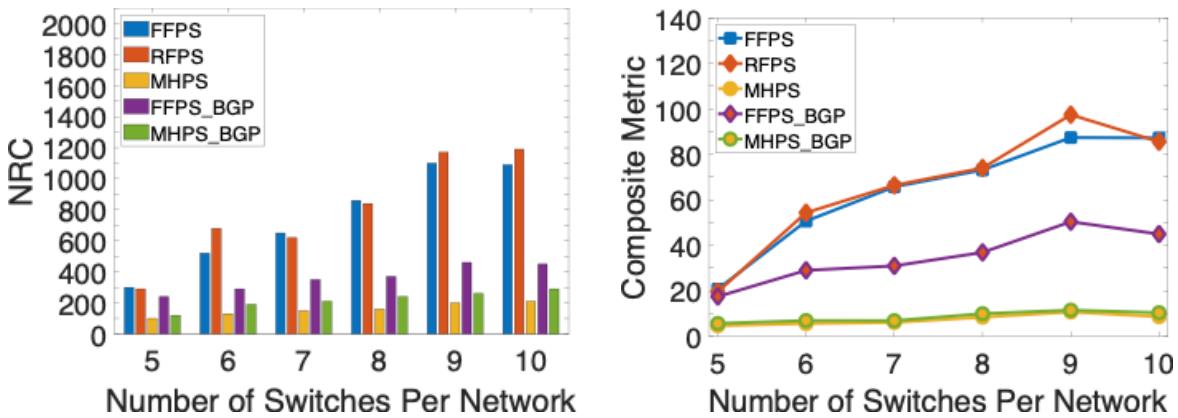
4.3.3 Ağ Kaynağı Tüketimi (Network Resource Consumption-NRC)

Uçtan uca yol talepleri için hizmet kalitesini kurmak, yönetmek ve izlemek için ağda mevcut kaynakları kullanmak, ağın nasıl uygulandığının bir başka göstergesidir. İSS'ler arası ağlarda uçtan uca bir yol oluşturmak için sınırlı görünürlüğe sahipken, çeşitli hizmetler ve uygulamalar için ağ performansı sorunlarını ve QoS'yi ele almanın en iyi yollarından biri ağ akışını (yani bant genişliğini) izlemektir. Bu nedenle, yol seçim stratejilerini analiz etmek için, bir başka önemli performans ölçütü olan Ağ Kaynağı Tüketimini (Network Resource Consumption-NRC), yaklaşık bir tahmin olarak sıkıştırılmış bant genişliği-atlama sayısı çarpımı aracılığıyla kullanılmıştır.



(a) Akış Kurulum Süresi (Flow Setup Time-FST)

(b) İletilen ve İşlenen Mesajlar (Messages Exchanged and Processed - MEP)



(c) Ağ Kaynağı Tüketimi (Ağ Kaynağı Tüketimi-Bant Genişliği)

(d) Bileşik Metrik (Bant Genişliği ve Gecikme)

Şekil 33: Ağ başına Ağ başına artan anahtar(lar) sayısı (USNET İSS Ağı)

Şekil 31c-Şekil 33c ve Şekil 32c-Şekil 34c sırasıyla NSFNET ve US Backbone inter-İSS ağ topolojileri ile bir hizmet talebi için talep edilen bant genişliğini ve her bir ağ için anahtar sayısını değiştirirken toplam ağ kaynağı tüketimini göstermektedir. Sayısal verilerden de görülebileceği gibi, MHPS birçok ağı kapsayan QoS tabanlı uçtan uca bir bağlantı oluştururken NRC'yi belirgin bir şekilde azaltmaktadır. Bunun nedeni, daha önce de belirtildiği gibi, MHPS'nin bir servis talebi için bir yol seçerken en düşük atlama sayısına sahip olanı bulmak için mevcut tüm blokzincir işlemlerini araması, diğer yaklaşımın ise yolu daha hızlı kurmak için ilk mevcut ve rastgele uygun işlemi seçmesidir. Şekil 31c ve Şekil 33c'de, bir hizmet talebinin bant genişliği talebi 10'dan yüksek olduğu aralıkta MHPS yaklaşımı diğer yaklaşılardan en az %70 daha iyi performans sergilemektedir. Ayrıca, MHPS ve MHPS_BGP'nin İSS'ler arası benzer yol seçimi kullandığı görülürken, MHPS blokzincirdeki mevcut tüm blokzincir işlemlerini kullanarak tüm ağ için daha az sayıda atlama ile uçtan uca bir yol seçmeyi tercih etmektedir. Benzer

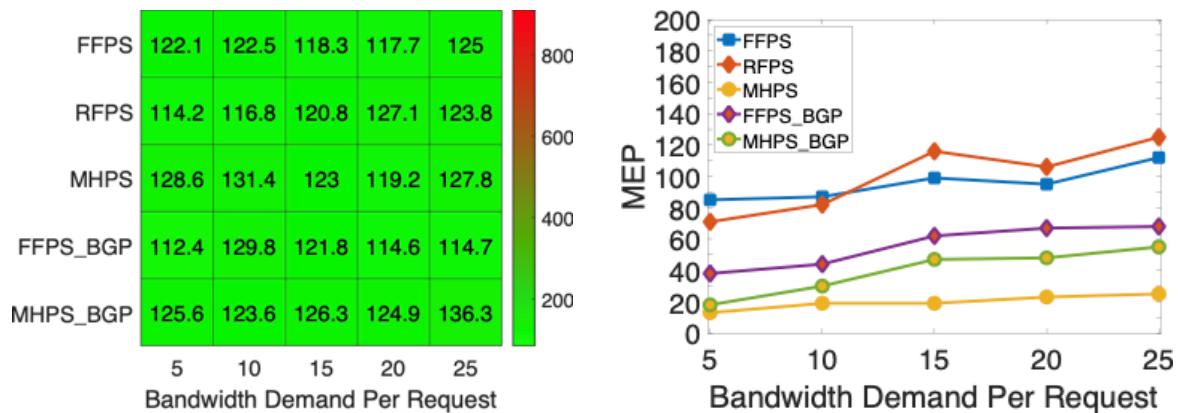
şekilde, Şekil 32c ve Şekil 34c'de, MHPS, ağ içi 6'dan fazla anahtar olduğunda diğer yaklaşılardan %50'ye varan oranda daha iyi performansa ulaşmaktadır.

4.3.4 Bir Yolun Bileşik Metriği (Composite Metric of a Path-CMP)

NRC'nin bir diğer olası göstergesi, yol oluşturulurken taleplerin yaşadığı gecikme miktarını dikkate almaktadır. Buna ek olarak, bu, kullanıcı deneyiminin kalitesinin genel olarak iyileştirilmesine de katkıda bulunacaktır. Bir Yolun Bileşik Metriği (Composite Metric of a Path-CMP), seçilen çeşitli yollar tarafından sergilenen bant genişliği ve gecikme değerlerinin kombinasyonu için bizim önerimizdir. CMP, Denklem (11) de gösterildiği gibi hesaplanır. Ağın tamamındaki her bir bağlantı ij için C_{ij} Denklem (11) de hesaplanır, burada fiziksel bir bağlantının bant genişliğini kapasitesini gösterir ve fiziksel bir bağlantının gecikmesi d_{ij} olarak temsil edilir. QoS parametrelerinin önem faktörünü belirlemek için Denklem (11) de $[0, 1]$ aralığındaki α ve β değerlerini kullanılmıştır.

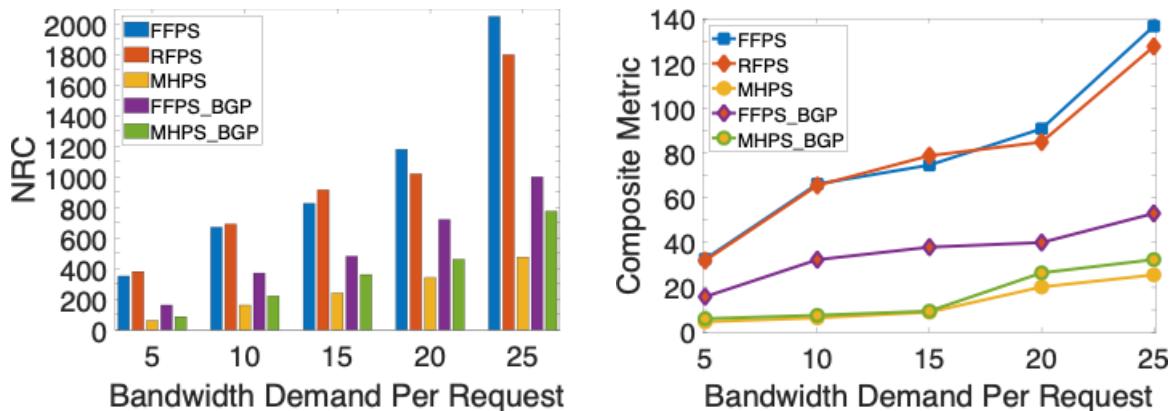
$$C_{ij} = \alpha * \frac{1}{b_{ij}} + \beta * d_{ij}, \forall ij \in I \quad (11)$$

$$\alpha + \beta = 1, \alpha, \beta \in \mathbb{R}^+$$



(a) Akış Kurulum Süresi (Flow Setup Time-FST)

(b) İletilen ve İşlenen Mesajlar (Messages Exchanged and Processed - MEP)



(c) Ağ Kaynağı Tüketimi (Ağ Kaynağı Tüketimi-Bant Genişliği) (d) Bileşik Metrik (Bant Genişliği ve Gecikme)

Şekil 34: Bir Hizmet Talebinde artan bant genişliği talebi (USNET İSS Ağı)

Şekil 31d-Şekil 33d ve Şekil 32d-Şekil 34d de ve değerlerini sırasıyla 0,7 ve 0,3 olarak ayarlarken, aynı zamanda bir talebin bant genişliği gereksinimini ve her bir ağdaki anahtar sayısını önceki bölümlerde yapılan benzer şekilde değiştirilmiştir. Ağlar arası QoS tabanlı bir uçtan uca kanal oluştururken aynı zamanda gerekli bileşik metriği en aza indirme hedefine ulaşmak için, ağdaki servis talebinin bant genişliği gereksinimini karşılayabilen her bağlantının bileşik metriğini hesaplıyoruz. Uçtan uca yoldaki atlama sayısını en aza indirmeye benzer şekilde, MHPS ve MHPS_BGP stratejileri, servis talebinin bant genişliği taleplerini ve ağ başına anahtar sayısını artırırken diğer yaklaşımlardan daha iyi performansa sahiptir. Bunun nedeni MHPS ve MHPS_BGP'nin mevcut tüm blokzincir işlemlerini kullanarak en az sayıda atlama ile QoS parametrelerini karşılayan uçtan uca bir yol seçmeye odaklanmasıdır. Böylece, Şekil 31d ve Şekil 33d de, bu MHPS ve MHPS_BGP teknikleri, bir servis talebinin bant genişliği talebi 20'ye kadar iken benzer sonuçlara sahiptir. Bant genişliği talebi 20'den yüksek olduğunda, MHPS yaklaşımı tüm ağı göz önünde bulundurarak minimum atlama yolunu bulması sayesinde MHPS_BGP'den biraz daha iyi sonuçlara sahiptir. Benzer şekilde, Şekil 32d ve Şekil 34d'de, MHPS ve MHPS_BGP yaklaşımı, ağ başına anahtar sayısını artırırken CMP metriğinde diğer stratejilerden en az %100 daha iyi performans göstermektedir. Bunun sonuçlarının nedeni, MHPS ve MHPS_BGP'nin CMP metriği ile en az atlama sayısına sahip bir yol seçmesidir.

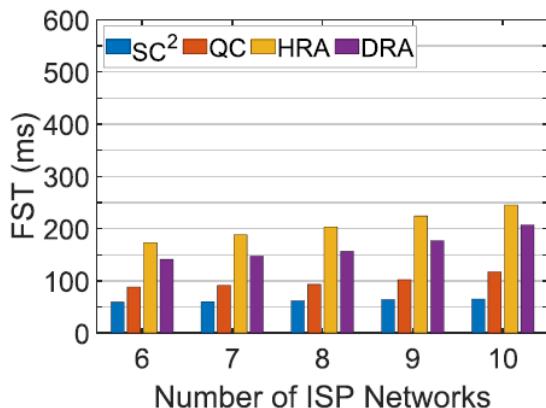
4.4 SmartContractChain (SC²): SDN ve Blokzincir ile Çoklu-İSS QoS Trafik Yönetim Çerçeveesi

SC² çerçevesinin performansını YTA ağlarında üç QoS tabanlı yönlendirme yöntemi ile karşılaştırılmıştır: Önceki bölümlerde açıklanan QoS destekli İSS'ler arası yönlendirme çerçevesi, *QoSChain* (QC) (Karakus vd., 2021), Hiyerarşik Yönlendirme Yaklaşımı (Hierarchical Routing Approach-HRA) (Karakus & Durresi, 2015) ve geleneksel ağların mevcut çalışma modunu yansıtan Dağıtılmış Yönlendirme Yaklaşımı (Distributed Routing Approach-DRA). DRA, İSS düzeyinde en kısa yol seçimini kullanırken BGP yönlendirme mekanizmasını kullanır. Böylece, bu yöntem benzer yolları neredeyse aynı yerel tercih, ağırlık ve toplam adreslerle tespit edebilmektedir.

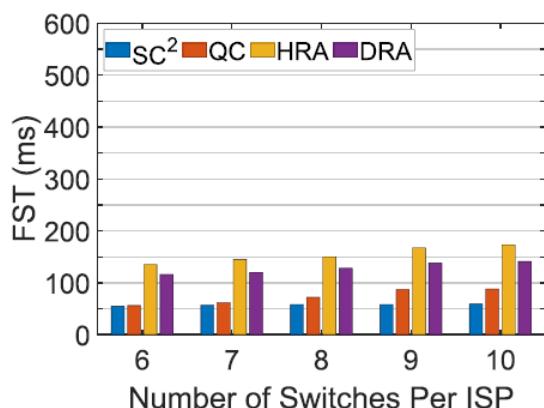
SC² çerçevesini diğer yaklaşımalarla karşılaştırmak için Uçtan Uca Akış Kurulum Süresi (E2E Flow Setup Time - FST) ve Değişen ve İşlenen Mesajlar (Messages Exchanged and Processed - MEP) performans metrikleri kullanılmıştır. Simülasyonlarda, önerilen SC², QC, HRA ve DRA yaklaşımlarının topoloji açısından hassasiyetini test etmek için İSS-içi anahtar sayısını [5, 10] aralığında değişken tutarken, İSS düzeyinde Erdos-Renyi rastgele grafik modeli (Erdős P. and Rényi, 1964), NSFNET ve US Backbone ağ topolojileri ile 0,5 bağlantı derecesine sahip Random-14 ve Random-24 ağ topolojilerini kullanılmıştır. Simülasyonlarda, her bir İSS arası servis talebinin bant genişliği talebi 1 Mbps olarak ayarlanmıştır ve İSS'ler arası sistemin altyapısı, ağ kaynağı sınırlamaları nedeniyle servis taleplerinin reddedilmesini göz ardı ederken, fiziksel bağlantınlarda yeterli bant genişliği (yani 100 Gbps) sağlamaktadır. Bunun üstüne, her bir İSS ağında 10 İSS-içi anahtar bulunurken, [6, 10] aralığında çeşitli sayıda İSS denetleyicisine sahip özel bir blokzinciri ağı kurulmuştur.

4.4.1 Akış Kurulum Süresi (Flow Setup Time – FST)

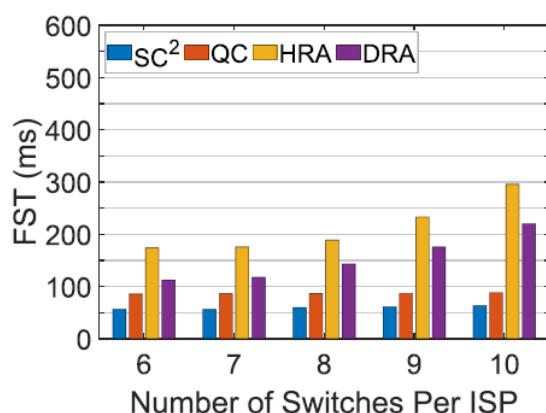
Yol boyunca rezervasyon durumu bilgilerinin ayarlanması için gereken süreyi, YTA veri düzlemi cihazlarının akış tablosundaki akış kuralı girişlerini yükleme süresini temsil eden Uçtan Uca Akış Kurulum Süresi (E2E FST) olarak belirtilmiştir. Hizmet isteğini etkinleştirmek için bir hizmet isteğinin yolu boyunca, FST, SDN ağlarındaki trafik yönetimi çerçevelerinin QoS'sını ve yönlendirme performansını değerlendirmek için önemli bir ölçümdür çünkü yayılma gecikmesini, işlem süresini ve yol hesaplama gecikmesini içerir. Bir ağa FST azaltılırsa hizmet sağlama gecikmesi de azalır, böylece QoS iyileşir.



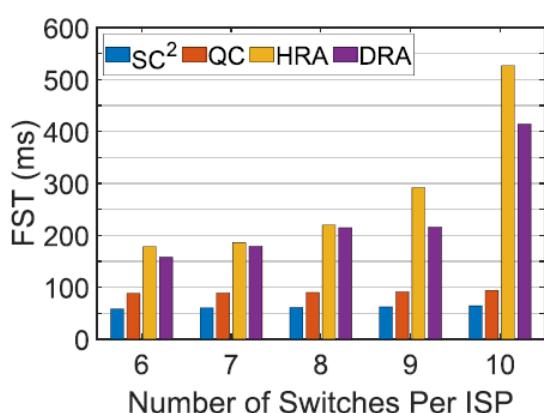
(a) Her İSS için 10 anahtar



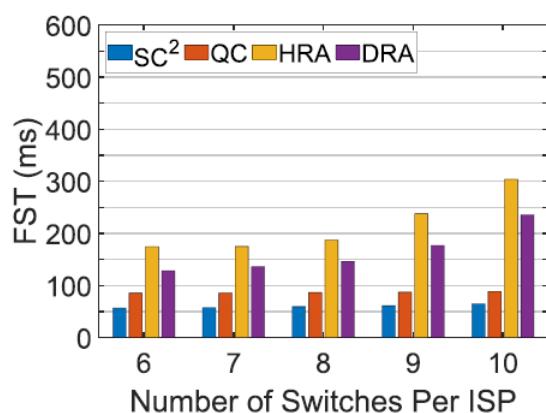
(b) Random (6-İSS)



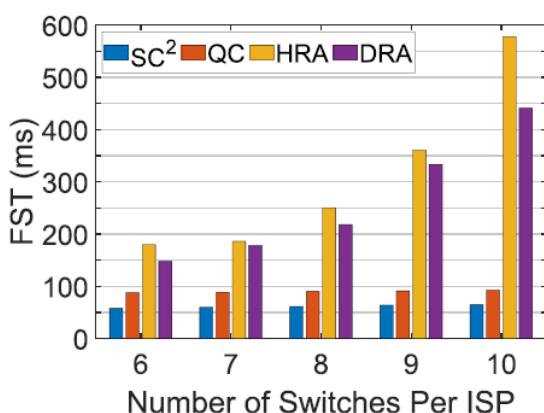
(c) Random (14-İSS)



(d) Random (24-İSS)



(e) NSFNET (14-İSS)



(f) US Backbone (24-İSS)

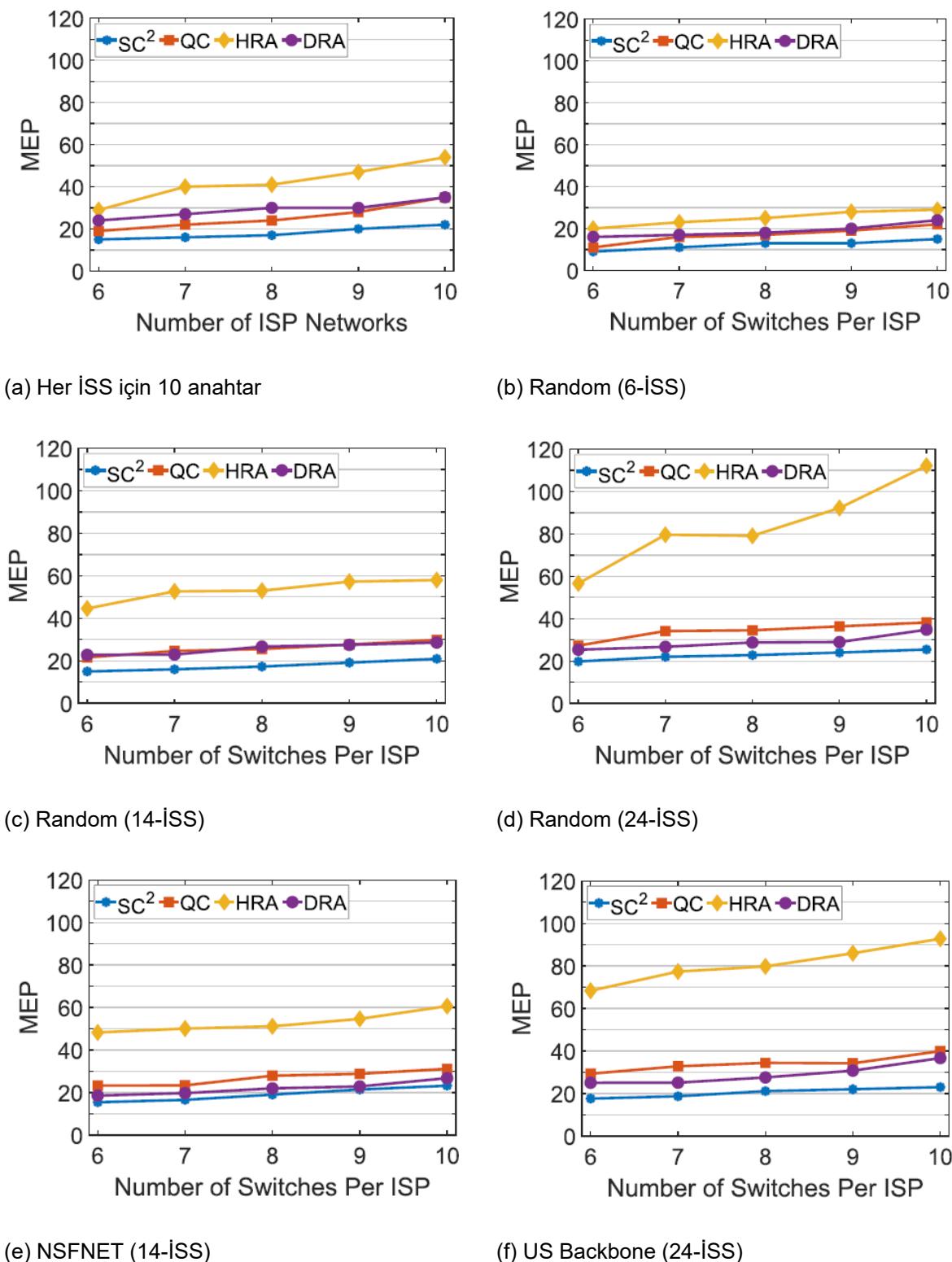
Şekil 35: SC², QC, HRA ve DRA'nın Akış Kurulum Süresi (FST)

Şekil 35, İSS ağı başına 10 adet sabit sayıda anahtarla İSS ağlarının sayısını değiştirirken FST değerlerini ve farklı hizmet isteklerini (kaynak ve hedefin farklı İSS ağlarında olduğu farklı kaynak-hedef çiftleri) kullanarak sabit sayıda İSS ağıyla her bir İSS ağındaki

anahtar sayısını göstermektedir. Tüm şemalar için, şekilde İSS sayısı arttıkça FST değeri de artar. Bunun nedeni, daha fazla sayıda İSS'nin, her bir İSS ağından gelen daha fazla yolcu yüküne yol açmasıdır. Şekil 35b-Şekil 35f'de, farklı İSS'ler arası ağ topolojileri (örneğin, Random-14, -24, NSFNET ve US Backbone) için sabit sayıda İSS ağıyla her bir İSS ağındaki anahtar sayısını değiştirirken FST değerleri verilmiştir. Şekillerde görüldüğü gibi SC² yaklaşımı, toplam FST'deki artış oranı açısından diğer yaklaşılardan daha iyi performans gösterir. Bunun nedeni SC²'nin akış kurulum süreci, diğer yaklaşımlara kıyasla minimum sayıda kontrol mesajıyla QoS tabanlı E2E yolunu seçmek için mevcut işlemlerden yararlanır. Bu da daha az anahtar-denetleyici ve denetleyici-denetleyici iletişim süresi demektir. Diğer taraftan, FST değeri anahtar sayısı arttıkça artar. Bunun nedeni, daha fazla sayıda İSS'nin, her bir İSS ağından gelen daha fazla yolcu yüküne yol açmasıdır. Açıkça görüldüğü gibi, SC², FST bakımından QC'den %33'e kadar ve diğerlerinden en az %50 daha iyi performans göstermektedir. Çünkü SC² akış kurulumu, doğası gereği, anahtar-denetleyici, denetleyici-denetleyici vb. gibi cihazlar arasındaki azalan sayıdaki kontrol mesajlarından süreyi kısaltmak için akış kurulum sürecini birleştirir.

4.4.2 İletilen ve İşlenen Mesajlar (Messages Exchanged and Processed - MEP)

İletilen ve İşlenen Mesaj (MEP), YTA ağlarındaki ağ denetleyicileri tarafından İletilen, alınan ve işlenen mesajların sayısını tanımlar. Sunucu, cihaz ve kullanıcı sayısı gibi ağ boyutları büyündüğünde ölçeklenebilirliğin yanında getirdiği bir başka zorluk da kaynakları zorlayabilecek QoS taleplerinin kurulumu, yönetimi ve sürdürülmesine ilişkin mesaj karmaşıklığıdır. YTA'da ki denetleyiciler, ağ boyutu büyündüğünde (yani sunucuların, ağ cihazlarının vb. sayısı arttığında) farklı İSS'ler arasında talep edilen QoS tabanlı uçtan uca yolları kurmak için artan akış talepleriyle başa çıkmak ve ağ cihazları arasında daha fazla mesaj iletmek, almak/işlemek zorunda kalabilir. Bu nedenle, ağ denetleyicileri tüm ağı yöneten ve yeni paketleri işleyen ağ bileşenleri oldukları için önemlidir. Bu cihazlar ağ görevlerini işlemek için yoğun bir şekilde çalışmaktadır. Ancak, diğer tüm ağ cihazları gibi işlemci, bellek vb. sınırlı kaynaklar nedeniyle sınırlı işlem kapasitesine sahiptirler. Bu cihazlar servis talepleri için uçtan uca bir yol oluştururken ağ denetleyicileri üzerindeki yükü azaltmak, ağdaki daha fazla servis talebine yanıt vermelerini ve bunları ele almalarını sağlayarak ağdaki QoS'i arttırmır. Bundan dolayı, YTA'da ki trafik yönetimi çerçevelerinde QoS performansını ölçmek için bir diğer önemli metrik, ağ denetleyicilerindeki kaynakları çok kullanması nedeniyle MEP'dir.



Şekil 36: SC², QC, HRA ve DRA'nın İletilen ve İşlenen Mesaj (MEP) sayıları

SC² çerçevesinin performansını MEP metrikleri ile değerlendirebilmek için Şekil 36'da SC², QC, DRA ve HRA modellerinde FST deneylerine benzer bir ortamda MEP değeri tekrar

gösterilmektedir. Şekil 36a'da, İSS başına sabit 10 adet anahtar sayısı ile ağdaki İSS sayısını değiştirirken MEP değeri ölçülmüştür. Benzer bir şekilde, Şekil 36b-Şekil 36f'de NSFNET ve US Backbone topolojileri ile tasarlanan rastgele ağlar üzerinde sırasıyla 6-, 14-, 24-İSS kullanan ağlar üzerinde İSS başına anahtar sayısının artırılmasının MEP performansı üzerindeki etkisini göstermektedir. Şekil 36a, HRA'nın diğer yöntemlere göre %50 daha fazla MEP gerektirdiğini, önerilen SC² yönteminin ise tüm İSS durumlarında en düşük MEP değerine sahip olduğunu göstermektedir. Şekil 36b-Şekil 36f'de, SC² çerçevesi, İSS ağı başına anahtar sayısı değişken tutulurken farklı İSS topolojilerinde diğer yaklaşımlardan bir miktar daha iyi performansa sahiptir. Bunun nedeni QC'nin blokzincir işlemlerini kullanarak İSS düzeyinde en kısa yolu kullanması, DRA'nın BGP sıradaki atlama yöntemi ile yol bulmasından kaynaklanır. Bu performans grafiklerinden de görülebileceği gibi, SC², QC ve DRA çerçeveleri için MEP performansı, her ortamda artan İSS-içi anahtarlar için uçtan uca yollar üzerindeki anahtar sayısını artırırken çok yakındır, ancak SC² çerçevesi yine de Şekil 35'de gösterildiği gibi daha düşük FST değerlerinin ek avantajına sahiptir. HRA'nın diğer yöntemlere göre %50 daha fazla MEP gerektirdiğini, önerilen SC² yönteminin ise tüm İSS durumlarında en düşük MEP değerine sahip olduğunu görülmektedir. Şekil 36b-f'de, SC² çerçevesi, İSS ağı başına anahtar sayısı değişken tutulurken farklı İSS topolojilerinde diğer yaklaşımlardan bir miktar daha iyi performansa sahiptir. Bunun nedeni QC'nin blokzincir işlemlerini kullanarak İSS düzeyinde en kısa yolu kullanması, DRA'nın BGP sıradaki atlama yöntemi ile yol bulmasından kaynaklanır. Bu performans grafiklerinden de görülebileceği üzere SC², QC ve DRA çerçeveleri için MEP performansı, her testte artan İSS-içi anahtarlar için uçtan uca yollar üzerindeki anahtar sayısını artırırken çok yakındır, ancak SC² çerçevesi yine de Şekil 35'de gösterildiği gibi daha düşük FST değerlerinin ek avantajına sahiptir.

Sonuç olarak, çalışmada önerilen SC² yöntemi, diğer yönlendirme stratejilerinden farklı olarak, SC aracılığıyla dahili bir otomatik kaynak kullanılabilirliği kontrolü gerçekleştirmektedir. Böylece SC², ağ kaynağının kullanılabilirliğini teyit etmek için diğer ağlara ekstra mesajlar göndermeyi gerektirmez. DRA'nın İSS düzeyinde BGP tabanlı en kısa yol hesaplamalarından yararlanması nedeniyle DRA'nın azaltma büyülü NSFCNET için biraz daha düşüktür. Sonuçlar incelendiğinde, SC², QC ve DRA çerçevelerinin MEP performansının, her yapılandırmada uçtan uca yollardaki anahtar sayısı arttıkça karşılaştırılabilir olduğu görülmektedir. Fakat SC² çerçevesi, Şekil 35'de gösterildiği gibi FST'de bir düşüş sergilemektedir.

4.5 Yazılım Tanımlı Optik Ağlar (YTOA)'da Blokzincir ile Geliştirilmiş Çoklu-İSS Spektrum Atama Çerçevesi: SpectrumChain

Bu bölümde, *SpectrumChain* (SC) çerçevesinin performansını Hiyerarşik Yönlendirme Yaklaşımı (HRA) ve mevcut operasyonel modu yansıtan BGP tabanlı en kısa yol seçiminin kullanan Dağıtılmış Yönlendirme Yaklaşımı (DRA) ile karşılaştırılması aktarılmıştır. Önceki bölümlerde açıklanan dalga-boyu dönüşümünün HWS, BWS ve NWS senaryoları altındaki performans analizi için Uçtan Uca Akış Kurulum Süresi (E2E FST), İletilen ve İşlenen Mesajlar (MEP), Hizmet Verilen İstekler (RS) ve Kabul Oranı (AR) metriklerini kullanılmıştır.

Erdos-Renyi rastgele ağ topolojisini 0.7 bağlantı derecesi ile dağıtırken, çeşitli dalga-boyu anahtarlama senaryolarında *SpectrumChain* (SC), HRA ve DRA'nın topolojiye göre hassasiyetini test etmek için NSFNET, US Backbone, Random-14 ve Random-24 ağ topolojileri aracılığıyla İSS düzeyinde düğüm sayılarını [5, 10] aralığında artırılmıştır. SC, HRA ve DRA ile HWS, BWS ve NWS senaryolarının çarprazlamalarını SC^H , SC^B , SC^N , HRA^H , HRA^B , HRA^N , DRA^H , DRA^B , ve DRA^N olarak sırasıyla tanımladık. Simülasyonlarda, en az iki optik taşıyıcı talebi olan İSS'ler arası hizmet taleplerini dikkate alınmıştır. Her bir fiber optik bağlantı, ağ kaynağı sınırlamaları nedeniyle hizmet reddini engellemek için rastgele kullanılabilir 55 optik taşıyıcı bulundurmaktadır.

Tablo 7: Performans değerlendirmesinde kullanılan kısaltmalar ve anlamları

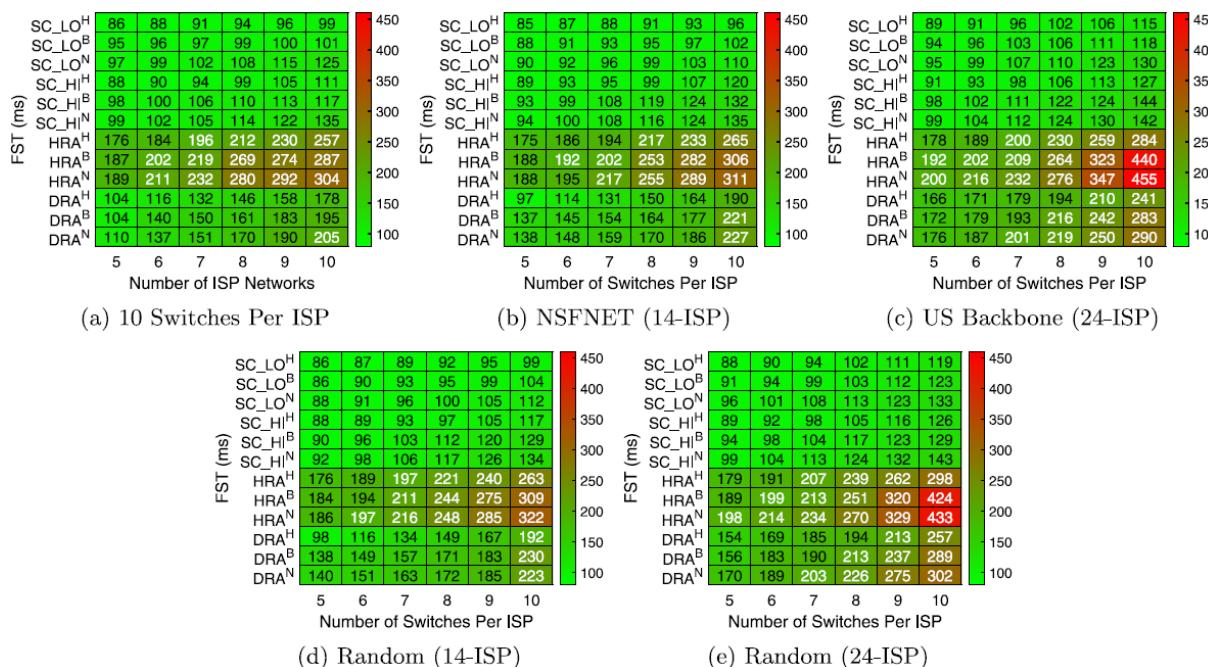
Kısaltma	Anlamı
HWS	Hop-by-Hop dalga-boyu değiştirme senaryosu
BWS	Sınır dalga-boyu değiştirme senaryosu
NWS	Dalga-boyu değiştirme senaryosu yok
$SC_LO^H = SC^H$	HWS'deki orijinal işlem sayısıyla birlikte
$SC_LO^B = SC^B$	SC, BWS'deki orijinal işlem sayısıyla birlikte
$SC_LO^N = SC^N$	SC, NWS'deki orijinal işlem sayısıyla birlikte
SC_HI^H	SC, HWS'de yüksek işlem hacmine sahip
SC_HI^B	SC, BWS'de yüksek işlem hacmine sahip
SC_HI^N	SC, NWS'de yüksek işlem hacmine sahip
HRA^H	HWS'de HRA (Hiyerarşik Yönlendirme Yaklaşımı)
HRA^B	BWS'de HRA (Hiyerarşik Yönlendirme Yaklaşımı)
HRA^N	NWS'de HRA (Hiyerarşik Yönlendirme Yaklaşımı)
DRA^H	HWS'de DRA (Dağıtık Yönlendirme Yaklaşımı)
DRA^B	BWS'de DRA (Dağıtık Yönlendirme Yaklaşımı)
DRA^N	NWS'de DRA (Dağıtık Yönlendirme Yaklaşımı)

Her bir performans metriğinin şekillerde, Random, NSFNET ve US Backbone topolojilerinin İSS-içi ağlarında aynı rastgele topolojileri kullanılmıştır. Tüm performans durumları için, farklı İSS'lerden kaynak ve hedef düğümleri olan 600 hizmet talebi

tanımlanmıştır. Ayrıca, blokzincir defter boyutunu orijinal işlem sayısından yüksek işlem hacmine (yani 1 milyon işlem) çıkararak boyut duyarlılığı analizi yapmak için defter büyütükçe blokzincir boyutunun etkisi de analiz edilmiştir. Tablo 7'de performans değerlendirmesinde kullanılan kısaltmalar ve anlamları listelenmektedir.

4.5.1 Uçtan Uca Akış Kurulum Süresi (E2E Flow Setup Time – FST)

Uçtan Uca Akış Kurulum Süresini (E2E FST), bir uçtan uca optik fiber yolu üzerinden hizmeti/akışı etkinleştiren ilgili akış kuralını anahtarlarla kurmak için gereken süre olarak tanımlanmıştır. Denetleyiciden anahtarlar veya denetleyiciden denetleyiciye mesaj/paket iletiminin gidiş-dönüş süresi, paket/mesaj işleme süresi ve anahtarlar veya denetleyiciler arasındaki denetleyici optik yol hesaplama süresi gibi ağ/topoloji tabanlı gecikmeler FST metriğini etkiler. Bu gecikmeler önemli olduğunda, anahtar akış tablolarında akış kuralları eklenirken, silinirken veya güncellenirken akış kurulum gecikmesine neden olurlar. Bu nedenlerden dolayı FST, SDON'da tüm ajan ölçeklenebilirlik performansını ölçmek ve analiz etmek için önemli bir metriktir.



Şekil 37: SC, HRA, ve DRA için Akış Kurulum Süresi

Şekil 37'de, üç dalga-boyu dönüşüm senaryosu (HWS, BWS ve NWS) altında düşük ve yüksek işlem hacimli SC, HRA ve DRA çerçevelerindeki toplam FST değerleri sunulmuştur. Şekil 37a'da gösterildiği gibi, SC çerçevesinin FST değeri, her bir İSS'nin İSS-içi seviyede sabit 10 adet sayıda anahtara sahip olduğu, artan İSS sayısı altında QoS tabanlı uçtan uca İSS'ler arası yolun belirlenmesinde HRA ve DRA'a göre daha düşüktür. SC çerçevesinin hem düşük

hem de yüksek işlem boyutlarındaki FST değeri, İSS ağlarının sayısı 7'den azken sırasıyla DRA'nın %60'ı ve HRA'nın %50'si kadar düşüktür. İSS ağlarının sayısı 7'den fazla olduğunda, SC ile diğer ikisi (DRA ve HRA) arasındaki farkın büyülüğu daha da belirgin hale gelir. Şekil 37b - Şekil 37e'de, yukarıda bahsedilen performans farkının aynısını NSFNET, US Backbone, Random-14 ve Random24 topolojileri üzerinde görmekteyiz. Şekil 37b ve Şekil 37d'de, SC çerçevesi, ağ başına artan anahtar sayısı dikkate alındığında, kaynak İSS'den hedef İSS ağlarına mevcut işlemleri daha az FST ile seçebilmektedir. Şekil 37c ve Şekil 37e'de, BWS ve NWS dalga-boyu senaryoları kullanılarak ağ başına anahtar sayısı 7'den yüksek olduğunda HRA ve DRA yaklaşımlarının FST değerleri iki katına kadar artmaktadır. Bunun nedeni, HRA ve DRA yaklaşımlarının kaynak İSS'den hedef İSS ağlarına giden uçtan uca yolu boyunca ardisık ve bitişik spektrumlara dayalı mevcut yolu aramasıdır. Öte yandan, önerilen SC çerçevesi, tüm dalga-boyu senaryoları için HRA ve DRA yaklaşımlarından daha az FST ile QoS tabanlı uçtan uca yolunu seçmek için blozincirdeki mevcut işlemleri kullanmaktadır. İlginç bir şekilde, NWS'li SC yaklaşımı (yani SC^N), mevcut yolu seçmek için HWS'li SC yaklaşımından (SC^H) ve Şekil 37c ve Şekil 37e'deki güncellenmiş 1M işleminden (SC_H/H^H) daha yüksek arama süresine ihtiyaç duyar. Bunun nedeni SC^N çerçevesinin NWS senaryosunda ardisık ve bitişik spektrumları seçmek için tüm işlemleri kontrol etmesi gerekip SC_H/H^H in HWS dalga-boyu senaryosu göz önünde bulundurularak gerekli QoS parametresine sahip ilk mevcut işlemi seçmesidir. Son olarak, DRA yaklaşımı FST değerleri açısından HRA'dan daha iyi performans göstermektedir çünkü BGP tabanlı en kısa yolu İSS'ler arası kullanırken, HRA uçtan uca fiber optik yol üzerinden broker aracılığıyla tüm İSS denetleyicileri ile iletişim kurmaktadır.

4.5.2 İletilen ve İşlenen Mesajlar (Messages Exchanged and Processed – MEP)

Ağ düğüm sayısı (örn. ana bilgisayarlar, ağ cihazları, vb.) açısından büyükçe, denetleyiciler, İSS ağları üzerinden bir fiber optik uçtan uca yolu oluşturmak için ağ cihazları arasında daha fazla akış talebiyle ilgili mesaj alışverişi ve işleme ile uğraşmak zorunda kalabilir. CPU ve bellek gibi sınırlı hesaplama kaynakları nedeniyle, bu ek mesaj alışverişi ve işleme süreçleri denetleyicilerde darboğazlara yol açabilir. Bu nedenle, İletilen ve İşlenen Mesajlar (MEP), denetleyicilerin ağıın genel ölçeklenebilirlik performansını değerlendirmesi ve incelemesi bakımından bir başka önemli metriktir. MEP metriği ilgili karar verici denetleyiciler (yani, uçtan uca yolu üzerindeki denetleyiciler ve Broker (varsayı)) tarafından bir Kaynak-Yönlendirme akışı için bir uçtan uca yolu oluşturmak üzere İletilen ve İşlenen ilgili mesajların miktarını ifade eder.

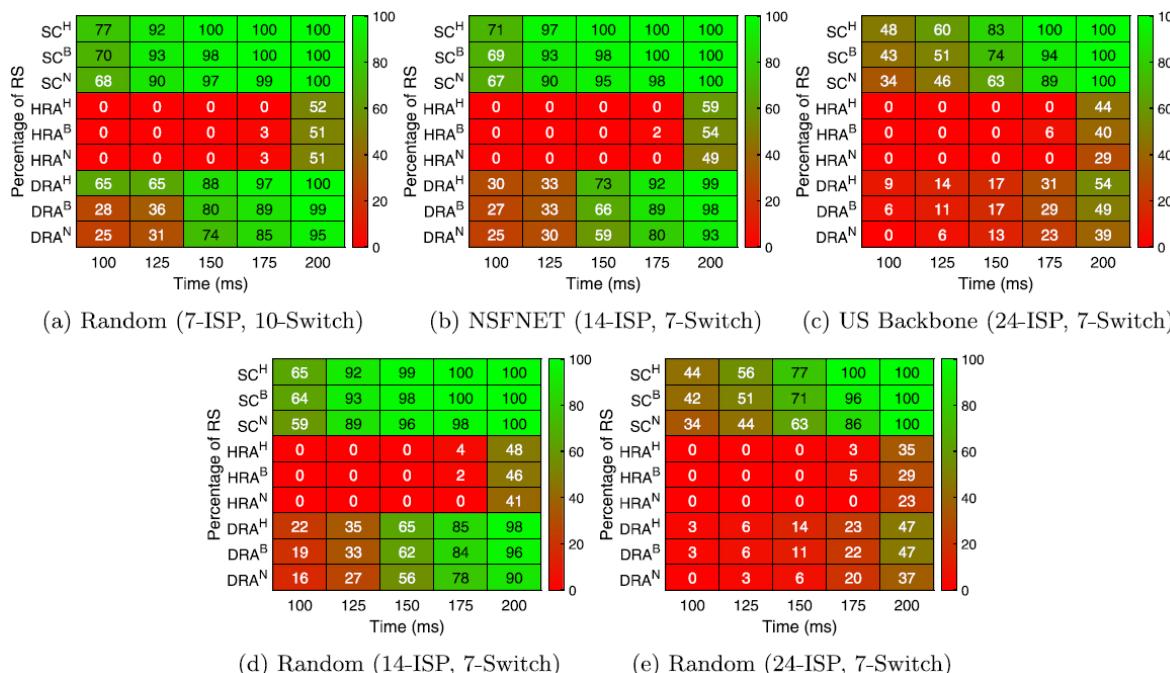
Tablo 8: HWS, BWS ve NWS senaryolarına göre SC, HRA ve DRA'da İletilen ve İşlenen Mesajların (MEP) sayısı

(a) 10 Switches per ISP						(b) NSFNET (14-ISP)					(c) US Backbone (24-ISP)									
Cases	Number of ISPs					Cases	Number of Switches per ISP					Cases	Number of Switches per ISP							
	5	6	7	8	9		5	6	7	8	9		5	6	7	8	9	10		
SC ^H	9	11	14	18	21	25	SC ^H	24	28	31	35	39	42	SC ^H	25	31	37	41	47	51
SC ^B	10	12	16	19	22	25	SC ^B	27	33	36	38	40	44	SC ^B	26	39	42	47	51	56
SC ^N	12	14	18	21	25	28	SC ^N	32	39	40	44	47	51	SC ^N	30	46	50	53	59	63
HRA ^H	19	20	22	25	29	34	HRA ^H	33	36	43	49	55	65	HRA ^H	38	42	48	53	63	74
HRA ^B	19	20	23	26	30	36	HRA ^B	38	44	53	63	68	72	HRA ^B	43	47	56	65	73	80
HRA ^N	21	23	27	30	34	42	HRA ^N	44	51	61	69	80	79	HRA ^N	49	55	66	78	81	88
DRA ^H	11	14	16	19	21	23	DRA ^H	19	22	24	27	30	34	DRA ^H	20	23	24	30	34	38
DRA ^B	13	16	17	19	22	24	DRA ^B	22	25	26	29	34	36	DRA ^B	23	24	30	34	36	40
DRA ^N	15	18	19	21	25	28	DRA ^N	26	28	29	34	40	42	DRA ^N	27	28	33	38	43	48
(d) Random (14-ISP)						(e) Random (24-ISP)														
Cases	Number of Switches per ISP					Cases	Number of Switches per ISP					Cases	Number of Switches per ISP							
	5	6	7	8	9		10	5	6	7	8		9	10	5	6	7	8	9	10
SC ^H	23	26	30	36	38	41	SC ^H	23	30	34	39	46	53	SC ^B	27	37	43	48	53	57
SC ^B	26	31	36	39	41	46	SC ^B	30	44	48	56	62	64	SC ^N	30	46	51	55	60	64
SC ^N	29	36	42	46	49	53	SC ^N	30	44	48	56	62	64	HRA ^H	34	39	45	56	63	75
HRA ^H	34	39	45	49	56	63	HRA ^H	39	44	49	54	66	75	HRA ^B	37	42	51	62	69	74
HRA ^B	37	42	51	62	69	74	HRA ^B	42	68	55	64	74	82	HRA ^N	41	47	59	70	81	88
HRA ^N	41	47	59	70	81	88	HRA ^N	50	77	63	73	84	92	DRA ^H	20	22	25	27	31	35
DRA ^H	20	22	25	27	31	35	DRA ^H	20	23	25	30	33	37	DRA ^B	23	25	27	30	35	38
DRA ^B	23	25	27	30	35	38	DRA ^B	22	26	31	36	39	42	DRA ^N	27	29	31	35	39	43
DRA ^N	27	29	31	35	39	43	DRA ^N	26	29	36	40	44	47							

Yukarıda bildirilen FST testlerine benzer bir ortamda, Tablo 8a'da sabit sayıda anahtar (İSS başına 10 anahtar) altında tüm dalga-boyu anahtarlama senaryoları ile SC, HRA ve DRA çerçeveleri için MEP değerleri sunulmaktadır. Tablo 8b - Tablo 8e'de NSFNET, US Backbone, Random-14 ve -24 ağ topolojileri için her bir İSS ağında artan sayıda anahtar simülle edilmektedir. Tablo 8a'da, SC çerçevesi, İSS ağılarının sayısı 8'e kadar çıktılarında QoS tabanlı bir uçtan uca optik fiber yol kurmak için daha düşük bir MEP değeri vermektedir. İlginç bir şekilde, İSS sayısı 8'i aşındırında, BGP'lerin normal çalışmasının bir parçası olarak İSS'ler arası düzeyde zaten hesaplanmış olan BGP tabanlı en kısa yollardan yararlandığı ve böylece düşük akış İSS'leriyle gereksiz iletişimleri azalttığı için DRA'nın MEP değeri daha düşük olmaktadır. Sonuç olarak, DRA, Tablo 8b - Tablo 8e'de İSS başına değişen anahtar sayısı altında SC çerçevesinden biraz daha iyi performans göstermektedir. Hiyerarşik yapısı ve bir broker aracılığıyla koordinasyon nedeniyle HRA, İSS'ler arası ağılar üzerinde uçtan uca optik fiber yollarını hesaplamak için doğal olarak daha yüksek ek yük ile çalışır. Dolayısıyla, SC ve DRA yaklaşımları tüm dalga-boyu anahtarlama senaryoları için MEP değeri açısından HRA'dan daha iyi performans göstermektedir. Farklı dalga-boyu anahtarlama senaryolarının, yani HWS, BWS ve NWS'nin etkisini göz önünde bulundurduğumuzda, diğer her şey eşit olduğunda, HWS'nin MEP'de BWS'den daha iyi sonuçlar ürettiğini ve NWS performansının en kötüsü olduğunu gözlemliyoruz. Daha önce FST metriği ile ilgili sonuçlara benzer şekilde; HWS'nin yol seçiminde BWS'den çok daha fazla seçenek sunduğu ve NWS'nin en az potansiyel yol adaylarıyla kısıtlandığı açıktır.

4.5.3 Hizmet Verilen Talepler (Requests Serviced – RS)

Çalışmada dikkate aldığımız bir diğer ölçüt de Hizmet Verilen İstek Sayısı (RS)'dır. Her bir hizmet talebi için ağ kontrol örnekleri tahsis ederek İSS'ler arası ağlar üzerinden uçtan uca optik fiber yollarını hesaplamak için FST metriği ile bir şekilde ilişkilidir. FST daha yüksek olduğunda, tüm ağ boyunca bir hizmet talebinin oluşturulması daha uzun sürer. Yine de çeşitli test senaryoları altında daha önceki FST gözlemlerini daha fazla doğrulamak ve desteklemek için etkiyi doğrudan RS metriği perspektifinden görebilmek istedik. Daha önceki FST simülasyon ayarlarına benzer şekilde Şekil 38, [100 ms, 200 ms] aralığında zaman değişirken 600 hizmet talebi için RS yüzdesini göstermektedir. Şekil 38a'da, temel FST 175 ms'ye kadar HRA ve DRA yaklaşımından daha düşük olduğu için SC hizmetleri karşılamak adına daha iyi performansa sahiptir. Zaman 175 ms'den yüksek olduğunda, SC ve DRA yaklaşımı Şekil 38a'daki maksimum sayıda hizmeti karşılamaktadır. Şekil 38b ve Şekil 38d, SC'nin daha düşük FST değerlerinin doğrudan bir sonucu olarak, SC çerçevesinin daha fazla talebe hizmet verebildiğini ve dolayısıyla daha yüksek bir RS değerine sahip olduğunu göstermektedir. HRA yaklaşımının 175 ms'den daha kısa bir uçtan uca optik fiber yolu bulamadığını da unutmamak gereklidir. SC çerçevesi talep edilen hizmetlerin çoğunu yönetir ve süre 175 ms'ye kadar artırıldığında diğer yaklaşılardan iki kat daha iyi performans gösterir.



Şekil 38: SC, HRA ve DRA için Hizmet Verilen Talepler (RS)

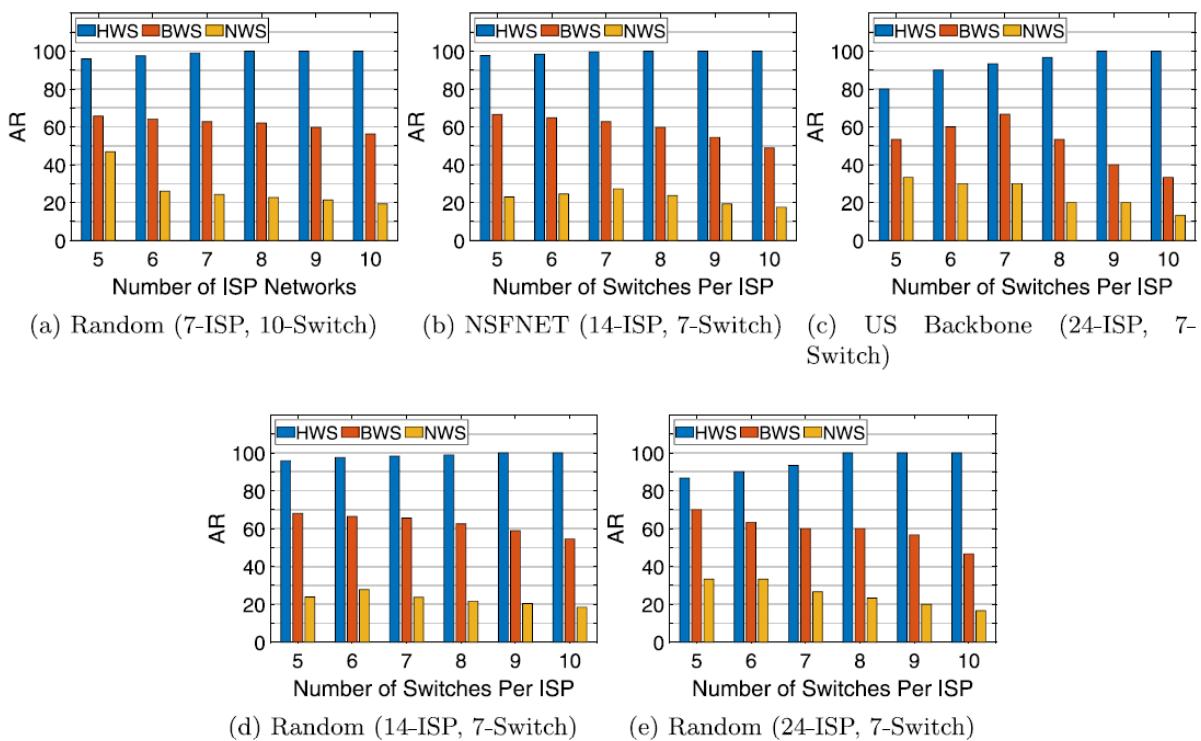
Son olarak, DRA yaklaşımı, Şekil 38c ve Şekil 38e'de daha geniş bir İSS yelpazesinde ve her bir İSS içindeki anahtarlarında daha düşük hacimde talep edilen hizmetleri

karşılayabilmektedir. Ancak, HRA yaklaşımı tüm İSS'ler arası iletişim için bir broker kullandığından, süre 175 milisaniyeye ulaştığında hizmet taleplerini karşılamaya başlayabilir. Sonuç olarak, önerilen SC çerçevesinin İSS ağları üzerinden bir uçtan uca optik fiber yolunu hesaplamak için İSS denetleyicileri tarafından zaten onaylanmış olan blokzincir işlemlerini kullandığı açıklıktır. Dalga-boyu anahtarlama senaryoları için NWS senaryosu, fiber optik yollarını uçtan uca uçtan uca fiber optik yolları boyunca hizalanmaya zorladığından hem HWS hem de BWS'ye kıyasla talep edilen hizmetlerin kurulmasını azaltmaktadır.

4.5.4 Hizmetlerin Kabul Oranı (Acceptance Ratio of services – AR)

Her bir dalga-boyu anahtarlama stratejisinin tüm yol seçim yaklaşımları üzerindeki genel etkisini değerlendirmek ve analiz etmek için, Şekil 39'de SC, HRA ve DRA yaklaşımılar için talep edilen hizmetlerin kabul oranı verilmektedir. Şekil 39a'da İSS-içi düzeyde sabit anahtar sayısı (10 adet) ile İSS ağlarının sayısı artırılırken ve Şekil 39b - Şekil 39e'de İSS'ler arası düzeyde NSFNET, US Backbone, Random-14 ve Random-24 düğüm ağı topolojilerinde İSS başına anahtar sayısı artırılırken ortalamaları aracılığıyla sunulmaktadır.

Hop-by-hop Dalga-Boyu Anahtarlama (HWS) stratejisi, Şekil 39a - Şekil 39e'de gösterildiği gibi farklı İSS ağları boyunca kaynaktan hedef düğüme QoS tabanlı uçtan uca optik fiber yollarının talebini karşılamak için Sınır-Node-Sadece Dalga-Boyu Anahtarlama (BWS) ve Dalga Boyu Anahtarlama Yok (NWS) senaryolarından daha iyi performans göstermektedir. Bunun nedeni, HWS yaklaşımının tüm ağ boyunca uçtan uca optik fiber yolunu oluştururken optik fiber bağlantılarını seçmek için mevcut ardışık spektrumları kullanabilmesidir. Öte yandan, BWS, farklı İSS'ler arasında bitişik spektrumlar oluşturmak için uygunluk yoksa, her bir İSS ağının sınır veya kenar düğümlerindeki dalga boyalarını değiştirme yeteneğine sahiptir. Bu nedenle BWS, birden fazla İSS ağının üzerinde kaynaktan hedef düğümlere giden bir fiber optik yol boyunca herhangi bir dalga-boyunu değiştiremeyen NWS yaklaşımından da daha iyi performans gösterir.



Şekil 39: HWS, BWS ve NWS senaryolarına göre Hizmetlerin Kabul Oranı

İlginc bir şekilde, Şekil 39a'daki İSS ağlarının sayısı ve Şekil 39b - Şekil 39e'deki İSS ağı başına anahtar sayısı arttığında, HWS senaryosunun kabul oranı artarken, BWS ve NWS'ninki düşmektedir. Bunun nedeni, HWS'nin ağ kaynağı arttığında bir uçtan uca optik fiber yolu oluşturmak için mevcut ardışık spektrumları seçerek kullanmasıdır. Buna karşılık, BWS ve NWS senaryoları, bir fiber optik yol bulmak için İSS-içi ve İSS'ler arası ağlar aracılığıyla ardışık ve bitişik spektrumları hizalama zorluğuyla karşı karşıyadır.

5. TARTIŞMA

Bu bölümde, projede gerçekleştirilen çalışmalar sırasında karşılaşılan çeşitli sınırlamalar, zorluklar ve gelecekteki potansiyel çalışma fikirleri hakkında daha fazla araştırma başlatmak amacıyla bir tartışma sunulmaktadır.

5.1 Blokzincir Defterinin Güncellenme Süresi

Projede gerçekleştirilen çalışmalarında, İSS'ler içerisinde bulunan parça yol bilgilerini tutan blokzincir işlemleri, ağ dinamikleri ve trafik koşulları nedeniyle sürekli değişebilirler. Daha önce açıklandığı gibi, bir parça yolda yapılan bir güncellemeden sonra yeni bir blokzincir işlemi oluşturulur. Blokzincir işlemi oluşturulduktan sonra, blokzincir defterine eklenene kadar bir dizi

adımdan geçer (blok oluşturucu düğümlere gönderme, yeni bloğa dahil edilmeyi bekleme, yeni bloğun blozkincir ağına yayılması, vb.). Blozkincir işlemindeki parça yolun, blozkincir defterine yansıyan durum değişikliği deftere yansıyana kadar devam etmeyebilir çünkü bu sırada ağıda ilgili parça yolun durumuna ilişkin bir başka bir güncelleme daha gerçekleşebilir. Bu nedenle, bazı blozkincir işlemleri blozkincirde ki güncel parça yol durumlarını yansıtmayabilir. Bu durum geleneksel ağlardaki kullanılan yönlendirme güncellemelerine benzemektedir.

5.2 Blozkincir Defter Boyutu

Blozkincir İşlemleri, yapılan çalışmalarda ağlardaki parça yolları yansıtmaktadır. Bu işlemler, topoloji ve QoS ile ilgili verileri saklamaktadırlar. Ağlar boyut, topoloji, hizmetler, kullanıcılar ve benzeri konularda karmaşık yapılara sahip olduklarıdan, ağ işlemleri sırasında sık sık değişiklikler güncellemeler olabilir. Bu güncellemeler sisteme yeni blozkincir işlemleri ile yansıtılır ve ağıdaki tüm katılımcılara yayılır. Bu nedenle, depolama uzun vadede bir kısıtlama haline gelebilir. Ayrıca bu durum, denetleyicilerin uğraştığı ağ ek yükünü de etkileyebilir.

5.3 Blok Süresi

Blok süresi aynı zamanda blok aralığı olarak da bilinir. Bu süre oldukça önemli bir metriktir çünkü tüm modelin daha yüksek blozkincir işlem hacmi elde etmesine yardımcı olabilir. Ayrıca, İSS'lerden gelen parça yol durumlarıyla ilgili olarak blozkincir defterinin güncel tutulmasına yardımcı olur. Öte yandan, blok aralığı ağ denetleyicilerinde daha fazla hesaplama yapılmasına neden olabilir. Bu durum ise ağlarda daha fazla bağlantı kaynağı kullanmasına yol açabilir. Dolayısıyla blok oluşturma süresi, buna bağlı hesaplama yükü ve blozkincir defterinin eskimesi arasında hassas bir denge vardır.

5.4 Uzlaşma Protokolü Yükü

Uzlaşma protokollerı, YTA ortamında ağ denetleyicileri üzerindeki ek yük açısından geliştirilen çerçevelerin genel performanslarında önemli bir noktada yer almaktadır. Çeşitli dengeleri daha iyi anlamak için farklı; hafif, etkili ve verimli uzlaşma protokollerini araştırmak, gelecekte yapmayı planladığımız çalışmalarımız arasında yer almaktadır. Öngörülen modelde blozkincir düğümleri olarak İSS denetleyicileri mevcut durumda ağ ile ilgili görevler ve mesajlarla meşguldür. Yeni bir blok oluşturmak için bu denetleyicilere kriptografik problemleri çözmek gibi yoğun hesaplama gerektiren görevler yüklemek, daha fazla hesaplama yükü gerektirdiğinden ağıla ilgili performanslarını sınırlayabilir. Bu nedenle, daha hafif ancak verimli bir uzlaşma protokolünün kullanılması denetleyicilerin yükünü hafifletip performansı artırabilir.

Sonuç olarak, yukarıda bahsedilen sınırlamaların ve zorlukların yapısal boyutu, ağ ve denetleyici ek yüküne ilişkin daha derinlemesine çalışmalara da fayda sağlayacaktır.

6. SONUÇ

Bu projede, Yazılım Tanımlı Ağ (YTA), Blokzincir ve Makine Öğrenmesi teknolojilerinin entegrasyonuyla İnternet Servis Sağlayıcıları (İSS) arasında daha akıllı ve verimli bütünsel bir yönlendirme modeli geliştirilmiştir. Elde edilen sonuçlar, kullanıcı taleplerinin hizmet kalitesine odaklanan bağlantı isteklerini daha hızlı ve etkin bir şekilde işleyerek ağların ölçeklenebilirliğini artırmak için önemli bir adım olmuştur. Özellikle, *QoSChain* gibi blokzincir tabanlı yönlendirme çerçeveleri ve *SmartContractChain* gibi akıllı sözleşme destekli koordinasyon mekanizmaları, ağ sağlayıcılarının iş birliğini kolaylaştırarak hizmet kalitesini iyileştirmekte ve ağ performansını artırmaktadır. Ayrıca, projede gerçekleştirilen çalışmalar çeşitli yol seçim stratejilerinin performansını inceleyerek, en uygun yönlendirme stratejilerini belirlemeye yönelik kapsamlı bir analiz sunmuştur. Bu durumda, internet trafiğinin farklı özelliklerine uygun olarak ağların daha etkili yönetilmesine olanak tanır. Son olarak, çalışmanın bir diğer önemli çıktısı da Pekiştirmeli Öğrenme (RL) algoritmalarının kullanılmasıyla gerçekleştirilen QoS motivasyonlu ağlar arası trafik yönlendirilmesidir. Bu yönlendirme modeli, ağların dinamik koşullarına uyum sağlayarak, hizmet kalitesini sürekli olarak iyileştirebilir. Bu sonuçlar, ISS'lerin ağ yönetimini daha akıllı ve etkin bir şekilde yapmalarına yardımcı olabilir. Böylece kullanıcı deneyimini artırırken ağ performansını da iyileştirir. Ayrıca, bu teknolojik gelişmelerin ağlar arası iş birliği ve hizmet kalitesi standardizasyonu gibi alanlarda daha geniş çaplı etkileri olabilir.

KAYNAKLAR

- Abujoda, A., Kouchaksaraei, H. R., & Papadimitriou, P. (2016). SDN-based source routing for scalable service chaining in datacenters. *International Conference on Wired/Wireless Internet Communication*, 66-77.
- Ak, E., & Canberk, B. (2019). BCDN: A proof of concept model for blockchain-aided CDN orchestration and routing. *Computer Networks*, 161, 162-171. <https://doi.org/10.1016/j.comnet.2019.06.018>
- Alemany, P., Vilalta, R., Muñoz, R., Casellas, R., & Martinez, R. (2022). Evaluation of the abstraction of optical topology models in blockchain-based data center interconnection. *Journal of Optical Communications and Networking*, 14(4), 211-221. <https://doi.org/10.1364/JOCN.447833>
- Alemany, P., Vilalta, R., Muñoz, R., Martínez, R., & Casellas, R. (2020). Managing Network Slicing Resources Using Blockchain in a Multi-Domain Software Defined Optical Network Scenario. *2020 European Conference on Optical Communications (ECOC)*, 1-4. <https://doi.org/10.1109/ECOC48923.2020.9333352>
- Arins, A. (2018). Blockchain based Inter-domain Latency Aware Routing Proposal in Software Defined Network. *AIIEEE*, 1-2.
- Bhaumik, P., Zhang, S., Chowdhury, P., Lee, S.-S., Lee, J. H., & Mukherjee, B. (2014). Software-defined optical networks (SDONs): a survey. *Photonic Network Communications*, 28(1), 4-18.
- Caesar, M., & Rexford, J. (2005). BPG routing policies in ISP networks. *IEEE Network*, 19(6), 5-11. <https://doi.org/10.1109/MNET.2005.1541715>
- Chatterjee, B. C., Sato, T., & Oki, E. (2018). Recent research progress on spectrum management approaches in software-defined elastic optical networks. *Optical Switching and Networking*, 30, 93-104.
- Chen, F., Li, Z., Li, B., Deng, C., Tian, Z., Lin, N., Wan, Y., & Bao, B. (2020). Blockchain-based Optical Network Slice Rental Approach for IoT. *2020 IEEE Computing, Communications and IoT Applications (ComComAp)*, 1-4. <https://doi.org/10.1109/ComComAp51192.2020.9398886>
- Clarence Filsfils, Stefano Previdi, Ahmed Bashandy, Bruno Decraene, Stephane Litkowski, Martin Horneffer, Igor Milojevic, Rob Shakir, Saku Ytti, Wim Henderickx, Jeff Tantsura, & Edward Crabbe. (2013). Segment routing with IS-IS routing protocol. *Draft-previdi-filsfils-isis-segment-routing-00 (Work in Progress)*.
- De Angelis, S. (2018). Assessing security and performances of consensus algorithms for permissioned blockchains. *arXiv preprint arXiv:1805.03490*.
- Ding, S., Shen, G., Pan, K. X., Bose, S. K., Zhang, Q., & Mukherjee, B. (2020). Blockchain-Assisted Spectrum Trading Between Elastic Virtual Optical Networks. *IEEE Network*, 34(6), 205-211. <https://doi.org/10.1109/MNET.011.2000138>
- Erdős P. and Rényi, A. (1964). On the strength of connectedness of a random graph. *Acta Mathematica Academiae Scientiarum Hungarica*, 12(1), 261-267.
- Fichera, S., Sambo, N., Paolucci, F., Cugini, F., & Castoldi, P. (2019). Leveraging blockchain to ratify QoT performance in multi-domain optical networks. *45th European Conference on Optical Communication (ECOC 2019)*, 1-4. <https://doi.org/10.1049/cp.2019.1047>
- Fichera, S., Sgambelluri, A., Paolucci, F., Giorgetti, A., Sambo, N., Castoldi, P., & Cugini, F. (2021). Blockchain-Anchored Disaggregated Optical Networks. *Journal of Lightwave Technology*, 39(20), 6357-6365. <https://doi.org/10.1109/JLT.2021.3098851>
- Gerstel, O., Jinno, M., Lord, A., & Yoo, S. J. Ben. (2012). Elastic optical networking: a new dawn for the optical layer? *IEEE Communications Magazine*, 50(2), s12-s20. <https://doi.org/10.1109/MCOM.2012.6146481>
- Ghiasian, A. (2020). Impact of TCAM size on power efficiency in a network of OpenFlow switches. *IET Networks*, 9(6), 367-371. <https://doi.org/10.1049/iet-net.2019.0125>
- Guimarães, R. S., Dominicini, C., Martínez, V. M. G., Xavier, B. M., Mafioletti, D. R., Locateli, A. C., Villaca, R., Martinello, M., & Ribeiro, M. R. N. (2022). M-PolKA: Multipath

- Polynomial Key-Based Source Routing for Reliable Communications. *IEEE Transactions on Network and Service Management*, 19(3), 2639-2651. <https://doi.org/10.1109/TNSM.2022.3160875>
- Guo, C., Lu, G., Wang, H. J., Yang, S., Kong, C., Sun, P., Wu, W., & Zhang, Y. (2010). Secondnet: a data center network virtualization architecture with bandwidth guarantees. *Proceedings of the 6th International Conference*, 1-12.
- Holst, A. (2021). Total data volume worldwide 2010-2025. İçinde Statista. <https://www.statista.com/statistics/871513/worldwide-data-created/>
- Johnson, D., Hu, Y., & Maltz, D. (2007). *The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4*. <https://doi.org/10.17487/rfc4728>
- Johnson, D., Hu, Y., Maltz, D., & others. (2007). *The dynamic source routing protocol (DSR) for mobile ad hoc networks for IPv4*.
- Judtmayer, A., Stifter, N., Schindler, P., & Weippl, E. (2018). Blockchain: Basics. İçinde *Business Transformation through Blockchain: Volume II* (ss. 339-355). https://doi.org/10.1007/978-3-319-99058-3_13
- Jyothi, S. A., Dong, M., & Godfrey, P. B. (2015). Towards a Flexible Data Center Fabric with Source Routing. *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research*. <https://doi.org/10.1145/2774993.2775005>
- Kamboj, P., & Pal, S. (2019). QoS in Software Defined IoT Network Using Blockchain Based Smart Contract: Poster Abstract. *Sensys'19*, 430–431. <https://doi.org/10.1145/3356250.3361954>
- Karakus, M., & Durresi, A. (2015). A Scalable Inter-AS QoS Routing Architecture in Software Defined Network (SDN). *Proceedings - International Conference on Advanced Information Networking and Applications*, AINA, 2015-April, 148-154. <https://doi.org/10.1109/AINA.2015.179>
- Karakus, M., & Durresi, A. (2017a). A survey: Control plane scalability issues and approaches in Software-Defined Networking (SDN). *Computer Networks*, 112. <https://doi.org/10.1016/j.comnet.2016.11.017>
- Karakus, M., & Durresi, A. (2017b). Quality of Service (QoS) in Software Defined Networking (SDN): A survey. İçinde *Journal of Network and Computer Applications* (C. 80, ss. 200-218). <https://doi.org/10.1016/j.jnca.2016.12.019>
- Karakus, M., & Durresi, A. (2017c). Quality of Service (QoS) in Software Defined Networking (SDN): A Survey. *Journal of Network and Computer Applications* , 80, 200-218. <https://doi.org/http://dx.doi.org/10.1016/j.jnca.2016.12.019>
- Karakus, M., Guler, E., & Uludag, S. (2021). QoSChain: Provisioning Inter-AS QoS in Software-Defined Networks With Blockchain. *IEEE Tran on Netw and Service Management*, 18(2), 1706-1717. <https://doi.org/10.1109/TNSM.2021.3060476>
- Karakus, M., Guler, E., & Uludag, S. (2023). Smartcontractchain (SC\$\$^2\$\$): Cross-ISP QoS traffic management framework with SDN and blockchain. *Peer-to-Peer Networking and Applications*, 16(6), 3003-3020. <https://doi.org/10.1007/s12083-023-01561-2>
- Komajwar, S., & Korkmaz, T. (2021). SPRM: Source Path Routing Model and Link Failure Handling in Software-Defined Networks. *IEEE Transactions on Network and Service Management*, 18(3), 2873-2887. <https://doi.org/10.1109/TNSM.2021.3066156>
- Kou, S., Yang, H., Zheng, H., Bai, W., Zhang, J., & Wu, Y. (2017). Blockchain Mechanism Based on Enhancing Consensus for Trusted Optical Networks. *2017 Asia Communications and Photonics Conference (ACP)*, 1-3.
- Kreutz, D., Ramos, F. M. V, Esteves Verissimo, P., Esteve Rothenberg, C., Azodolmolky, S., & Uhlig, S. (2015). Software-Defined Networking: A Comprehensive Survey. *Proceedings of the IEEE*, 103(1), 14-76. <https://doi.org/10.1109/JPROC.2014.2371999>
- Kumar, G. N., Katsalis, K., Papadimitriou, P., Pop, P., & Carle, G. (2021). Failure Handling for Time-Sensitive Networks using SDN and Source Routing. *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*, 226-234. <https://doi.org/10.1109/NetSoft51509.2021.9492666>

- Labovitz, C., Iekel-Johnson, S., McPherson, D., Oberheide, J., & Jahanian, F. (2010). Internet Inter-domain Traffic. *Proceedings of the ACM SIGCOMM 2010 Conference*, 75-86. <https://doi.org/10.1145/1851182.1851194>
- Lin, P., Bi, J., Wolff, S., Wang, Y., Xu, A., Chen, Z., Hu, H., & Lin, Y. (2015). A west-east bridge based SDN inter-domain testbed. *Communications Magazine, IEEE*, 53(2), 190-197. <https://doi.org/10.1109/MCOM.2015.7045408>
- McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., & Turner, J. (2008). OpenFlow: Enabling Innovation in Campus Networks. *SIGCOMM Comput. Commun. Rev.*, 38(2), 69-74. <https://doi.org/10.1145/1355734.1355746>
- Oktian, Y. E., Witanto, E. N., Kumi, S., & Lee, S. (2019). ISP Network Bandwidth Management: Using Blockchain and SDN. *ICTC*, 1330-1335.
- Papadopoulos, K., & Papadimitriou, P. (2019). Leveraging on Source Routing for Scalability and Robustness in Datacenters. *2019 IEEE 2nd 5G World Forum (5GWF)*, 148-153. <https://doi.org/10.1109/5GWF.2019.8911691>
- Podili, P., & Kataoka, K. (2021). TRAQR: Trust aware End-to-End QoS routing in multi-domain SDN using Blockchain. *Journal of Network and Computer Applications*, 182, 103055. <https://doi.org/10.1016/j.jnca.2021.103055>
- Ramezan, G., & Leung, C. (2018). A Blockchain-Based Contractual Routing Protocol for the Internet of Things Using Smart Contracts. *Wireless Communications and Mobile Computing*, 2018. <https://doi.org/10.1155/2018/4029591>
- Ramos, R. M., Martinello, M., & Esteve Rothenberg, C. (2013). SlickFlow: Resilient source routing in Data Center Networks unlocked by OpenFlow. *38th Annual IEEE Conference on Local Computer Networks*, 606-613. <https://doi.org/10.1109/LCN.2013.6761297>
- Saad, M., Anwar, A., Ahmad, A., Alasmari, H., Yuksel, M., & Mohaisen, A. (2019). Routechain: Towards blockchain-based secure and efficient bgp routing. *ICBC 2019 - IEEE International Conference on Blockchain and Cryptocurrency*, 210-218. <https://doi.org/10.1109/BLOC.2019.8751229>
- Soliman, M., Nandy, B., Lambadaris, I., & Ashwood-Smith, P. (2014). Exploring source routed forwarding in SDN-based WANs. *2014 IEEE International Conference on Communications (ICC)*, 3070-3075.
- Thyagaturu, A. S., Mercian, A., McGarry, M. P., Reisslein, M., & Kellerer, W. (2016). Software Defined Optical Networks (SDONs): A Comprehensive Survey. *IEEE Com Surv Tutor*, 18, 2738-2786. <https://doi.org/10.1109/COMST.2016.2586999>
- Ventre, P. L., Salsano, S., Polverini, M., Cianfrani, A., Abdelsalam, A., Filsfils, C., Camarillo, P., & Clad, F. (2021). Segment Routing: A Comprehensive Survey of Research Activities, Standardization Efforts, and Implementation Results. *IEEE Communications Surveys & Tutorials*, 23(1), 182-221. <https://doi.org/10.1109/COMST.2020.3036826>
- Wang, P., Liu, X., Chen, J., Zhan, Y., & Jin, Z. (2018). QoS-Aware Service Composition Using Blockchain-Based Smart Contracts. *Proceedings of the 40th International Conference on Software Engineering (ICSE): Companion Proceedings*, 296-297. <https://doi.org/10.1145/3183440.3194978>
- Xia, J., Cui, P., Li, Z., & Lan, J. (2021). SRCV: A Source Routing based Consistency Verification Mechanism in SDN. *2021 3rd International Conference on Advances in Computer Technology, Information Science and Communication (CTISC)*, 77-81. <https://doi.org/10.1109/CTISC52352.2021.00022>
- Yaga, D., Mell, P., Roby, N., & Scarfone, K. (t.y.). NISTIR 8202 Blockchain Technology Overview. İçinde *National Institute of Standards and Technology*.
- Yang, H. (2022). Optical and Wireless Convergence Network Based on Blockchain. İçinde Y. and H. S. and V. M. Bhatt Chintan and Wu (Ed.), *Security Issues in Fog Computing from 5G to 6G: Architectures, Applications and Solutions* (ss. 131-143). Springer International Publishing.
- Yang, H., Liang, Y., Yao, Q., Guo, S., Yu, A., & Zhang, J. (2019). Blockchain-based secure distributed control for software defined optical networking. *China Communications*, 16(6), 42-54. <https://doi.org/10.23919/JCC.2019.06.004>

- Zeng, Z., Zhang, X., & Xia, Z. (2022). Intelligent Blockchain-Based Secure Routing for Multidomain SDN-Enabled IoT Networks. *Wireless Communications and Mobile Computing*, 2022, 1-10. <https://doi.org/10.1155/2022/5693962>
- Zhang, Y., Beheshti, N., Beliveau, L., Lefebvre, G., Manghirmalani, R., Mishra, R., Patneyt, R., Shirazipour, M., Subrahmaniam, R., Truchan, C., & Tatipamula, M. (2013). StEERING: A software-defined networking for inline service chaining. *2013 21st IEEE International Conference on Network Protocols (ICNP)*, 1-10. <https://doi.org/10.1109/ICNP.2013.6733615>