Avrupa Bilim ve Teknoloji Dergisi
Özel Sayı, S. 325-331, Nisan 2020

Araştırma Makalesi

www.ejosat.com ISSN:2148-2683

European Journal of Science and Technology
Special Issue, pp. 325-331, April 2020

Research Article

IEEE 802.1 Time-Sensitive Networking Standardının Yazılım Tanımlı Ağ Yaklaşımıyla Değerlendirilmesi*

Mustafa Burak Şenyiğit[1], Deniz Turgay Altılar[2]

[1] ASELSAN A.Ş., Bilgi Yönetim Direktörlüğü, Ankara, Türkiye
[2] İstanbul Teknik Üniversitesi, Bilgisayar ve Bilişim Fakültesi, İstanbul, Türkiye

(Konferans Tarihi: 5-7 Mart 2020)

(**DOI:** 10.31590/ejosat.araconf42)

ATIF/REFERENCE: Şenyiğit, M. B. & Altılar, D. T. (2020). IEEE 802.1 Time-Sensitive Networking Standardının Yazılım Tanımlı Ağ Yaklaşımıyla Değerlendirilmesi. *Avrupa Bilim ve Teknoloji Dergisi*, (Özel Sayı), 325-331.

Öz

Endüstriyel otomasyon sistemleri, havacılık, ulaşım, sağlık ve askeri alanda kullanılan zaman kritik sistemler sıfır paket kaybı, sınırlı gecikme ve gecikmede sapma gibi gerçek zaman gereksinimlerine sahiptir. Zaman içinde bu tarz sistemlere özgü yüksek maliyetli donanımlar üzerinde koşan protokoller geliştirilmiştir. Malzeme teknolojisindeki ilerlemeler ile birlikte Ethernet ucuz donanımlar üzerinde çalışan ve her yerde bulunabilen bir hale geldi fakat gerçek zamanlı haberleşme gereksinimlerini karşılayabilecek mekanizmalara sahip değildi. IEEE 802.1 Time-Sensitive Networking (TSN), birkaç protokolden oluşan Ethernet bazlı gerçek zamanlı haberleşme sağlayan bir standarttır. Yazılım tanımlı ağlar (YTA) ise yönetilebilirlik, dinamizm ve maliyet açısından etkinlik sağlayan yeni bir mimaridir. Bu bildiride, TSN standardının yönlendirme mekanizmasına katkı sağlanmış ve TSN protokolleri YTA yaklaşımıyla gerçekleştirilip sonuçları incelenmiştir.

Anahtar Kelimeler: Yazılım Tanımlı Ağlar, Zaman Duyarlı Ağlar, Gerçek Zamanlı Haberleşme Sistemleri, Deterministtik Ağlar.

Evaluating Software-Defined Networking Approach Over IEEE 802.1 Time-Sensitive Networking Standard

Abstract

Time-critical systems like the ones used in industrial automation systems, aviation, transportation, in-vehicle, healthcare and military systems demand zero packet loss and real-time guarantees such as bounded latency and jitter from the underlying communication network. For those systems, different communication protocols have been enhanced and evolved. However, each protocol requires specific hardware ending up with increasing costs. Ethernet is cheap and ubiquitous, but it was designed to provide best effort communication and lacks mechanisms to meet real-time constraints. IEEE 802.1 Time-Sensitive Networking (TSN) is a set of protocols providing real-time Ethernet-based communication. Software-Defined Networking (SDN) is an emerging architecture which provides manageability, dynamism and cost-effectiveness. This paper makes contribution to forwarding mechanism of TSN and shows the results of applying TSN with SDN approach.

Keywords: Software-Defined Networking, Time-Sensitive Networks, Real-Time Communication Systems, Deterministic Networks.

* Bu makale ***International Conference on Access to Recent Advances in Engineering and Digitalization (ARACONF 2020)*** de sunulmuştur.

1. Giriş

Zaman kritik sistemler, dağıtık sensör ve aktuatorleri kontrol ettikleri için, sınırlı gecikme ve seğirme, sıfır paket kaybı, yüksek güvenilirlik gibi katı gerçek zaman gereksinimlerine sahiptir. Ağ üzerindeki düğümler, veri ve komut iletimi için ağ altyapısına güvendiklerinden, tahmin edilemeyen gecikme veya zaman seğirmesi sistem performansını ciddi şekilde indirger. Zamanlamaya ek olarak, güvenlik, güvenilirlik ve hata toleransı, kritik paketlerin iletimi için önemli kısıtlamalardır. Bu tarz sistemlerin gereksinimlerini karşılayabilmek için deterministtik bir ağ iskeletinin uygulanması gerekir. Materyal teknolojisindeki gelişmeler sayesinde, Ethernet ucuz ve yüksek performans veren bir ürün haline gelmesine rağmen gerçek zaman gereksinimlerini karşılayabilecek mekanizmalara sahip değildir. Genel eğilim, zaman duyarlı algoritmaları uygulayarak Ethernet ile gerçek zamanlı haberleşme sağlamaktır. Buna ek olarak, gerçek zamanlı ve gerçek zamanlı olmayan uygulamaların aynı ağ altyapısı üzerinde çalışması sağlanmalıdır.

IEEE 802.1 Time-Sensitive Networking (TSN) çalışma grubu, ağ katmanında, zamanda senkronize edilmiş Ethernet bazlı düşük gecikmeli haberleşme sağlamaya odaklanmış bir organizasyondur (Time-Sensitive Networking Task Group, t.y.). Grubun öncelikli hedefi deterministtik olmayan gecikmeleri yok edip paket kaybını önleyerek zaman duyarlı sistemlere deterministtik bir davranış kazandırmaktır. Bunu gerçekleştirmek için, IEEE 802.1 TSN zamanda senkronizasyon sağlayan protokolleri kullanarak zaman tetiklemeli haberleşme ile paketlerin iletimini planlama, kuyruk yönetimi, bağlantı veya cihaz arızalarına karşı paket replikasyonu, yönlendirme yol rezervasyonu gibi komplike metotlar önermiştir (Bello ve Steiner, 2019).

Bu metotları uygulamak için global ağ topoloji bilgisi ve ağ trafik karakteristiği gibi bilgilere ihtiyaç duyulmaktadır. Bu bilgiler sağlandığında, ağ üzerinde konuşan iki uç düğüm kuyruklarında uygun paket iletim planı, deterministtik ağ iletim yollarının hesaplanması, paket replikasyonu gibi işlemler gerçekleştirilebilir. Bu algoritmaları gerçekleştirmek, kuyruklarda oluşacak gecikme problemini çözeceği gibi veri hacmini de yükseltecektir. Bunlara ek olarak, iki uç düğümün tasarlanan algoritmaya uyup uymadığından emin olunmalıdır. Bunun için bir izlem mekanizmasının gerçekleştirilmesi gerekir.

Yazılım tanımlı ağlar (YTA), ağ kontrolünü ve yönlendirme mekanizmasını veri düzlemi ve kontrol düzlemi adı verilen iki mantıksal düzleme ayıran bir ağ mimarisidir. Kontrol düzlemi, yazılım ile yönlendirme fonksiyonunu programlanabilir hale getirir. Bu metot, ağa birçok fonksiyonun modüler olarak eklenmesine olanak sağlar. YTA kontrolcüleri, YTA uyumlu anahtarlarla OpenFlow protokolünü kullanarak global ve güncel ağ durum bilgisini almak ve veri düzlemini yönetmek için haberleşir. Bu sayede, yazılım tanımlı ağlar dinamik ve yeniden konfigüre edilebilen bir ağ ve optimal kaynak yönetimi sağlar. Yazılım tanımlı ağların bu özelliklerinden faydalanmak, zaman kritik sistemlerin ihtiyaçlarını karşılamaya yardımcı olabilir.

TSN standardı, gerçek zamanlı paketlerin iletimi için Kısıtlı En Kısa Yol yönlendirme algoritmasını kullanır. Bu algoritma gerçek zaman gereksinimlerini karşılayan en kısa yol üzerinden yönlendirme yapılmasını sağlar (Ojewale ve Yomsi, 2020). Bu bildiride, yazılım tanımlı ağlar topoloji keşfi/yeniden keşfi, bağlantı maliyeti hesaplaması, yönlendirme yolunun rezervasyonu, paket replikasyonu gibi bazı TSN mekanizmalarının değerlendirilmesi için kullanılmıştır. Buna ek olarak, TSN' e ait yönlendirme mekanizması, yazılım tanımlı ağlar çerçevesinde Dijkstra En Kısa Yol Önceliği algoritması kullanılarak iyileştirilmiş ve daha dinamik bir yol hesaplaması araştırılmıştır.

Bu bildirinin devamı şu şekilde şekillendirilmiştir. İkinci bölümde, yazılım tanımlı ağ temelli gerçek zaman ağ çözümleri incelenmiştir. Üçüncü bölümde geliştirdiğimiz model tanıtılıp dördüncü bölümde elde edilen sonuçlar ile devam edilmiştir. Beşinci bölümde ise bildiri sonuçlandırılmıştır.

2. İlgili Çalışmalar

Gerçek zamanlı ağlar için optimal parametreleri seçmek, trafik tipi, linkteki gecikme ve ağ topolojisi gibi güncel ve global ağ bilgisine bağımlı bir işlemdir. Eğer bu ağ bilgileri sürekli değişkenlik gösteriyorsa, statik ve manüel konfigürasyon metotları en iyi sonucu vermeyecektir. Bu yüzden dinamik ve merkezi konfigürasyon yöntemleri daha iyi bir ağ optimizasyonu için elzemdir. Diğer yandan, gerçek zamanlı sistemler için geliştirilmiş farklı protokoller, protokole özgü donanımlara ihtiyaç duyar ve maliyet artar. Ek olarak, farklı gerçek zamanlı ağ protokolleri bir arada çalışamayabilir. Yazılım tanımlı ağ yaklaşımı bu sorunu çözebilir. Gerçek zamanlı sistemler için geliştirilmiş özel protokollerle kıyaslandığında, ticari kullanıma hazır anahtarlarla yakın performanslar elde edilebilir. Hata toleransı ve determinizm sağlarken maliyetleri düşürür ve daha dinamik bir ağ yapılandırması sunar. Bu sebeplerden dolayı yazılım tanımlı ağ temelli gerçek zamanlı haberleşme çözümleri hem endüstri hem de akademi için sıcak bir araştırma maddesidir.

Heise ve ark. (2015), OpenFlow protokolünü aviyonik ortamda kullanarak performans ve konfigürasyon yönünden değerlendirmişlerdir. Bu çalışmada OpenFlow' da yer alan "meter" komutu kullanılarak nasıl deterministtik bir davranış elde edilebileceği değerlendirilmiştir. Aviyonik ağ cihazları özel gereksinimlerinden dolayı pahalı oldukları için ticari kullanıma hazır anahtarları aynı performansı verecek şekilde yapılandıracak bir model geliştirmeye çalışmışlardır. Bu modelde Aviyonik Tam Dubleks Anahtarlı Ethernet protokolünde yer alan sanal link kavramı yazılım tanımlı ağ yaklaşımında eşlenmiştir.

Cevher ve ark. (2018), entegre modüler aviyonik (EMA) alt sistemleri için yüksek güvenilirlikle haberleşme sağlayan bir metot önermişlerdir. EMA platformları modüler mimari, deterministtik yerel ağ haberleşmesi, veri iletiminde katı zamanlama kısıtlamaları ve yüksek güvenilirlik gibi özelliklerle karakterize edilir. Deterministtik Ağ (Deterministic Networking, t.y.), bu özelliklerin sağlanması için danışılan anahtar teknolojilerden biridir. DetNet, EMA alt sistemleri arasında ağ arızası halinde deterministtik haberleşmeyi sürdürmek için yeniden yapılandırma kabiliyetine sahiptir. Bu güvenirlilik, iki uç nokta arasında, paketlerin ayrık iki

farklı yoldan kopyalanarak iletilmesiyle sağlanır. Yazılım tanımlı ağlar bu determinizmi yeniden yapılandırma ve yerleşik trafik politika mekanizmalarıyla sağlayabilir. Yazılım tanımlı ağ ve DetNet teknolojilerini kombine etmek yedeklilik sağlanmasına yardımcı olur.

Kumar ve ark. (2017), gerçek zaman gereksinimlerini karşılayabilmek için yazılım tanımlı ağların topolojiyi en tepeden görme imkânı ve merkezi yönetilebilme özelliklerini kullanmayı önermişlerdir. YTA mimarisi gecikmelerden haberdar bir mimari olmadığı için gerçek zamanlı haberleşmede uygulanabilecek bir yol aramışlardır. Ticari kullanıma hazır anahtarlar ve YTA yaklaşımını kullanarak gerçek zamanlı iletilmesi gereken paketleri anahtarlardaki önceliği yüksek kuyruklara göndererek, diğer paketlerden izole etmeyi ve uçtan uca gecikmede daha iyi performans sağlamayı hedeflemişlerdir.

3. Materyal ve Metot

3.1. Motivasyon

YTA temelli çözümler ile TSN arasındaki ilişki IEEE ve IETF gibi deterministtik ağlar için otomatik ve esnek bir yapılandırma iskeleti sunmaya çalışan organizasyonlar tarafından da desteklenmektedir. TSN' deki merkezi yapılandırma yaklaşımı, trafik planlama, kısıtlama temelli yönlendirme ve yedekli yollardan paket gönderimi gibi özellikleriyle de bağıntılıdır. IETF DetNet ise Katman-2 ve Katman-3' te deterministtik veri yollarından paket gönderimini garantiye almak için merkezi ağ kontrolünü önerir. Tüm argümanlar bir araya getirildiğinde, bu çalışmada merkezi yapılandırma modelini seçmek daha uygundur.

TSN ağlarına eklenecek her yeni mekanizmanın getirdiği ek parametrelerden doğan yapılandırma karmaşıklığı, TSN ağları için araştırılması gereken bir konudur. Ayrıca, parametreler için uygun değerleri seçmek trafik modeli, link gecikmesi ve ağ topolojisi gibi güncel ağ durum bilgilerine bağlıdır. Ağdaki bu bilgiler sık aralıklarla değişiyorsa, statik ve manüel yapılandırma metotları yetersiz kalacaktır. Bu yüzden YTA gibi dinamik ve merkezi bir yapılandırma modeli seçilmesi gerekir. YTA aynı zamanda ağ izlem ve link gecikme hesapları yapmaya da yardımcı olur. Yapılandırma dışında bir değer gördüğü durumda, ölçülen metriklere göre yeniden yapılandırma yoluna gidebilir.

3.2. Uygulama

OpenFlow, kontrolcü ve yönlendirme cihazları arasında bağlantı sağlayan bir haberleşme protokolüdür. Yazılım tanımlı ağlarda defacto standart olarak kabul görür. OpenFlow protokolünde paket yönlendirme için reaktif akış örnekleme ve proaktif akış örnekleme adında iki farklı yaklaşım vardır. İlk yaklaşımda bir anahtara yeni bir paket geldiğinde anahtarın akış tablosu gözden geçirilir. Eğer bu tabloda bir eşleşme görülmezse, anahtarda OpenFlow protokolüne ait "packet-in" paketi oluşturulup kontrolcüye gönderilir ve kontrolcü bu paketin düşürülmesine, gönderilmesine, gönderilecekse hangi porttan gönderilmesine karar verir. Reaktif modda kontrolcüye danışılır ve belirlenen ağ politikasına göre akış tablolarında kurallar oluşturulur. Diğer yandan, proaktif yaklaşımda gelen paketlere reaksiyon göstermez. Kontrolcü, paket gelmeden önce anahtarın akış tablolarına girdileri kaydeder. Anahtar gelen paketi bu tablodaki eşleşmeye göre iletir ya da düşürür.

Bu çalışmada, reaktif yaklaşım uygulanmıştır. Kontrolcü ağ topolojisini otomatik olarak keşfeder ve düğümler arasındaki ara yüzleri oluşturduğu bir listede tutar. Her ne zaman bir makine ya da anahtarın bağlantısı koparsa, kontrolcü topolojiyi yeniden keşfeder ve YTA yaklaşımından beklenildiği gibi güncel ve dinamik bir yapı sunar. Topoloji keşfine ek olarak, düğümler arasında link gecikmeleri de hesaplanır.

Bu araştırmada, benzetim ortamı olarak Mininet (Mininet, t.y.) kullanılmıştır. Mininet, basit bir makine üzerinde gerçekçi sanal ağlar oluşturup, bu ağları özelleştirip uygulamaya yardımcı olur. Bu özelliğiyle geliştirme, öğretme ve araştırma için kullanışlıdır. OpenFlow standardını destekleyen farklı tipte kontrolcü platformları mevcuttur. Bunlar Java, Python, C++ gibi farklı programlama dillerinde uygulanmıştır. Bu çalışmada kontrolcü olarak, Mininet sanal makinesiyle birlikte gelen Python tabanlı POX (POX Wiki, t.y.) kontrolcüsü seçilmiştir.

Önerdiğimiz modelde, kontrolcü beş farklı işleyiciden oluşur: ConnectionUp, ConnectionDown, LinkEvent, HostEvent ve PacketIn. Bu işleyiciler, kontrolcüde tanımlanmış Python sözlüklerini, listelerini ve ağ durum bilgisini taşıyan diğer değişkenlerin güncellenmesini sağlar. LinkEvent ve HostEvent işleyicileri "openflow.discovery" ve "pox.host_tracker" modüllerine bağımlıdır. Bu modüller, POX kontrolcüsüyle beraber gelen yerleşik modüllerdir. Bu yüzden Python' da kontrolcü nesnesi oluşturulmadan önce bu modüllerin başlatılması gerekmektedir.

1) ConnectionUp: Bu işleyici, kontrolcüye bir anahtar bağlandığında çağırılır. "Datapath ID (DPID)" adı verilen özgün bir kimlik numarası döndürür. "Datapath" sanal OpenFlow anahtarlarını tanımlar. DPID, bu anahtarları belirten 64-bit uzunluğunda bir kimliktir. Kontrolcüye bir anahtar bağlandığında, ConnectionUp işleyicisi anahtar ismini ve DPID kimliğini ilgili sözlüğe kaydeder.

2) ConnectionDown: ConnectionUp işleyicisinin tam aksine, bu işleyici herhangi bir anahtar ile kontrolcü arasındaki bağlantı koptuğu zaman ayağa kalkar. Bağlantısı kopan anahtar sözlükten silinir. Böylece, kontrolcü yönlendirme kurallarını güncellenmiş ağ topolojisine göre belirleyebilir.

3) LinkEvent: LinkEvent işleyicisi, iki anahtar arasında link tespit edildiği zaman çalışır. POX kontrolcüsünün içinde gelen "openflow.discovery" modülünü kullanır. Bu modül, DPID, birinci ve ikinci anahtarları ve birbirlerine bağlı oldukları port numarası gibi bilgileri döndürür. Bu bilgiler, ağ topolojisinin oluşturulması için kullanılır.

4) HostEvent: Topoloji ve anahtarlar arası port listesi tespit edildikten sonra, topolojiyi tamamlamak için anahtarlara bağlı makinelerin keşfedilmesi gerekir. HostEvent işleyicisi bu görevden sorumludur. Ağ üzerinde herhangi bir makine "pox.host_tracker"

modülü vasıtasıyla tespit edildiği taktirde bu işleyici ayağa kalkar. Makine tespitini başlatmak için, tüm makinelerin birbirlerine ICMP mesajı göndermesi gerekmektedir. Bu işlem Mininet' te "pingall" komutuyla başlatılabilir. Bir makine tespit edildiği takdirde, bu makine ile ilgili isim ve hangi anahtara hangi porttan bağlı bilgisi elde edilebilir. Bu bilgiler kullanılarak, ağ topolojisi tamamıyla keşfedilir.

5) PacketIn: PacketIn işleyicisi, yönlendirme, gecikme hesaplama, paket replikasyonu, yönlendirme yol rezervasyonu gibi işlemleri gerçekleştirdiği için bu çalışmadaki en önemli modüllerden biridir. Anahtara bir paket ulaştığında, bu modül ayağa kalkar ve paket tipi, kaynak ve hedef IP adresi, kaynak MAC adresi gibi bir çok bilgiyi döndürür. Bu bilgiler, bir sonraki bölümde detaylı olarak anlatılan Dijkstra modülünde kullanılarak, uçtan uca en uygun yönlendirme yolu hesaplaması yapılır. Bununla birlikte, ardışık iki anahtar arasında link gecikme hesaplaması da bu işleyici sayesinde gerçekleştirilir. Hesaplanan gecikmeler, ağ topolojisinin tutulduğu sözlük değerlerine link maliyeti olarak girilir. Her paket iletimi sırasında anahtarlarda bu işlem gerçekleştirildiği için, düğümler arası link gecikmeleri sürekli güncel tutulur. Bu güncelleme yapılırken linklerin simetrik olduğu varsayılır. Bu modül, her paket geldiğinde, işlemin yapıldığı anahtarın Dijkstra modülünün bulduğu en uygun yönlendirme yolu üzerindeki elemanlardan biri olup olmadığını kontrol eder. Eğer bu durum sağlanıyorsa, "of.ofp_packet_out" OpenFlow paketi kontrolcü tarafından anahtara gönderilir ve gelen paketi nasıl yönlendirmesi gerektiğini bildirir. Bu OpenFlow paketi içinde hesaplanan çıkış portu bilgisi yer alır.

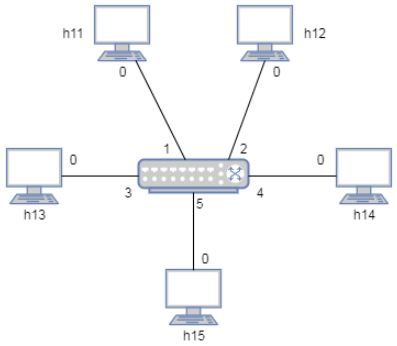
6) Dijkstra Modülü: Dijkstra modülü, Dijkstra' nın En Kısa Yol algoritmasını koşturur. Yukarıda anlatılan modüllerin sağladığı ağ topolojisi ve link maliyet bilgilerini kullanır. Ek olarak, kaynak ve hedef makine isimleri de bu modüle girdi olarak verilir. En uygun yol sonucu bir liste içinde döndürülür. Örnek olarak, h11 ve h61 makineleri arasındaki en uygun yol [h11 s1 s3 s6 h61] şeklinde belirtilir. Burada, h11 ve h61 sırasıyla kaynak ve hedef düğümlerini, s1, s3 ve s6 ise en uygun yol üzerinde bulunan anahtarları belirtir.

4. Araştırma Sonuçları ve Tartışma

TSN' i yazılım tanımlı ağ yaklaşımıyla gerçekleştirmek, topoloji keşfi, yönlendirme yol rezervasyonu, paket replikasyonu, link gecikme ölçümü, anahtar giriş politikası, optimal yönlendirme yolundan paket iletimi gibi konular ele alındığında, sonuç olarak özelleştirilmiş herhangi bir topolojide çalışabilen dinamik ve efektif bir çözüm sağlamaktadır.

4.1. Topoloji Keşfi ve Yeniden Keşfi

Topoloji bulma işlemi basit topolojilerden karmaşık topolojilere kadar birçok farklı ağ topolojisinde test edilmiştir. Ağ üzerindeki anahtarlar ve aralarındaki bağlantılar milisaniye mertebesinde saptanmıştır. Fakat makinelerin tespit edilmesi, düğümlerin birbirlerine ping mesajı atıp cevaplarını beklemesi sebebiyle, özellikle büyük topolojilerde zaman alan bir işlem halini aldı. Bu yüzden sistem devreye alınmadan önce gerekli kurulumların yapılması önerilmektedir. Kontrolcünün farklı topolojiler için döndürdüğü sonuçlar aşağıdaki loglarda gösterilmektedir.



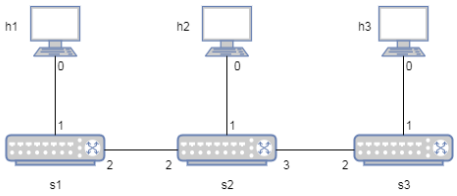
Şekil 1. Tek anahtar topolojisi

[dynamicController] Switch Port List: {‘h11’: {‘s1’: 0}, ‘s1’: {‘h11’: 1, ‘h12’: 2, ‘h13’: 3, ‘h14”: 4, ‘h15’: 5}, ‘h13’: {‘s1’: 0}, ‘h14’: {‘s1’: 0}, ‘h15’: {‘s1’: 0}, ‘h12’: {‘s1’: 0}}

"dynamicController" isimli kontrolcü topolojiyi Python sözlüğü içinde aşağıdaki gibi döndürür.

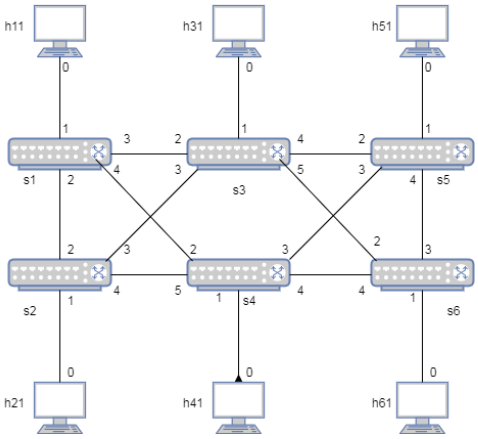
‘a’: {‘b’: 1, ‘c’:2}

Bu notasyon, "a" düğümünün "b" düğümüne kendi üzerindeki 1 numaralı porttan, "c" düğümüne ise 2 numaralı porttan bağlı olduğunu belirtir.



Şekil 2. Üç anahtarlı lineer topoloji

[dynamicController] Switch Port List: {‘s3’: {‘s2’: 2, ‘h3’: 1}, ‘s2’: {‘s3’: 3, ‘s1’: 2, ‘h2’: 1}, ‘s1’: {‘s2’: 2, ‘h1’: 1}, ‘h2’: {‘s2’: 0}, ‘h3’: {‘s3’: 0}, ‘h1’: {‘s1’: 0}}



Şekil 3. Kontrolcü testi için oluşturulmuş karmaşık topoloji

[dynamicController] Switch Port List: {‘s3’: {‘s2’: 3, ‘s1’: 2, ‘h31’: 1, ‘s6’: 5, ‘s5’: 4}, ‘s2’: {‘s3’: 3, ‘s1’: 2, ‘h21’: 1, ‘s4’: 4}, ‘s1’: {‘s3’: 3, ‘s2’: 2, ‘h11’: 1, ‘s4’: 4}, ‘h21’: {‘s2’: 0}, ‘s6’: {‘s3’: 2, ‘h61’: 1, ‘s5’: 3, ‘s4’: 4}, ‘s5’: {‘s3’: 2, ‘s6’: 4, ‘h51’: 1, ‘s4’: 3}, ‘s4’: {‘h41’: 1, ‘s2’: 5, ‘s1’: 2, ‘s6’: 4, ‘s5’: 3}, ‘h31’: {‘s3’: 0}, ‘h51’: {‘s5’: 0}, ‘h41’: {‘s4’: 0}, ‘h11’: {‘s1’: 0}, ‘h61’: {‘s6’: 0}}

Kontrolcü aynı zamanda makinelerin veya anahtarların durumunda herhangi bir değişiklik olduğunda topolojiyi yeniden keşfedip ilgili sözlüklerin güncellenmesini sağlayabilir. Bu sayede, anahtara yeni bir paket geldiğinde, bu paketin yönlendirilmesi için en uygun yol, güncel ağ durum bilgisi kullanılarak bulunmuş olur.

4.2. Link gecikme ölçümü

Her link için gecikme ölçümü bir anahtara paket geldiği esnada PacketIn işleyicisi içinde bir düğümden diğer düğüme iletilirken gerçekleştirilir. Ölçüm tamamlandıktan sonra, link maliyetleri güncellenir. Optimal yönlendirme yolu, paketler her anahtara ulaştığında hesaplandığı için, bu süre içinde optimal yol üzerinden iletilmeleri garanti altına alınmış olur.

4.3. Optimal yol üzerinden paket yönlendirme

Bu bölümde sonuçları göstermek için Şekil 3' te yer alan özel topoloji kullanılmıştır. "h11" makinesinden "h61" makinesine gönderilen ping mesajı gözlemlenmiştir. PacketIn işleyicisinin "s6" anahtarı üzerindeki logları Şekil 4' te yer almaktadır. Log incelendiğinde ilgili paketin h11 > s1 > s4 > s6 > h61 yolunu takip ettiği görülmektedir. Bununla birlikte, "s6" ve "s4" düğümleri arasındaki gecikme 10.55 milisaniye olarak ölçülmüştür. h11 > s1 > s3 > s6 > h61 ve h11 > s1 > s3 > s5 > s6 > h61 yolları da "h11" ve "h61" düğümleri arasındaki iletim için alternatif yollar olarak gösterilebilir. Fakat Dijkstra modülü optimal yolu link maliyetlerini göz önünde bulundurarak hesaplamıştır.

```
[dynamicController] ********************************************************
[dynamicController] Packet Type: IPv4
[dynamicController] Packet SRC IP: 10.0.0.1, DST IP: 10.0.0.6
[dynamicController] SRC Host: h11 DST Host: h61
[dynamicController] Dijkstra Path: ['h11', 's1', 's4', 's6', 'h61']
[dynamicController] Packet is on s6
[dynamicController] Current Node = s6 Previous Node: s4 DELAY: 10.5559825897 ms
```

Şekil 4. PacketIn işleyici logları

Hesaplanan optimal yol üzerindeki bir anahtarda sorun olması durumunda kontrolcünün nasıl davranacağını test etmek için "s4" anahtarı Mininet üzerinden switch s4 stop komutuyla kapatılır. Bu durumda, kontrolcü Şekil 5' te görüleceği üzere h11 > s1 > s3 > s5> s6 > h61 yolunu tercih eder. Çünkü hesaplanan link maliyetlerine göre h11 > s1 > s3 > s6 > h61 yolu daha az sayıda durağa sahip olmasına rağmen daha maliyetli bir yoldur.

```
[dynamicController] Switch s4 is disconnected
[dynamicController] ********************************************************
[dynamicController] Packet Type: IPv4
[dynamicController] Packet SRC IP: 10.0.0.6, DST IP: 10.0.0.1
[dynamicController] SRC Host: h61 DST Host: h11
[dynamicController] Dijkstra Path: ['h61', 's6', 's5', 's3', 's1', 'h11']
[dynamicController] Packet is on s6
[dynamicController] Current Node = s6 Previous Node: h61 DELAY: 10.4329586029 ms
```

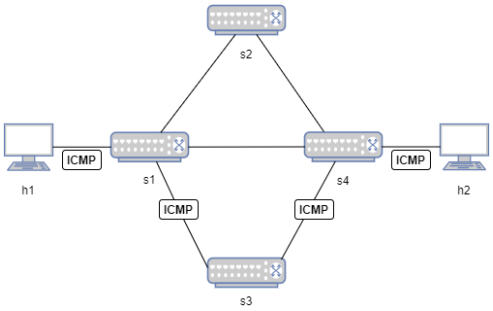
Şekil 5. s4 anahtarı kapatıldığında gözlemlenen PacketIn işleyici logları

4.4. Yönlendirme yolunun rezervasyonu

IEEE 802.1Qca Path Control and Reservation, Katman-2 ağlarda birden çok yolu yapılandırmak için kullanılan bir TSN protokolüdür. Kısıtlı En Kısa Yol yönlendirme algoritmasını kullanır (Ojewale ve Yomsi, 2020). Bu çalışmada, gerçek zamanlı paketler Dijkstra modülü tarafından bulunan optimal yol üzerinden yönlendirilmektedir. Yol rezervasyonu sağlayıp determinizmi emniyete almak için gerçek zamanlı olmayan paketlerin optimal yola erişimleri engellenmelidir. Her paket tipi için önceden tanımlanan yollar verinin korunmasına ve determinizme katkı sağlar. K-en kısa yol algoritması ikincil optimal yolun bulunması için önerilmiştir (Eppstein, 1994). Fakat, Dijkstra en kısa yol algoritması da bu iş için kullanılabilir. Önce optimal yol bulunur. Topoloji kopyalanır ve optimal yol üzerindeki link maliyeti sonsuza çekilir. Daha sonra, modifiye edilmiş topoloji üzerinde Dijkstra algoritması yeniden çalıştırılır ve elde edilen sonuç ikinci optimal yolu verir. Gerçek zamanlı olmayan paketler bu yol üzerinden iletilir ve farklı önceliğe sahip paketler farklı yollardan yönlendirilmiş olur.

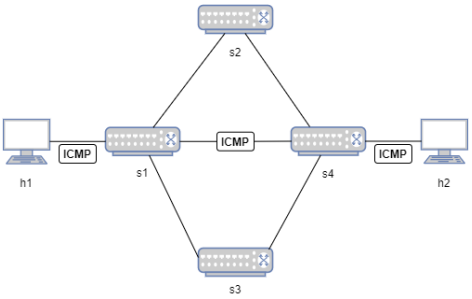
Önerdiğimiz modelde, gerçek zamanlı ve gerçek zamanlı olmayan paketler, Servis Tipi (Type of Service (ToS)) değerine göre sınıflandırılır. ToS değeri 0x10 (16) ise paket optimal yol üzerinden, değilse ikinci optimal yol üzerinden iletilir ve gerçek zamanlı olmayan paket muamelesi görür. Aşağıdaki örnekte, kolaylık olması için, tüm link maliyetleri eşit olarak tanımlanmıştır. Buna göre optimal yol h1 > s1 > s4 > s2 ve ikincil optimal yol h1 > s1 > s2 > s4 > h2 ve h1 > s1 > s3 > s4 > h2 olur.

mininet > h1 ping –c 10 h2



Şekil 6. Gerçek zamanlı olmayan paket iletimi

mininet > h1 ping –Q 0x10 –c 10 h2



Şekil 7. Optimal yol üzerinden gerçek zamanlı paket iletimi

4.5. Paket replikasyonu

Çalışmamızda uygulanan bir diğer TSN mekanizması IEEE 802.1CB Frame Replication and Elimination for Reliability protokolüdür. Bu protokol, her paketin kopyasının birden çok ayrık yol üzerinden iletilerek yedeklilik sağlanmasını önermektedir. Bu standart aynı zamanda DetNet' te de tanımlanmıştır. Bizim uygulamamızda, optimal yol hesaplandıktan sonra, paket ilk anahtarın tüm ara yüzlerinden kopyalanarak iletilir. Bu sayede birden çok ayrık yoldan gönderilen paketler, link veya cihaz arızası durumunda paket kaybını önler. Aşağıdaki örnekte, replike edilecek paketler 0x20 (32) ToS değeriyle belirlenmiştir ve link kayıp oranı %5' tir.

mininet > h1 ping –c 100 h2

100 packets transmitted, 59 received, **41% packet loss**, time 99236ms

Replikasyon olmadan elde edilen paket kaybı %41' dir.

mininet > h1 ping –Q 0x20 –c 100 h2

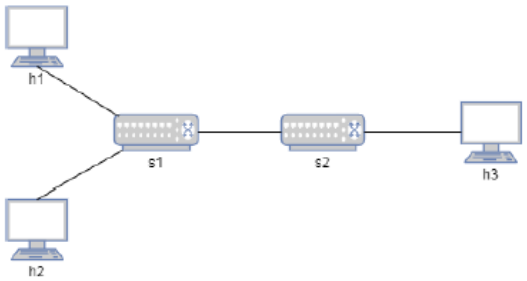
100 packets transmitted, 93 received, +195 duplicates, **7% packet loss**, time 99156ms

Replikasyon ile paket kayıp oranı %7 oranına düşürülmüştür.

4.6. Paket replikasyonu

IEEE 802.1Qci Per-Stream Filtering and Policing dayanıklılığı ve bant genişliği kaynak verimliliğini arttıran bir diğer TSN protokolüdür. Bu çalışmada son olarak OpenFlow "metering" fonksiyonu trafikler sınıflandırıldıktan sonra anahtarlar üzerinde uygulanmıştır. Anahtar giriş politikası filtreler sunarak veri akışını bloklar ya da azaltır. Bu fonksiyon gerçek zamanlı olmayan trafiklere uygulandığı takdirde ağ düğümleri üzerinde tıkanmayı engelleyecektir. Gerçek zamanlı olmayan aşırı miktardaki paketlerden kaynaklanan kuyruklar engellendiği takdirde gerçek zamanlı paket iletimi garanti altına alınabilecektir. Anahtarlar üzerindeki her ara yüz için önceden belirlenen veri hızları ayarlanarak, anahtarın davranışı gözlemlenmiştir.

OpenFlow her ara yüzde anahtar giriş politikasını gerçekleştirebilmek için ovs-vsctl komutunu uygun parametrelerle kullanıma sunar. Şekil 8' de görülen topolojide, 10.0.0.3 IP adresli "h3" alıcı düğümü, sırasıyla 10.0.0.1 ve 10.0.0.2 IP adresli "h1" ve "h2" gönderici düğümlerini ifade eder. “iperf” ağ performansını ölçmek için kullanılan ve iki düğüm arasındaki veri hızını hesaplayan bir araçtır. "h3" üzerinde iperf -s -p 4000 & ve iperf –s –p 5000 & komutları çalıştırılarak 4000 ve 5000 numaralı portlardan gerekli ölçümlerin tamamlanması için bağlantı yapılması beklenir. Bu örnekte "h2" düğümünün gerçek zamanlı paketler, "h1" in ise gerçek zamanlı olmayan paketler gönderdiği farz edilir. Bu sebeple, "s1" anahtarının "h1" ile olan ara yüzüne 100 Kbps, "h2" ile olan ara yüzüne 1 Mbps veri hızı politika olarak tanımlanır.



Şekil 8. Anahtar giriş politikası için örnek topoloji

Her iki düğüm aynı anda "h3" düğümüne paket göndermeye başladığında "h3", bağlanan düğümleri aşağıda belirtildiği gibi tespit eder.

Ölçümler yapıldıktan sonra, “h1” sonuç olarak

local 10.0.0.1 port 37236 connected with 10.0.0.3 port 4000

Interval	Transfer	Bandwidth
0.0-22.2 sec	256 Kbytes	**94.5 Kbits/sec**

“h2” düğümü ise,

local 10.0.0.2 port 48938 connected with 10.0.0.3 port 5000

Interval	Transfer	Bandwidth
0.0-11.4 sec	1.38 Mbytes	**1.01 Mbits/sec**

dönüşünü sağlar.

Elde edilen sonuçlar, ağ üzerindeki düğümlerin belirlenen politikaya uyduğunu, gerçek zamanlı paketlerin belirlenen veri hızında iletiminin sağlandığını ve gerçek zamanlı olmayan paketlerin aynı ağ altyapısı üzerinde çalıştığını gösterir.

5. Sonuç

Bu çalışmada, IEEE 802.1 Time-Sensitive Networking (TSN) standardının yönlendirme mekanizmasına Yazılım Tanımlı Ağlar (YTA) yaklaşımı kullanılarak katkıda bulunulmuş olup, TSN standardında yer alan bir çok özellik modelimiz üzerinde gerçeklenip sonuçları analiz edilmiştir. YTA yaklaşımını kullanmak ağ topoloji bilgisinin global görüntüsünü ve güncel ağ durum bilgilerinin kullanabilmesini sağlamıştır. Dördüncü bölümde sağlanan örneklerden görüleceği üzere, önerilen model gerçek zamanlı sistemler için dinamik ağ kontrolü ve optimal iletim mekanizması sağlamıştır. Sonuç olarak, havacılık, ulaşım, sağlık ve savunma gibi alanlarda kullanılmak üzere geliştirilmiş gerçek zamanlı haberleşme protokolleriyle benzer performans veren maliyet etkin bir çözüm sunulmuştur.

Kaynakça

Time-Sensitive Networking (TSN) Task Group. Erişim adresi https://1.ieee802.org/tsn/

Bello, L. ve Steiner W. (2019). A Perspective on IEEE Time-Sensitive Networking for Industrial Communication and Automation Systems. *Proceedings of the IEEE,* 1-27.

Ojewale, M ve Yomsi, P. (2020). Routing heuristics for load-balanced transmission in TSN-based networks. *ACM SIGBED Review*

Heise, P., Geyer, F., Obermaisser, F. (2015). Deterministic OpenFlow: Performance Evaluation of SDN Hardware for Avionic Networks. *11th International Conference on Network and Service Management*

Cevher, S., Mumcu, A., Caglan, A., Kurt, E., Peker, M., Hokelek, I., Altun, S. (2018). A Fault Tolerant Software Defined Networking Architecture for Integrated Modular Avionics. *2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*

Deterministic Networking (DetNet). Erişim adresi https://datatracker.ietf.org/wg/detnet/about/

Kumar, R., Monowar, H., Padhy, S., Evchenko, K., Piramanayagam, L., Mohan, S., Bobba R. (2017). End-to-End Network Delay Guarantees for Real-Time Systems using SDN *2017 IEEE Real-Time Systems Symposium (RTSS)*

Mininet. Erişim adresi http://mininet.org

POX Wiki. Erişim adresi https://noxrepo.github.io/pox-doc/html/

Eppstein, D. (1994). Finding the k Shortest Paths. *Department of Information and Computer Science University of California Tech. Report 26-94*