GAZİ ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

BM402 BİLGİSAYAR AĞLARI

Gamze Aksu

171180005

ÖDEV-2

MART 2022

İÇİNDEKİLER

Sayfa

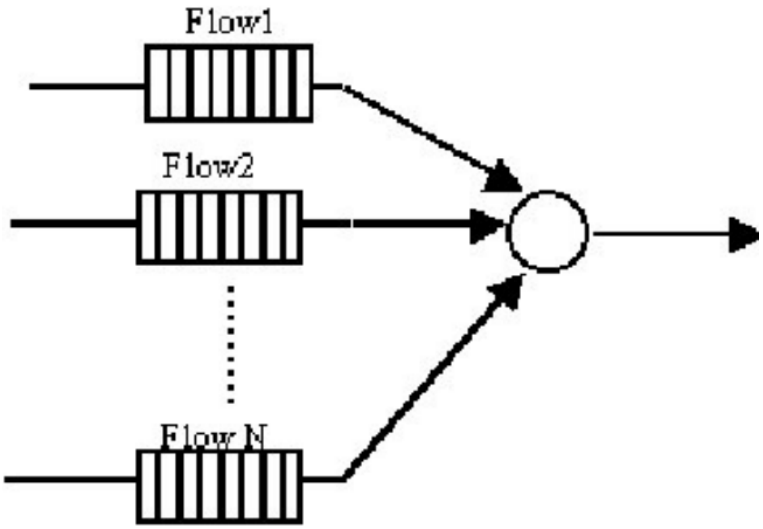
1. PAKET ANAHTARLAMA

Ağlar üzerinden gönderilmesi gereken büyük boyutlu bir verinin daha küçük parçalara ayrılarak gönderilmesine paket anahtarlama denir. Verinin bölünen küçük parçalarına da paket denir. İnternet içinde de paket anahtarlama kullanılarak veriler iletilir [1]. Verilerin paket olarak gönderilmesinin birçok faydası vardır. Buna en basit örnek olarak büyük bir dosya gönderilirken dosyada bir hata ortaya çıktığında tüm dosyanın tekrar gönderilmesi yerine hata olan küçük paketin gönderilmesi verilebilir. Ayrıca bu şekilde bant genişliği de daha verimli kullanılmış olur. Paketler iletilirken hedefe ulaşmak için farklı yollar kullanabilir. Bu durumda her zaman sıralı şekilde hedefe de ulaşamazlar. Hedefte bu paketler tekrar sıraya koyulur. Bunun gibi birçok sebepten paketlere başlık eklenir. Buna başka bir sebep olarak da yanlış gelen bir paketin tekrar istenmesi de verilebilir. [2]

Paketin başına eklenen başlıklardan birinde Servis Kalitesi (Quality of Service – QoS) başlığı yer alır. Ağ üzerinden gönderilen paketler için bir önceliklendirme de denilebilir. Gönderilen bazı paketler diğerlerinden daha önemlidir. Bazı uygulamalar için yavaşlama, gecikmeler kabul edilebilirken bazıları için kabul edilemez. Analoji olarak otoyol üzerindeki ambulanslar verilebilir. Bunun için trafik anında bazı paketlerin önce geçebilmesi paketlere QoS ile önceliklendirme verilir. [3]

Paketlerin ağlar arasında iletilmesi için yönlendirici (router) kullanılır. Bir yönlendiricinin kaç farklı giriş portu varsa o sayıda çıkış portları vardır [4]. Yönlendirici bir paketi işlerken aynı giriş portundan art arda gelen paketler beklemeye alınır. Paketler yönlendirici içerisinde bir kuyruk yapısında tutulur.[5] Bazı kuyruklama tipleri aşağıda sıralanmıştır:

- FIFO Kuyruklama
- Adil Kuyruklama (Fair Queueing)
- Sınıf Tabanlı Kuyruklama (Class Based Queueing)
- Ağırlıklı Adil Kuyruklama (Weighted Fair Queueing)



Şekil 1: Adil Kuyruklama

2. KUYRUK YÖNETİMİ ALGORİTMALARI

2.2. Aktif Kuyruk Yönetimi Algoritmaları

Kuyruk dolduğunda gelen paketler atılmaya başlanır. Paketlerin atılması verimsizdir. Bu bir tanıklık meydana geldiğinden çok fazla paketin atılacağı anlamına gelmektedir. Bu nedenle yaklaşan tıkanıklık durumlarını tespit etmek ve tıkanıklığı kontrolden çıkmadan aktif olarak yönetmek önem kazanmaktadır. Aktif kuyruk yönetiminde yönlendiriciler göndericilere yavaşlamaları gerektiğine dair sinyaller gönderir.

2.2.1.Drop Tail

Kuyruk yapısında en basit ve geleneksek kullanım "İlk gelen ilk gider." mantığıyla oluşturulmuş kuyruklardır. Bu mantık çerçevesinde ilk gelen paket ilk önce yönlendirilir. Bu mantık ile oluşturulmuş kuyruk yönetim yapısına Drop-Tail denir. Bir yönlendirici için kuyrukların alabileceği paket sayısı kısıtlıdır. Bu yüzden kabul edilebilecek paket sayısı dolana kadar paketler kabul edilir. Paket sayısı kabul edilebilir paket sayısını geçtiğinde ise son gelen paketler atılır. Günümüzde daha yönlendiriciler karmaşık kuyruk yapıları kullanmaktadır. Bu kuyruk yönetim yapısında iki temel sorun bulunmaktadır:

1. **Kilitlenme:** Drop-Tail yapısı bazen bir bağlantı veya birkaç bağlantının tüm kuyruğu ele geçirmesine izin verir. Bu şekilde diğer bağlantıların sıraya girmesi engellenir. Bu duruma kilitlenme denir. Senkronizasyon veya diğer zamanlama sorunları bu duruma sebebiyet verebilmektedir.

2. **Kuyruk Doluluğu:** Drop-Tail yapısında sadece kuyruk tam dolup da bir paket atıldığında tıkanıklık sinyali verilir. Bu olduğu için kuyruklar uzun süre boyunca dolu veya neredeyse dolu durumunda kalır. Kuyruk yönetiminde önemli bir amaç da kuyrukların boyutunu küçük tutmak olduğu için bu durum Drop-Tail kuyruk yönetim yapısının temel bir sorunu olmuştur.

Ağlara olan talep arttıkça Drop-Tail gibi etkin bir şekilde yönetimi olmayan bir yapı artık tıkanıklık yönetimi için uygun olmamaya başladı. Bu yüzden ilk AQM algoritması olan RED (Random Early Detection) geliştirildi. [6]

2.2.2. RED (Random Early Detection)

Drop-Tail yapısındaki kuyruk doluluğu sorununda TCP'nin kuyrukta bir doluluk olduğunu anlaması için bir paketin atılması gerekmektedir. Yönlendiricide atılan bir paket için oluşturulan sinyalin kaynakta algılanması için geçen zamanda kaynaktan bir sürü paket ağın destekleyebileceği hızdan çok daha fazla bir hızla yollanabilir. Bu durumda gönderilen çok sayıdaki paket atılır. RED yapısında ise tıkanıklığın başlangıcındayken erkenden tıkanıklık algılanır ve ana kaynağa bildirim gönderilir. TCP erkenden tıkanıklık olduğunu anladığı için paket gönderimi akışını azaltır. RED kuyruklardaki paketlere göre ortalama boyutu hesaplayarak rastlantısal olarak tıkanıklıktan haber verilecek kaynak belirler.[6]

RED'in tasarlanmasındaki amaçlar şu şekilde sıralanabilir:

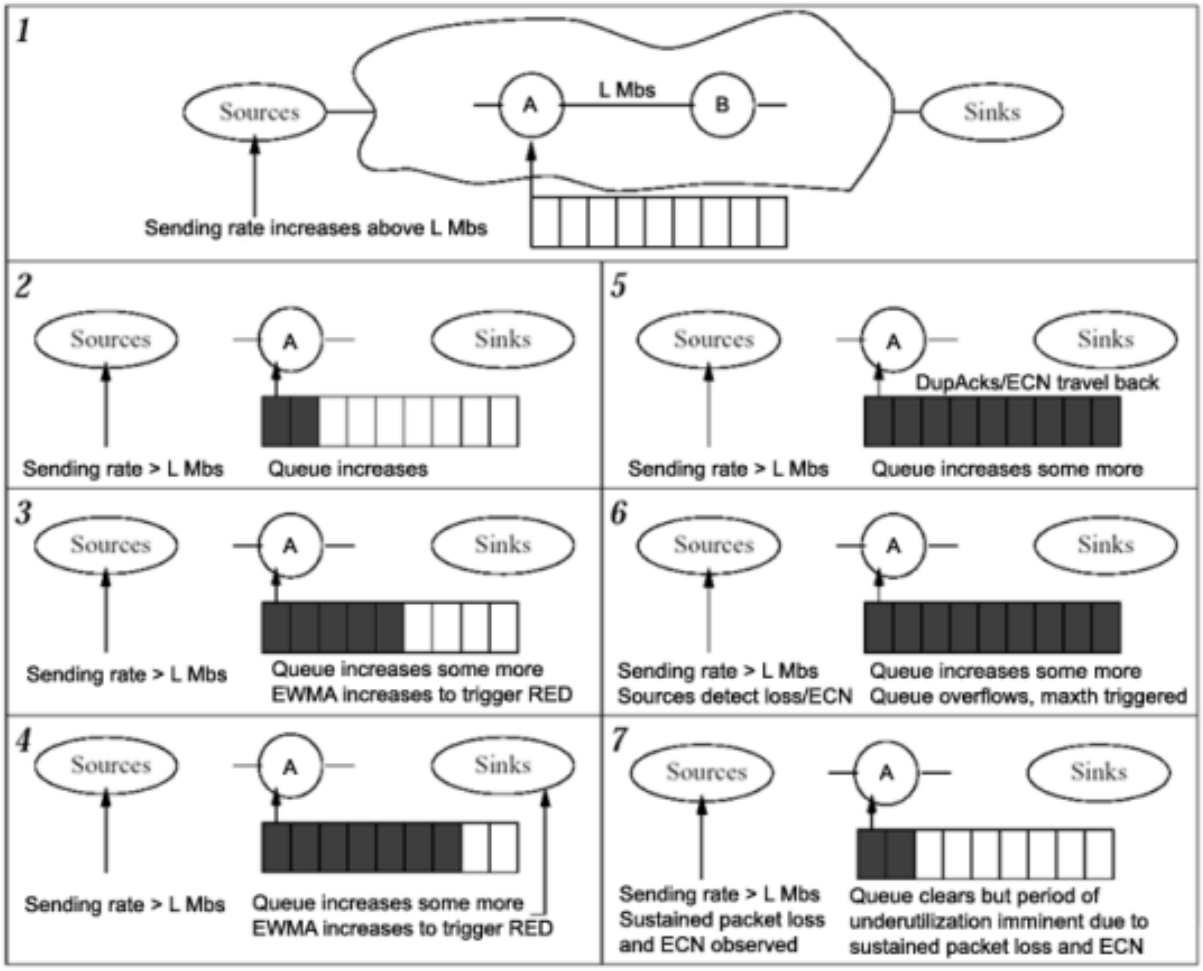
- Paket kaybını ve kuyruk gecikmesini en aza indirmek
- Kaynakların global senkronizasyonunu önlemek
- Yüksek bağlantı kullanımını sürdürmek

RED kuyruk yönetim yapısında kuyrukların taşması beklenmeden paketler olasılıksal olarak atılır. Paketlerin atılması için istatistiksel yöntemler kullanılır. Paketlerin olasılıklı olarak atılması bir kaynağın kuyruğu sabit boyutta tutmaya yetecek kadar yavaşlamasına neden olur. Böylece kaybolan paketlerin sayısı azaltılır. RED paketleri atarken iki önemli karar vermesi gerekmektedir. Bunlar paketlerin ne zaman atılacağı ve hangi paketlerin atılacağı kararlarıdır. RED yapısında ortalama kuyruk boyutu hesaplanır ve kontrol edilir. Ortalama boyut bir eşik değerini geçtiğinde paketler atılır. Kuyruğa yeni bir paket geldikçe bu ortalama boyut hesaplama işlemi tekrarlanır. Paketlerin atılması için tanımlanan eşik değerleri şunlardır: [7]

- Minimum Eşik: Bu eşik değerinde hiçbir paket atılmaz.
- Maksimum Eşik: Bu eşik değerinde tüm paketler atılır.

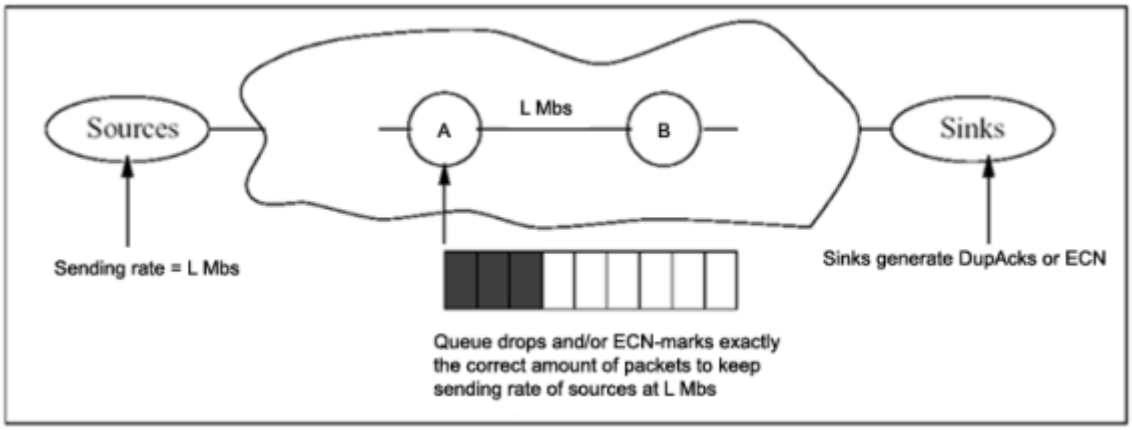
Bazen paketler atılmadan önce de kaynağa kuyruk boyutu hakkında bilgi verilebilir. Bunu kullanan Açık Tıkanıklık Bildirimi (Explicit Congestion Notification – ECN) yapısı bulunmaktadır. [8]

Şekil 2'de çok sayıda TCP kaynağı ile birlikte az miktarda ara bellek alanı kullanıldığında ortaya çıkabilecek bir senaryo gösterilmiştir. Toplu TCP kaynağının paket gönderme hızına ve ara bellek alanının boyutuna bağlı olarak büyük miktarda paket kaybı ve ECN işaretlemesi yapılabilir. Bu sorunu çözmek için arabellek alanının arttırılması gerekmektedir.



Şekil 2: RED örneği

Şekil 3'te ideal bir senaryonun nasıl olması gerektiği gösterilmiştir. RED yapısının bu ideal senaryoya ulaşabilmesi için yeterli arabellek alanına ihtiyacı vardır. Ayrıca parametrelerinin de doğru bir şekilde belirlenmiş olması gerekmektedir.



Şekil 3: RED yapısında ideal senaryo

Her paket geldiğinde
Ortalama kuyruk boyunu (avg) hesapla
$min_{th}<avg<max_{th}$ ise
p_a paket düşürme olasılığını hesapla
Paketleri p_a olasılığı ile işaretle
$max_{th}\leq avg$ ise
Paketi işaretle

Şekil 4: RED algoritması

2.2.3. BLUE Algoritması

BLUE algoritması tıkanıklığın tespit edilmesi için kuyruk boyutuna değil de paket atılmaları ve hattın geçmiş kullanım durumu gibi parametreler kullanır. BLUE algoritmasında p_m parametresi paketleri işaretlemek için kullanılır. Bir olasılık belirtir ve kuyruğun boş kalması ya da kuyruğun taşmasına göre değiştirilir. Paketler kuyruğun dolu olması sebebiyle hep atılıyorsa bu paket işaretleme olasılığının arttırılması gerektiği anlamına gelir ve p_m arttırılır. Tersi durumda, yani kuyruk boş duruma yaklaşıyorsa BLUE algoritmasına göre p_m değeri azaltılır. [8]

BLUE algoritmasının çalışabilmesi için 3 farklı parametre bulunmaktadır:

- freze_time: Kuyruktaki dalgalanmaların, anlık değişimlerin göz ardı edilmesi için kullanılan bir parametredir. Sabit bir sayı olarak belirlenir. Kuyruktaki değişimlere gereğinden büyük tepkilerin verilmesini önlemek amacındadır. Kuyruğun belirli aralıklarla güncellenmesini sağlar.

- d_i : Paket işaretleme olasılığının arttırılması gerektiğinde kullanılır.
- d_d : Paket işaretleme olasılığının azaltılması gerektiğinde kullanılır.

Şekil 5'te BLUE algoritması görülmektedir. BLUE algoritması RED algoritmasına göre daha küçük kuyruk boyutlarında paket atılma sayısını azalttığı ve hattı daha verimli kullandırdığı görülmüştür.

```
Paket kaybında (veya Q_len>L):
    if ((now – last_update) > freeze_time )
        p_m = p_m + d_i;
        last_update = now;
Link boş kaldığında
    if ((now – last_update) > freeze_time )
        p_m = p_m + d_d;
        last_update = now;
```

Şekil 5: BLUE algoritması

2.2.4. REM (Random Exponentıal Marking) Algoritması

REM algoritmasında tıkanıklık ölçümü ile performans ölçümü birbirinden ayrılmaktadır. Tıkanıklık ölçümünde artan bant genişliği isteği ve kullanıcı sayısı birer parametre olarak kullanılır. Performans ölçümünde ise tüm kullanıcılarının sabit bir performans düzeyinde kalması bir parametredir. REM algoritmasında kayıp paket ve gecikmelerin yok sayılabilecek düzeyde olmasını ve bant genişliğinin maksimumda kullanılmasını amaçlamaktadır. RED algoritmasından farkı tıkanıklık belirleme yöntemi ve paketleri işaretleme algoritmasıdır. REM algoritmasında her bir çıkış kuyruğu "price" olarak adlandırılan bir değer tutar. Bu değer tıkanıklığı ölçmede kullanılır. Aynı zamanda bu değer paketleri işaretleme olasılığına karar vermek için de kullanılır. Giriş oranı ve bağlantı kapasitesi farkına göre price değeri değiştirilir. Giriş oranı ve bağlantı kapasitesi farkı pozitif ise price değeri arttırılır, değilse azaltılır. [9]

TCP Reno, tıkanıklığı belirlemek için kuyrukta taşma olup olmadığına bakar. TCP Vegas ise kuyruktaki gecikmelere bakar. REM algoritmasında tıkanıklık belirlenmesi için price değerine bakılır. REM algoritmasında paket kaybı, kuyruk uzunluğu ve gecikmeler gibi performans ölçümüne dair parametreleri kullanmadığı için performans ölçümü ile tıkanıklık ölçümü birbirinden ayrılmış olur. [9]

2.2.5. CHOKe Algoritması

CHOKe algoritmasında yanıt vermeyen akışların kontrolünü basit tutmak amaçlanmıştır. Bu algoritma RED algoritması ve Drop-Tail algoritmasının birleşiminde küçük değişiklerin yapılmasıyla elde edilmiştir. Bir paket geldiğinde ortalama kuyruk boyutu eşik değerinden büyükse CHOKe algoritmasına göre kuyruktan rastgele bir paket çekilir ve gelen paketle karşılaştırılır. Aynı akıştan geliyorlarsa iki paket de atılır. Eğer aynı akıştan gelmiyorlarsa RED algoritmasına göre hesaplanan olasılık ile kabul edilir. Bu işlemlerin yapılmasının ardındaki sebep FIFO kuyruğunda yanıt vermeyen akışların gönderdiği paketlere sahip olma olasılığının daha fazla olmasıdır. Aynı zamanda karşılaştırma için seçilme olasılıkları da daha fazladır. Ancak CHOKe algoritmasında sadece tıkanıklık olduğunda yanıt vermeyen akışları kontrol edebilir. [10]

2.3. Proktif Kuyruk Yönetimi Algoritmaları

2.3.1. GREEN Algoritması

GREEN algoritması TCP paketlerinin proaktif olarak atılmasının sağlamaktadır. Bu işlem TCP'nin kararlı durumdaki davranışlarından elde edilen bilgiler ile yapılır. Böylelikle TCP akışları arasında hat adaletli olarak dağıtılmış olur. GREEN algoritması tıkanıklığın tahmin edilmesi değil de aktif olarak tespit edilmesi üzerine kurulmuştur. Yani tıkanıklıktan kaçınmak yerine tıkanıklığı önleme amacındadır. Tüm bunlara ek olarak GREEN algoritması kuyruk uzunluğunu da düşük tutar. [9]

KAYNAKÇA

1. Türk,E. (2019) Devre Anahtarlama ve Paket Anahtarlama https://esenbaharturkay.medium.com/devre-anahtarlama-ve-paket-anahtarlama-fda60601590b
2. https://www.netinbag.com/tr/internet/what-is-packet-switching.html
3. Wikipedia katılımcıları (2021). QoS (port önceliği). *Vikipedi, Özgür Ansiklopedi*. Erişim tarihi 14.25, Mart 18, 2022 url: https://tr.wikipedia.org/wiki/QoS_(port_%C3%B6nceli%C4%9Fi)
4. https://nerdbook.wordpress.com/2018/10/20/network-ve-protokoller/
5. Metin,B., Deliç.H, (n.d.) IP Ağlarında Kuyruklama Yöntemlerinin Servis Kalitesine Etkisi https://www.emo.org.tr/ekler/6051b3bfe716cc4_ek.pdf
6. Özekes.S, (2005) EVALUATION OF ACTIVE QUEUE MANAGEMENT ALGORITHMS https://ticaret.edu.tr/uploads/kutuphane/dergi/f7/M00105.pdf
7. Socrates, C. Beulah Devamalar, P.M. Kannamma Sridharan, R. (2014). Congestion Control for Packet Switched Networks: A Survey http://www.ijsrp.org/research-paper-1214/ijsrp-p3608.pdf
8. Okuroğlu. B, (2000), TCP/IP'DE AKTİF KUYRUK YÖNETİM MEKANİZMALARI VE İNTERNET İÇİN KALİTELİ HİZMET
9. Şimşek. M, (2012) PAKET ANAHTARLAMALI AĞLARDA HİZMET KALİTESİ TABANLI ULAŞTIRMA PROTOKOLÜ VE KUYRUK YAPISI GELİŞTİRİLMESİ VE UYGULAMASI https://dspace.gazi.edu.tr/bitstream/handle/20.500.12602/177692/fc8c0233f00a171a587c6b6bcdbcb93e.pdf?sequence=1&isAllowed=y
10. Prakash, S.J. Amalarethinam, G. Raj, G.D. (2011) QoS Congestion Control AQM Algorithms: A Survey https://www.researchgate.net/publication/269938906_QoS_Congestion_Control_AQM_Algorithms_A_Survey