TCP/ RENO /RED/AQM/BLUE SIKIŞIKLIK MODELLERİ

Prof.Dr. Remzi YILDIRIM

2016-AYBÜ-ANKARA

TCP/ RENO /RED/AQM/BLUE SIKIŞIKLIK MODELLERİ

Prof.Dr. Remzi YILDIRIM

Ankara Yıldırım Beyazıt Üniversitesi

(DERS NOTU / ÖDEV)

2016-ANKARA

İÇİNDEKİLER

BÖLÜM 1

AĞ TEMELLERİ

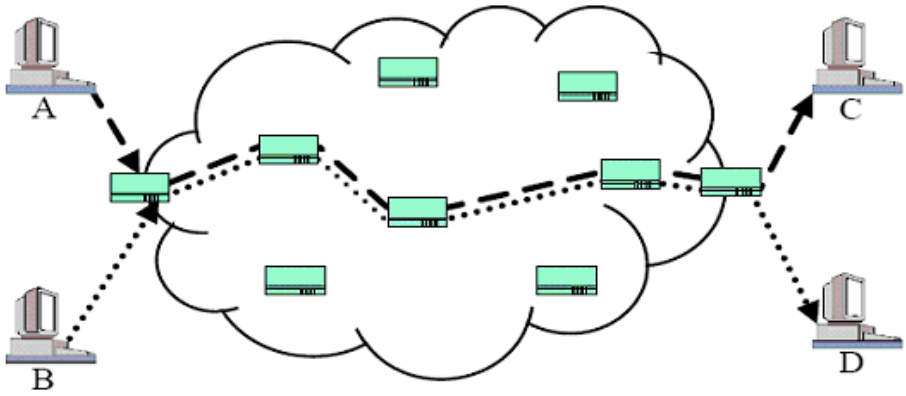
İnternette bilgisayarlar, mesajlarını paket adı verilen birimlere bölerek birbirleriyle haberleşirler. İnternette her bir bilgisayarda diğer bir bilgisayara bağlanmak için kablo olmadığından, paketler kaynaktan gidilecek yere *yönlendirici* adı verilen ara bilgisayarlar ile taşınırlar. Herbir paket kaynağın ve gidilecek yerin adresini içerir böylece yönlendiriciler paketin nereye iletileceğini ve kimin gönderdiğini bilirler. Paketler yönlendiricilere işlem yapılmasından ve gönderilmesinden daha hızlı ulaştığı zaman, yönlendirici paketleri bir sırada saklar. *Ağ sıkışması,* bu sıradaki paketlerin sürekli olan bir periyodda artması sonucunda oluşur. Sıranın daha uzun olması, sıranın sonundaki paketin transfer edilmesi için daha çok beklemesi demektir, bundan dolayı gecikmeleri artar. Ağ sıkışması sıranın tamamen dolmasına sebep olur. Bu şekilde sıra dolduğunda, gelen paketler düşürülür ve asla gidecekleri yere ulaşamazlar. TCP gibi iletişimi kontrol eden protokoller, bu paket kayıplarını dikkate almalıdırlar.

İnternetteki web sayfalarının transferi bir *bağlantı-tabanlı (connection-oriented)* servisdir. Bu *paket anahtarlamalı (packet-switched)* bir ağ üzerinde çalışır, transfer protokolü olarak TCP kullanır. Bu bölümün kalanında, paket-anahtarlamalı bir ağı, bağlantı tabanlı iletişimi ve TCP tanımlanacaktır.

1.1 Devre Anahtarlama ve Paket Anahtarlama

Bir iletişim ağı devre anahtarlamalı (circuit-switching) yada paket anahtarlamalı (packet-switching) tabanlı olabilir. Devre anahtarlamalı da, veri transferinin başlayabilmesi için kullanılacak tüm ağın rezerve edilmiş olması gerekmektedir. Buna bir örnek olarak telefon sistemlerini verebiliriz. Ağda, sınırlı bant genişliği kapasitesi vardır, aynı anda sabit sayıda bağlantıyı destekleyebilir. (ağın bant genişliğine ve herbir bağlantı tarafından ayrılan bant genişliği miktarına bağlıdır). Eğer bant genişliği ayrılmışsa ve

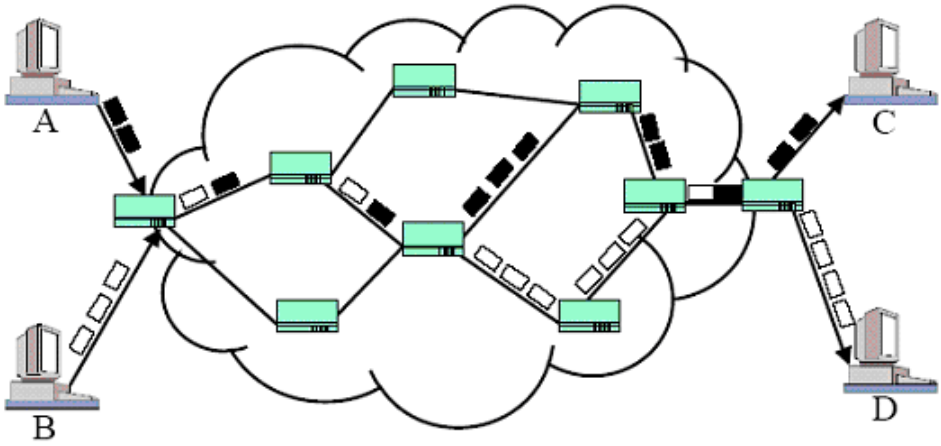
bağlantı kurulmuşsa ama veri transferi başlamamışsa, başka bir bağlantı boş da bulunan bant genişliğini kullanamaz çünkü o başka bir bağlantı tarafından ayrılmıştır. Şekil 1.1 de bir devre-anahtarlamalının örneği gösterilmiştir. A bilgisayarı, C bilgisayarı için bir ayırma yapmıştır ve B bilgisayarıda, D bilgisayarı için bir ayırma yapmıştır. Bu devrelerin her ikiside aynı yönlendirici boyunca gitmektedir. Yol her ikisi tarafından da kullanılmaktadır. Devre anahtarlamalı ağlarda, çoklama (Multiplexing) ile bant genişliği zaman bölümlerine (time-division multiple access) yada frekans bölünmelerine (frequency-division multiple access) ayrılır. Eğer bir bağlantı boşsa ve ayırma etkin iken, boş bant genişliğini başka kullanan bağlantı yoksa çoklama söz konusu değildir. Bunlara ilaveten, kaynaklar önce ayrılmıştır ve trafiğin bant genişliği kapasitesini aşmayacağı garanti edilmiştir, sıra yoktur ve dolayısıyla sıkışmada yoktur.



Şekil 1.1. *Devre – Anahtarlamalı örnek.*

Paket anahtarlama, internette kullanılan data transfer metodudur. Paket anahtarlamada ayırma yapılmaz ama verinin dağıtılacağının da garantisi yoktur. Ağ datayı hızlı bir şekilde transfer etmek için *"en iyi gücünü(best-effort)"* ortaya koyar. Datanın transfer edilebilmesi için paket adı verilen küçük parçalara bölünmesi gerekir. Herbir paket kaynağın ve gidilecek yerin adresini içerir. Ağdaki ara bilgisayarlar *yönlendiriciler* olarak adlandırılır ve ağda paketlerin kaynaktan ulaşılmak istenen yere iletilmesi için kullanılırlar. Ayırma olmadığından, birçok kaynaktan paketler araya girebilir. Bu işlem *istatiksel çoklama* olarak adlandırılır ve istenen sayıdaki bilgisayarın bilgilerini aynı anda aralarında değiş-tokuş etmesine izin verir. Paket anahtarlamalı bir örnek şekil 1.2 de gösterilmiştir. Şekil 1.1 deki gibi A bilgisayarı C bilgisayarına veri gönderiyor ve B

bilgisayarıda D bilgisayarına veri gönderiyor. A'dan giden datalar siyah paketler olarak B'den giden datalarda beyaz paketler olarak gösterilmiştir. Paket anahtarlamalı ağ ile, çeşitli bağlantılardaki paketler ağın bant genişliği kapasitesini paylaşırlar. Devre anahtarlamalı ağların aksine, paketler diğer bağlantılardan boşta kalana kapasiteyi kullanabilirler. Bunun yanında varışların oranı yönlendiricilerdeki gönderme kapasitesini arttırırlar, bir sıra oluşturulur, sıkışma ve paket kayıpları olabilir.



Şekil 1.2. *Paket – Anahtarlamalı örnek.*

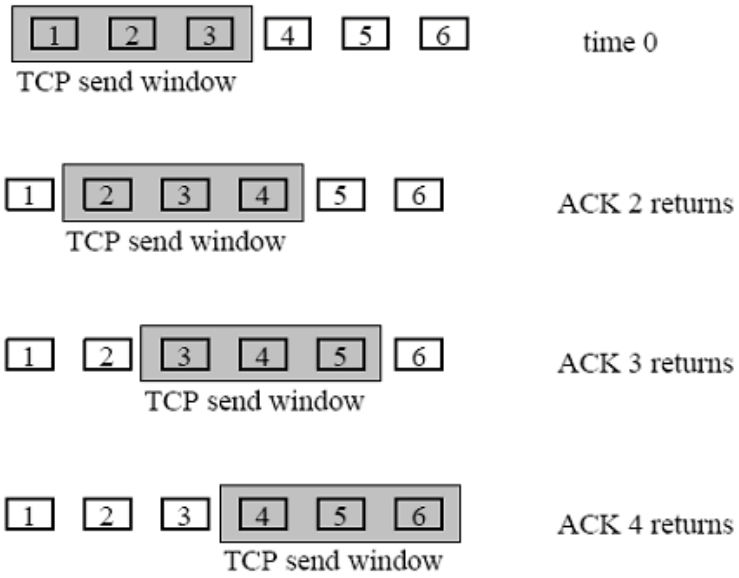
Bağlantısız sistemlerde, el sıkışma yoktur. Bilgi göndericiden alıcıya doğruluğu onaylanmadan gönderilir, buda daha çok mektubun yazılıp posta kutusuna koyulması işlemine benzer. Bağlantısız sistemler posta servisi gibidir, gönderici ne zaman alındığını veya alınıp alınmadığını bilemez.

İnternet bağlantısız servisleri UDP ile bağlantıya yönelik servisleride TCP ile sağlar. TCP; email, veri dosyaları ve web sayfaları vb. içeren birçok uygulamayı transfer için kullanılır. Araştırmam doğrudan TCP ile ilgili olduğundan bağlatıya yönelik servisler üzerinde odaklanacağım.

1.2 TCP Taşıması

TCP internetin bağlantıya yönelik taşıma servisidir. İnternet paket-anahtarlamalı olduğundan, TCP bilgiyi gönderilebilmesi için TCP parçalarına (*segmentlerine)* böler, bunlar daha sonra kaynak ve gidilecek yerin adreslerinide içeren paketlerde paketlenir. TCP kaynakdan gidilecek yere güvenli bir şekilde dağıtılacağına söz verir. Güvenli bir dağıtımından emin olmak için tüm parçalar alıcı tarafından doğrulanır (ACK gönderilir). Göndericiden gönderilen her parça bir sıra numarası (sequence number) ile gönderilir. TCP deki ACK'ler, sıra numarasına sahip tüm byte'ların toplu olarak alındığını bildirmek amacıyla kullanılır (ACK, alıcı tarafından göndericiden beklenen sıra numarasını tanıtır). Bir TCP göndericisi, bu ACK'leri kabaca, gönderilen fakat daha henüz doğrulanmayan dataların kayıtlarını tutmak ve *gönderilecek pencereyi (send window)* hesaplamak için kullanır. Dağıtımı sağlamak amacıyla, TCP alıcısı alınan her parçayı, hatalı sırada alınan sıra numaralarındaki boşluklar doldurulunca kadar tamponda (buffer) tutmak zorundadır.

TCP tarafından gönderilen mesajlar keyfi büyüklerde gönderilir, ama alıcının bu parçaları tutmak için kullandığı tampon sınırlıdır. Herbir ACK'da, alıcının tamponunda ne kadar boş yer kaldığının bilgisi vardır. Bu *alıcının ilan edilen penceresi*'dir. Alınan bu pencere güncelleninceye kadar, gönderici burada bildirilen miktardan daha büyük verinin gönderilmesine izin vermez (alıcının penceresinde hiç oda yoksa, yeni data gönderilmez). Bu TCP *akış kontrolünü (flow control)* nasıl sağlamaktadır ? Akış kontrolünün amacı göndericinin, alıcının tampon kapasitesinin üzerinde veri göndermemesinden emin olmasıdır. Akış kontrolü için gönderilen pencere, alıcının penceresinden geniş olmamalıdır. Eğer bir TCP göndericisi 6 parçaya bölünmüş bir mesaj göndermek istiyor ve alıcının pencere büyüklüğü 3 parçaysa, gönderici ilk başta 1-3 arasındaki parçaları gönderir, şekil 1.3. parça 1 için ACK geri döndüğü zaman, 4. parçayı gönderir. Bu işlem 6 parça da gönderilinceye kadar devam eder.



Şekil 1.3. *TCP Gönderilecek pencere.*

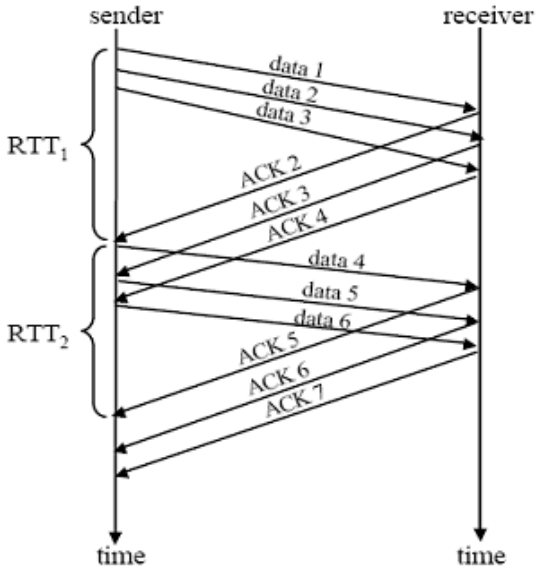
Gönderilen pencerenin büyüklüğü gönderici tarafından transfer edilen dataların oranını yürütür. Bunu göstermek için önceki örnek farklı bir biçimde şekil 1.4 de gösterilmiştir. Paralel çizgiler gönderici ve alıcıyı belirtirler, zaman ilerlemesi şeklin altında gösterilir. Burada da alıcının penceresi 3 parçadır. Bir parçanın gönderilmesi ve ACK'inin alınması arasındaki zaman *gidiş - dönüş zamanı (round-trip time, RTT)* olarak adlandırılır. 3 parçalık başlangıç penceresi ile, TCP ilk 3 parçayı arka arkaya gönderir. Bu parçalar için ACK'ler yakın boşluklarla alınırlar. RTT_1 parça 1 in RTT'sini belirtir, ve RTT_2 parça 4 ün RTT'sini belirtir. Bu transferin gönderme oranı RTT başına 3 parçadır, bundan dolayı her RTT'de ortalama 3 parça gönderilir. Daha genelleştirirsek, TCP göndericisinin oranı,

$$Oran = w / RTT \qquad (1.1)$$

şeklinde belirtilmiştir. Eşitlik (1.1)'de kullanılan w pencere büyüklüğüdür.

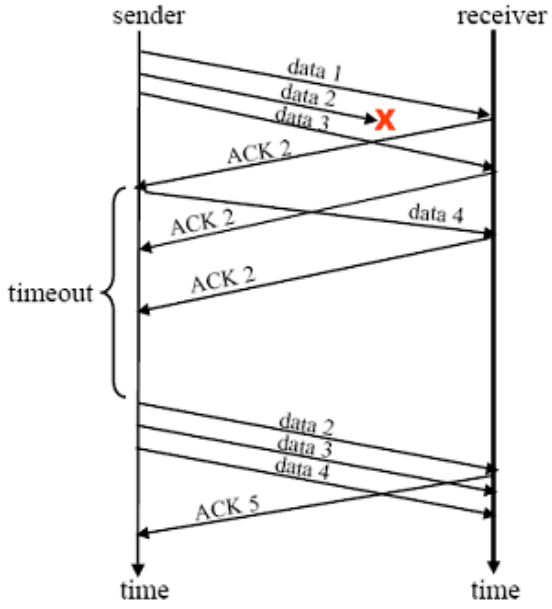
Şekil 1.5'de bir paketin düşmesi ve geri kurtarılmasının örneği gösterilmiştir. Bu örnek şekil 1.4 gibi 3 parçalık pencereyle başlar. Parça 2 ağ tarafından düşürülür. Parça 3'ün alındısı, alıcının parça 2 isteği için aynı ACK göndermesine sebep olur. Parça 2'yi

isteyen ilk ACK alındığında parça 2 geri döner (parça 1'in alındısı) ve TCP göndericisi bir zamanlayıcı ayarlar. Bu örnek de, zamanlayıcının süresi yeni bir veri isteğinden önce biter (parça 2 den başka bişey), böylece gönderici parça 2'nin kaybolduğunu varsayar. Pencere ölçüsü 3 parça olduğundan, TCP parça 2'den başlayarak 3 parça gönderir. Bu gösterim *"Go-Back-N"* olarak adlandırılan hata kurtarma yaklaşımıdır. Parça 2 alındığında, parça 5'i isteyen ACK gönderilir.



Şekil 1.4. *TCP Gönderme Oranı.*

Geciken ACK'ler : Esas olarak, TCP alıcıları bir parça alındığı zaman, hemen bir ACK gönderirler. İki yönlü trafik ile, geciken ACK'lar, göndericiye geri gönderilecek veri oluncaya kadar beklemesine izin verir ve alıcı ACK'i veri parçalarının sırtında göndericiye geri iletir. Eğer ACK'ler çok fazla gecikirse, gönderici parçanın kaybolmasından şüphelenir. Bu gecikmeyi sınırlandırmak için, geciken ACK'ler bir zamanlayıcı çalıştırır, genellikle 200 ms'ye ayarlanır. Zamanlayıcı dolmadan, eğer alıcıdan, göndericiye veri parçası gönderilmemişse (ACK üzerinde), doğrulanmamış veri vardır, hemen bir ACK gönderilir. Ayrıca göze çarpan bir ACK eşiği vardır, genellikle 2 parçaya ayarlanır, böylece, eğer iki doğrulanmamış parça varsa hemen bir ACK gönderilir.



Şekil 1.5. *TCP düşme ve geri kurtarma.*

BÖLÜM 2

TCP SIKIŞIKLIK KONTROLÜ

Günlük yaşantımızda hepimiz belli zamanlarda sıkışma yaşarız. Sıkışma sadece bilgisayar ağlarında değildir. Örneğin araba ile otoyola girişte, alışveriş için faturamızı ödemekde vb. durumlarda sıkışma yaşayabiliriz. Bu örneklerdeki esas kural aynıdır. Kapıdan geçmek isteyen varlıkların sayısı kapının kapasitesinden büyüktür. Bilgisayar ağlarında ise bir yönlendiriciye gelen paketlerin sayısının, yönlendiricinin işlem yapabileceği sayıdan büyük olmasıdır, yada çıkışın girişden yavaş olmasıdır. Genellikle yönlendiriciler, gelen paketlerin işlem yapılıncaya kadar saklandığı bir tampon ile sağlanır. Bir yönlendiricideki sıkışmada, tampon taşması yaşanır, sonuç olarak da paket kaybolur.

2.1 Ağ Sıkışıklığı

İnterneti kullanan herkes gecikmeleri farketmiştir. Web trafiği için gecikmeler, web sayfalarının yavaş yüklenmesine sebep olurlar. Ses ve video'ların oynatılmasında, gecikmeler boşluklara yada oynatılması sırasında düzensizliklere neden olur. Bu gecikmeler genellikle, yönlendiricilerde, gelen oran giden hat hızını aşarsa, belli bir zaman periyodunda oluşan *ağ sıkışklığı (network congestion)* dan kaynaklanır.

Ağ sıkışıklığı, internetin tasarlandığı paket-anahtarlamanın bir yan etkisidir. Paket anahtarlama, farklı kaynaklardan verinin aynı yol boyunca iletilmesine izin verir. İnternette yönlendiriciler, paketlerin anlık alım oranı dış-sınır iletim oranından büyükse, paketleri tampona almak için sıralar kullanırlar. Bu sıralar first-in/first-out (FIFO) yani ilk giren ilk çıkar prensibine göre çalışır ve sınırlı bir kapasiteleri vardır. Bir paketi yönlendiricilerde tampona alındığında, ilk önce daha önceden sıralanan paketlerin iletilmesini beklemek zorundadır. Sıranın uzaması yani sırada daha fazla paket olması,

daha uzun sıra gecikmesi demektir. Sıra sınırlı olduğundan, gelen paketler eğer dolu bir sıraya ulaşıyorlarsa, sıra tarafından düşürülürler. İnternetteki çoğu sıra *"drop-tail"* dır. Yani, eğer sıra doluysa gelen paketler sadece düşürülür.

Ağ sıkışıklığı, yönlendiricilerdeki sonlu sıraların ölçülerinin artmasına, er geç bu sıraların dolmasına ve gelen paketlerin düşmesine sebep olurlar. Sıradaki gecikmeler, verinin göndericiden alıcıya dağıtılmasını yavaşça azaltır, uygulamanın performansı kullanıcının farkedebileceği kadar düşer. Paket düşmeleri, TCP akışları için özellikle problemdir. TCP sıralı ve güvenli bir dağıtım için söz verir, böylece bir TCP paketi düşürülürse, düşen paket alıcıya ulaşıncaya kadar sonradan gelen paketler alıcıya dağıtılmaz. Bir paket düştüğü zaman, TCP düşmeyi tespit etmek ve kaybolan paketi yeniden göndermek zorundadır. Bunların her ikiside zaman alır. TCP'nin güvenilirlik özelliğinden dolayı, kayıp paketler son kullanıcı için gecikmeyi arttırır.

Kullanıcının amacı paketlerini mümkün olduğunca çabuk bir şekilde göndermektir. Bunu yapmak için bağlantı yolunun kendi tarafı ve karşı tarafı arasındaki en yavaş parçasına göre göndermelidir. TCP protokolü bir iletim penceresi (transmission window) ile çalışır. Bu önceki gönderilen paketin ACK'i alınmadan gönderilebilecek paketlerin sayısıdır. Alıcı periyodik olarak kendi ilan ettiği pencereyi (advertised window) gönderir . Bu alabileceği paketlerin sayısıdır. Sıkışmadan otomatik olarak kaçınmak için iletim penceresinin seçilmesi ve düzenlenmesi gerekmektedir.

Son nokta gibi, yönlendiriciler de sıkışıklıktan kaçınmanın önemli parçasıdır, genellikle yönlendiricilerin, gelen paketlerin tutulduğu sınırlı tampon kapasiteleri vardır ve bu bellek dolduğunda sıkışma oluşur. Tampon dolduğu zaman, yükü azaltmanın tek yolu paketlerin atılmasıdır. Paketlerin atılması sıkışıklığı düşürmek için bir yoldur fakat yeterli değildir. Paketlerin atılması devamlı sıkışıklık için etkili değildir ve dahası TCP protokolü sıkışıklığa ek olarak kaybolan paketleri yeniden gönderir. Bundan dolayı yönlendiriciler, herhangi bir sıkışıklık durumunda son noktalara sinyal iletmelidir, böylece iletim penceresi düşer. Bu sistemlerin gerçekleştirilmesinde, bazı görüntüler de dikkate alınmalıdır. Birincisi patlama ve devamlı sıkışıklık arasında ayrım yapmaktır. Diğer görüntü ise kaynakların dağıtımında dürüst olmaktır. Sonuç olarak eş zamanlılıktan

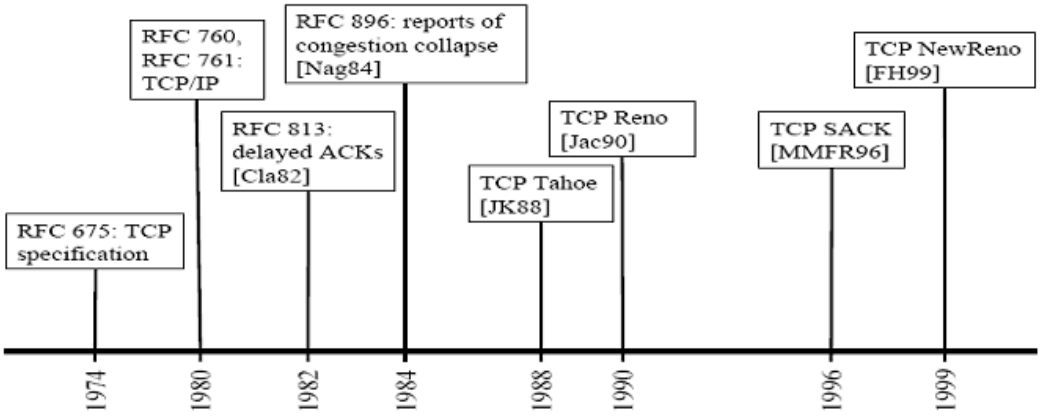
kaçınmak için, eğer yönlendirici bir zamanda aniden fazla paketi işaretlerse, tüm kaynakların oranı ve ağın performansı aynı anda dramatik bir şekilde düşer. Bu bölümde, ayrıca atılan paketlerin çeşitli ve daha etkili yollarla nasıl atıldığını göreceğiz.

Bir TCP bağlantısının akışı, paketlerin korunmuş bir şekilde iletilmesi ilkesine dayanır. Sıkışıklık kontrolü, bu kuralı bozan yerleri bulmaya ve düzeltmeye çalışır. Paketlerin korunmasında yaşanan başarısızlıkların sadece üç nedeni olabilir.

1. Bağlantı dengeli değildir.
2. Gönderici, henüz eski paket ayrılmadan yeni paketi göndermiş olabilir.
3. Bağlantı yolu boyunca kullanılan kaynak limitlerinden, denge sağlanmamış olabilir.

2.2 TCP'nin Gelişim Tarihi

TCP, ilk tanımlandığı 1974'den beri birçok değişim geçirmiştir. Şekil 2.1 TCP'deki anlamlı gelişmelerin zaman çizgisini göstermektedir.



Şekil 2.1. *TCP'nin gelişimi*

TCP sıkışıklık kontrolünün gelişimi için yönetim problemi 1980'lerin ortasında sıkışıklık çökmeleri (congestion collapse) ile ortaya çıkmıştır. Sıkışıklık çökmeleri yeni veri alınamadığı ve kayıp dataların yeniden iletimi ile ağın çok yüklenmesi durumunu tanımlar. TCP, bir paket gönderildiği zaman bir zamanlayıcı ayarlayarak kayıp parçaları tespit eder. Eğer paket alındığına dair ACK alınmadan önce zamanlayıcının süresi dolarsa,

paketin kaybolduğu varsayılır ve bu sıra numarasından başlayarak tüm paketler yeniden iletilirler (*Go-Back-N*). Bu sıkışıklık çökmesinden önce, TCP'de sadece akış kontrolü gerçekleştirilmişti. Ağ da sıkışma oluştuğunda ne yapılabileceği ile ilgili hiç birşey yoktu. Sıkışıklık çökmelerine cevap olarak, TCP için sıkışıklık kontrolü (congestion control) algoritması geliştirildi. Sıra taştığı zaman oluşan paket kaybındaki temel düşünce, sıkışıklığın işaretidir. Paket kayıpları tespit edildiği zaman, göndericiler, gönderme oranlarını düşürmelidirler.

2.3 TCP Tahoe

Sıkışıklık kontrolü için ilk olarak TCP Tahoe geliştirilmiştir ve esas TCP'den farklı olarak birçok değişiklik içermektedir. Yavaş başlama (slow-start) fazı, sıkışıklıkdan kaçınma (congestion avoidance) fazı ve hızlı tekrar iletim (fast retransmit) fazlarını içermektedir.

2.3.1 Yavaş Başlama

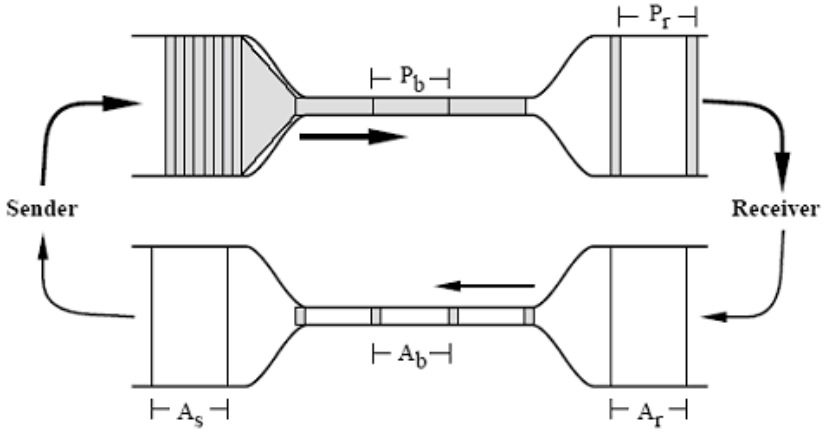
TCP, paketlerin muhafaza edilmesi düşüncesini takip eder. Bir paket ağı terk edinceye kadar yeni bir paket gönderilmez (ACK geri dönmeden). TCP Thaoe tanıtılmadan önce, başlangıç dışında paketlerin korunmasına uyulurdu. Başlangıçda, yeni parça göndermek için doğrulanmış bir paket yoktur, bundan dolayı gönderici dolu bir pencerenin değerini bir kere gönderir. Gönderici buna rağmen, ağın bir seferde ne kadar veri taşıyacağını bilemez, bundan dolayı sıklıkla olan patlamalar paketlerin yönlendiricilerde düşmesine izin verir.

Paket gönderen gönderici, paketlerin muhafaza edilmesi özelliğine bakmak ve ağa yeni bir paket göndermek için bir saat olarak ACK'leri kullanır. Bundan dolayı alıcı, paketlerin ağ üzerindeki alınmasından daha hızlı olmayacak şekilde ACK'ler oluşturur. Bu protokolün kendi kendine saatli denetimidir (Self - clocking) [1].

Kendi kendine saat denetimli sistemler, otomatik olarak bant genişliği ve gecikme çeşitlerini ayarlarlar ve geniş dinamik bir aralığa sahiptirler. (800 mbps Cray kanallardan, 1200 bps paket radyo linkine kadar) [1].

Saati (clock) başlatmak için bir "*yavaş-başlama*" algoritması geliştirilmiştir. Bu algoritmada iletilen datanın miktarı yavaş yavaş artmaktadır [1].

- Herbir bağlantı durumuna, bir sıkışıklık penceresi (cwnd) eklenir.
- Başladığında yada kayıp sonrası yeniden başladığında cwnd bire ayarlanır.
- Herbir ACK'de cwnd bir paket arttırılır.
- Gönderirken, alıcının bildirilen en küçük pencere ve cwnd'sini gönderilir.



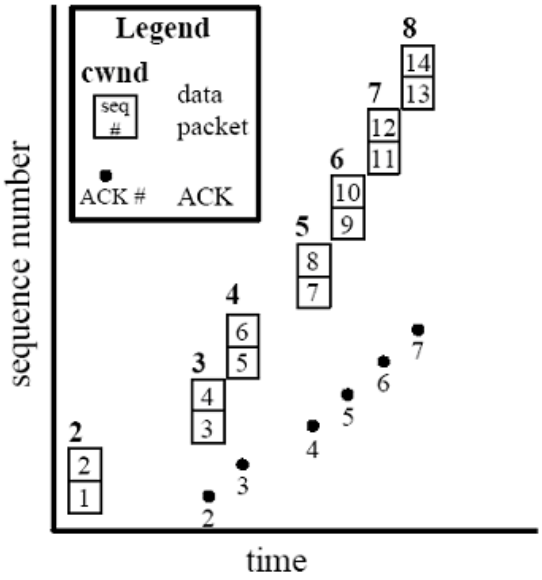
Şekil 2.2. *Pencere akış kontrolü "kendi kendine saatli denetim"*

Başlangıçda (ve paket düştükten sonra), paketlerin mümkün olduğunca hızlı bir şekilde gönderilmesi yerine, ACK'in alıcıdan iki kat oranında geri dönmesi için, yavaş başlama, ağa giren paketlerin oranını sınırlar. TCP Tahoe sıkışıklık penceresi (congestion window , cwnd) tanıtır. İlk başta bir pakete ayarlanır. Gönderilen pencere cwnd'nin en küçüğüne ve alıcının duyuru penceresine (advertised window) ayarlanır. Her ACK alındığında, cwnd bir paket arttırılır. Gönderilen pencerenin büyüklüğü, göndericinin gönderme oranını kontrol eder. (*oran = w / RTT*). Bu gönderme oranını arttırmak için, gönderici, cwnd penceresini arttırarak, gönderilen pencerenin değerini arttırabilir. Sıkışma

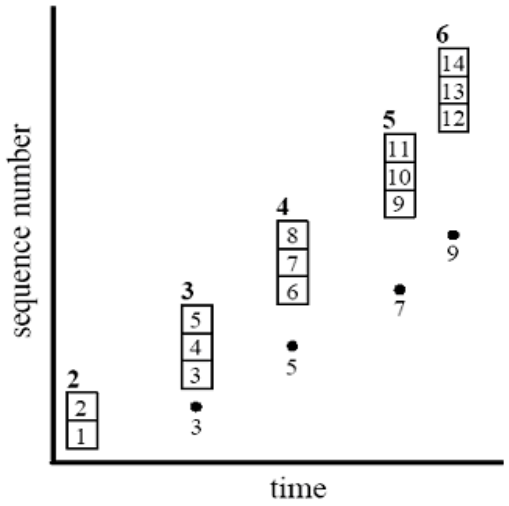
penceresinin ilk penceresindeki ilk paket doğrulanmadan önce, ilave dataların ağa transfer olmasına izin verir. Sıkışma penceresi sadece, alıcıdan datanın başarılı bir biçimde dağıtıldığını belirten ACK alındığında arttırabilir. Gönderici, ağ sıkışması tespit edilinceye kadar cwnd'yi arttırır.

Şekil 2.3 yavaş başlama işleminin bir örneğini göstermektedir. x ekseni zaman ve y eksenide sıra numarasıdır. Kutular paketin iletimini göstermektedir ve sıra numaraları ile etiketlenmişlerdir. Örneği tutmak için kutuların 1 byte'lık paketler olduğu varsayılır. Kutuların üstündeki numaralar, paketler gönderildiği zaman cwnd'nin değerini belirtir. Noktalar gelen ACK'yı belirtir ve en yüksek sıra numarasının y ekseni boyunca merkezlenir. ACK'lar aynı zamanda x ekseni boyunca, alınan ACK tarafından serbest bırakılan paketlerle merkezlenir. ACK'nın altındaki sayılar ACK'da taşınan sıra numarasını belirtir (sonraki paketin sıra numarası alıcı tarafından beklenir), ACK numarasına tekabül eder. Örneğin ilk nokta paket 1 in doğrulanmasıdır ve alıcı paket 2'yi almayı bekler. Bundan dolayı nokta y ekseninde konumlanmıştır, noktanın altındaki 2 ile paket 1 merkezlidir, bu da 2 nin ACK numarasının ACK ile taşındığını gösterir. Şekil 1.8'de cwnd'nin ilk değeri 2 dir, bundan dolayı 0. zamanda paket 1 ve 2 gönderilir. Paket 1 için ACK alındığında, cwnd 3'e arttırılır ve paket 3 ve 4 göndeilir – paket 3 gönderilir çünkü paket 1 doğrulanmıştır, paket 4 de sıkışma penceresini doldurmak için gönderilir. Bundan dolayı, yavaş başlama sırasında, herbir ACK alındığında, iki yeni paket gönderilir.

Yavaş başlamada, göndericinin sıkışma penceresi her bir ACK alındığında bir parça arttırılır. Eğer bir ACK, iki parçanın alındısını doğrularsa (geciken ACK da), ACK'nın alındısı tarafından iki parça bırakılır. Bir gönderici, bir alıcı ile iletişim kurar ve geciken ack'ları kullanmazsa, şekil 2.3 deki gibi herbir ACK alındığı zaman iki parça gönderilir (bir tanesi doğrulanan parça için ve bir taneside artan cwnd için gönderilir). Bir gönderici, bir alıcı ile iletişim kurar ve geciken ACK'ları kullanırsa, her bir ACK alındığında üç parça gönderir (iki tanesi doğrulanan ACK için bir taneside artan cwnd için). Bu şekil 2.4'de gösterilmiştir.



Şekil 2.3. *Yavaş başlama*



Şekil 2.4. *Geciken ack'larla yavaş başlama*

2.4 Gidiş – Dönüş Zamanı (RTT)

Yavaş başlama ile paket kaybetmediğimizi varsayarsak, belli bir zamanda sistem bir iletim penceresi tarafından belirtilen dengeye ulaşır. Bu noktada, ACK alınmazsa, sistemde bir başarısızlık olur, gönderici paketi yeniden gönderir. İletim ve ACK'in alınması arasındaki zamanı göz önüne almamız gerekir ki bu zaman *gidiş dönüş zamanıdır (RTT)* . RTT'nin temelinde, iletim ve ACK arasında geçen zamanın kaydını tutabilmek için gönderici bir zamanlayıcı tanımlar (retransmission timeout, RTO). Eğer

zaman, tanımlanan zamanı geçerse paket yeniden gönderilir. Zamanlayıcı için iyi bir zaman seçmek çok önemlidir, aksi takdirde eğer seçilen değer çok düşükse, yeniden gönderme çok fazla olur, ama paketler kaybolmaz. Eğer çok büyük olursa, bağlantının hızı düşer. TCP de, *M* RTT'nin tahmini değeridir ve paketin son RTT değeri ve önceki *M* değeri ile birlikte hesaplanır.

$$yeniM = a*eskiM+(1-a)*RTT \qquad (2.1)$$

Eşitlik (2.1)'de belirtilen *a* için 0.9 iyi bir değerdir. RTO seçilmiş olmalıdır, örneğin $\beta = 2$ *için* $\beta*M$ değeri. Bir RTO içerisinde bir ACK alınmadığı zaman, paketin kaybedildiği düşünülür. Eşik değere ulaşıldığında sıkışıklıkdan kaçınma başlar, pencere üstel olarak artma yerine doğrusal olarak artar [2].

2.5 Sıkışıklıkdan Kaçınma

Paketlerin kaybolmasının iki nedeni vardır, iletişim sırasında zarar görmeleri veya iletim yolunda bulunan bazı yerlerdeki tampon kapasitesinin yetersizliğinden oluşan ağ sıkışıklarıdır. Ağ yolunda paketlerin zarara uğramasından dolayı olan kayıplar çok azdır (<< %1), paketlerin kaybolmasının en büyük nedeni ağdaki sıkışıklıktır [1].

Sıkışıklıktan kaçınma stratijisinin iki bileşeni vardır; ağ sıkışmanın olduğu son noktaya sinyali iletmek zorundadır ve son noktalarında sinyal alındığı zaman kullanımı azaltacağı, sinyal alınmadığı zaman kullanımı arttıracağı bir politikası olmalıdır [1].

Eğer paketler her zaman sıkışmadan dolayı kayboluyorsa ve zaman aşımıda genellikle paketlerin bu şekilde kaybolmasından oluşuyorsa, "sıkışık ağ" sinyali için de iyi bir adayımız vardır. Özellikle özel bir değişiklik (paket başlığına yeni bir bit eklenmeden) olmadan, bu sinyal tüm varolan ağlar tarafından otomatik olarak dağıtılır [1].

Diğer bir sıkışıklıkdan kaçınma stratejisi, son düğüm hareketidir. Doğrudan birinci-derece zaman-serisi ağ modeli tarafından izlenir. Bu model, ağ yükünün, bazı uygun uzunlukların sınırlandırılmış aralıkları üzerindeki ortalama sıra uzunluğu ile ölçüldüğünü söyler. *i* aralığındaki yük L_i ise, L_i örnek alınan zamanla karşılaştırıldığında değiştini söyleyen, sıkışık olmayan bir ağ modellenir [1].

$$L_i = N \qquad (2.2)$$

N sabittir. Eğer ağda sıkışıklık söz konusu olursa, bu sıfırıncı dereceden model çalışmaz. Ortalama sıra uzunluğu iki terimin toplamı olur, yukarıdaki ***N*** esas gecikme ve yeni ağın ortalama gelişi olarak kabul edilir, yeni terimde son zaman aralığından ayrılan ve bu trafiğin etkisinin bir bölümü olarak kabul edilir. Örneğin irkilmiş yeni iletim,

$$L_i = N + \gamma L_{i-1} \qquad (2.3)$$

şeklinde tanımlanır [1]. Bunlar ***L(t)*** nin Taylor serisi açılımdaki ilk iki terimidir.

Ağ sıkıştığı zaman, γ genişlemek ve sıra uzunluğu üstel olarak artmaya başlamalıdır. Eğer trafik kaynakları en az sıranın artması kadar çabuk büyürse sistem kararlı olabilir. Bir kaynak yükü pencere – tabanlı bir protokolde, pencere büyüklüğünün (*W*) ayarlanması ile kontrol eder [1]. Sıkışıklık durumunda,

$$W_i = dW_{i-1} \qquad (d < 1) \qquad (2.4)$$

şeklinde tanımlanır.

Eğer sıkışma yoksa, γ sıfıra yakın olmak zorundadır ve yük hemen hemen sabittir. Ağ , talep fazla olduğunda, kaybolan bir paket yoluyla, bu durumu bildirir. Eğer bağlantı, paylaşımdan daha az kullanıyorsa hiç bir şey söylemez. Bundan dolayı bir bağlantı o anda ki limiti bulmak için bant genişliğinden daha fazla faydalanmalıdır. Örneğin, aynı ağ yolunu başka birisi ile paylaşıyorsunuz ve bir noktada birleştiniz ve

varolan bant genişliği yarı yarıya paylaştırıldı. Daha sonra karşı taraftaki kişi bağlantısını kapattı. Eğer siz pencere büyüklüğünüzü arttırmazsanız bant genişliğinin %50' si boş yere kullanılmamış olacaktır. Burada nasıl bir arttırma politikası izlenmelidir ?

En iyi arttırma politikası, pencere ölçüsünde küçük, sabit değişiklikler yapılarak bulunmuştur. Sıkışıklık yokken ,

$$W_i = W_{i-1} + u \qquad (u << W_{max}) \tag{2.5}$$

şeklinde tanımlanır ve W_{max} hattın bant genişliğidir.

Önceki, sıkışma kontrol algoritması sesi müthişdir. Ama bu değildir. Yavaş başlama gibi, kodun üç satırıdr.

- Herhangi bir zaman aşımında şimdiki cwnd yarıya düşürülür (çarpımsal düşüş).
- Yeni bir veri için gelen her bir ACK'de cwnd, *1/cwnd* kadar arttırılır (katkılı artış).
- Gönderirken, alıcının bildirdiği en küçük pencere ve cwnd gönderilir.

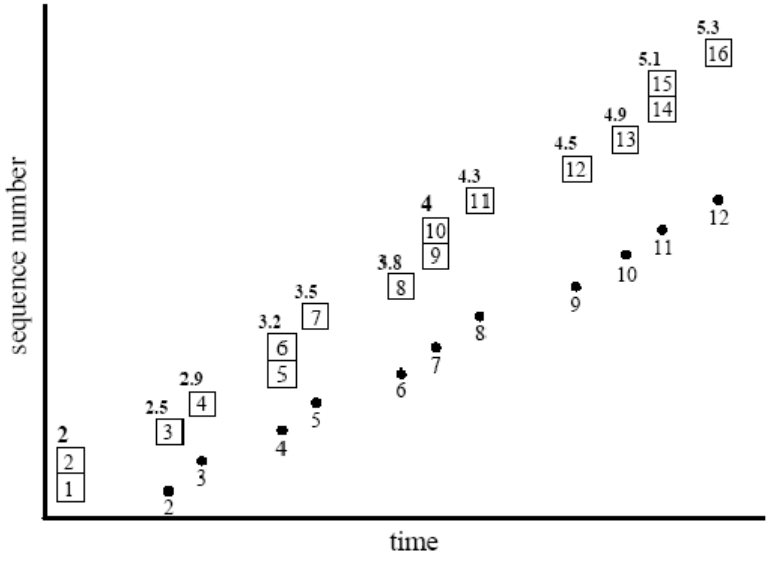
Bu algoritma sadece sıkışıklıkdan kaçınma içindir. Daha önceden tanımlanan yavaş başlamayı içermez. Paket kaybolmasında, sinyal sıkışmasında yeniden başlatılır. Bu yukarıdakilere ek olarak kesinlikle yavaş başlamaya ihtiyaç vardır. Çünkü sıkışıklıkdan kaçınma ve yavaş başlama bir zaman aşımı ile tetiklenir ve her ikiside sıkışıklık penceresini (cwnd) etkiler. Bunlar sıklıkla karıştırılır. Aslında tamamen farklı ve bağımsız algoritmalardır [1].

TCP'deki sıkışıklık kontrolünün amacı ağ sıkışmasına tepki verirken varolan tüm kaynakların kullanılmasını sağlamaktır (örneğin yönlendiricilerin sıralarındaki tampon bellek boşlukları). Varolan bant genişliğinin miktarı bilinen bir miktar değildir, akışların girip ayrılmasına göre değişir. TCP, varolan ek bant genişliklerini, sıkışma tespit edilinceye kadar gönderme oranını düzenli olarak arttırmak için araştırır. Ağda oluşan sıkışmanın tek bildirisi paketlerin kaybedilmesidir. Düşürülen paketler dışardan

bilinemez, yönlendirici gönderir, böylece gönderici paketin kaybolduğu ve hangi paketin kaybolduğunun sonucunu çıkarır [3].

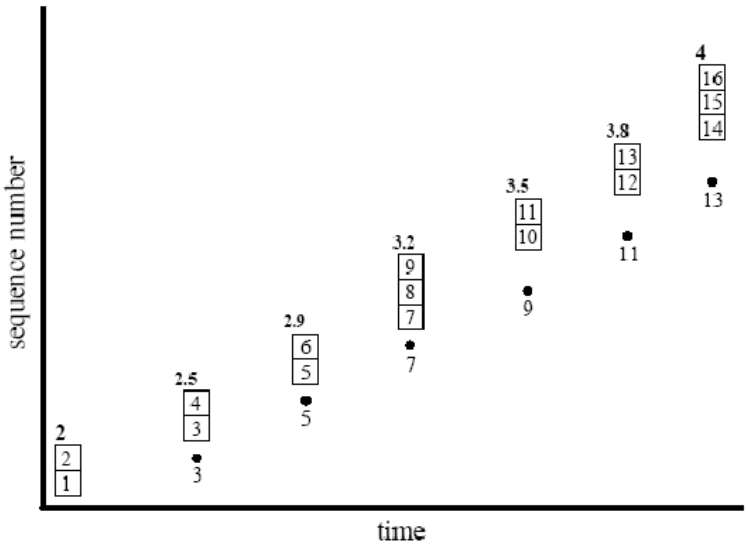
Sıkışmadan kaçınmada, doğrusal bir şekilde artan iletim oranı ek bant genişliklerini sıkı bir şekilde arar. TCP Tahoe'de, gönderici yavaş başlama ile başlar ve sonradan sıkışıklıkdan kaçınmaya geçiş yapar. TCP Tahoe, *ssthresh* isimli, yavaş başlamadan sıkışıklıkdan kaçınmaya geçiş için eşik değerini belirten bir değişken ekler. *cwnd* >*ssthresh* olduğunda, yavaş başlama sonlanır ve sıkışıklıkdan kaçınma başlar. Buradaki düşüncede her bir RTT'de, *cwnd* büyüklüğündeki bir pencere ile TCP göndericisi en fazla *cwnd* alır, bu sonuçlardan bir sıkışma penceresi, her RTT de en fazla bir paket artar. Bu doğrusal artış, her RTT'de iki katı artan yavaş başlama ile çelişir [3].

Şekil 2.5 sıkışıklıkdan kaçınma işlemini göstermektedir. Tekrar, *cwnd*' nin ilk değeri 2 dir, bundan dolayı iki paket iletişimi başlatmak için gönderilir. Burada *ssthresh*'ın ilk değeri de 2'dir. *cwnd* >= *ssthresh* olduğundan, sıkışıklıkdan kaçınma etkindir. Paket 1 için ACK alındığı zaman, cwnd , 2.5 olur (2+1/2) ve paket 1 ağı terkettiğinden bir paket gönderilir. paket 2 için ACK alındığında, *cwnd* 2.9 olur (2,5 + 1 / 2.5). Tekrar, ağda göze çarpan sadece iki paket tutmak için, bir paket gönderilir. paket 3 için ACK alındığı zaman, cwnd 3'den büyük olur, böylece iki paket gönderilebilir [3].



Şekil 2.5. *Sıkışıklıktan kaçınma*

Şekil 2.5'la karşılaştırıldığında, şekil 2.6, geciken ACK kullanıldığındaki sıkışıklıktan kaçınma operasyonunu gösterir [3].



Şekil 2.6. *Geciken ack'lar ile sıkışıklıkdan kaçınma.*

2.5.1 Yönlendiricilerde Sıkışıklıkdan Kaçınma

Son nokta gibi, yönlendiriciler de sıkışıklıktan kaçınmanın önemli bir parçasıdır. Genellikle yönlendiricilerin, gelen paketlerin tutulduğu sınırlı tampon kapasiteleri vardır ve bu tampon dolduğu zaman sıkışma oluşur. Tampon dolduğu zaman, yükü azaltmak için tek yol paketlerin atılmasıdır. Paketlerin atılması sıkışıklığı düşürmek için bir yoldur fakat yeterli değildir. Paketlerin atılması devamlı sıkışıklık için etkili değildir ve dahası TCP protokolü sıkışıklığa ek olarak kaybolan paketleri yeniden gönderir. Bundan dolayı yönlendiriciler, herhangi bir sıkışıklık durumunda son noktalara sinyal iletmelidir, böylece iletim penceresi düşer. Bu sistemlerin gerçekleştirilmesinde, bazı görüntüler hesap içine alınmalıdır. Birincisi patlama ve devamlı sıkışıklık arasında ayrım yapmaktır. Diğer görüntü ise kaynakların dağıtımında dürüst olmaktır. Sonuç olarak eş zamanlılıktan kaçınmak için eğer yönlendirici bir zamanda aniden fazla paketi işaretlerse, tüm kaynakların oranı ve ağın performansı aynı anda dramatik bir şekilde düşer.

2.5.1.1 Rastgele Erken Düşürme

Bu sistemin temeli, sıranın uzunluğu belli bir eşik değerini geçtiği zaman, paketlerin rastgele atılması temeline dayanır. Bu algoritma patlamış trafikleri ayıramaz. Basitçe görüldüğü gibi, bu algoritma eş zamanlılığa da liderlik eder, çünkü, aniden sıra uzunluğu eşik değerine ulaştığı zaman herhangi bir dikkat gözetilmeksizin tüm paketler düşürülür. Eş zamanlılıkdan kaçınmak için paketlerin atılması işlemi kademeli olarak, az ile başlanıp sonra arttırarak yapılmalıdır.

2.5.1.2 Rastgele Erken Tespit Etme (Random Early Detection - RED)

Sıkışmadan korunmak için en ilginç algoritmadır. Paketlerin düşürülmesi yolunda erken düşürmeye benzer, eşik değerinin üzerine çıkıldığı zaman biraz daha karmaşıklaşır [2].

Tanımlanan bazı değişkenler; *avg*, sıranın ortalama uzunluğu, *minimum* ve *maksimum* eşik değerleri ve *pa*, paketlerin düşürülme olasılığıdır. Bu algoritma şu şekilde çalışır [4],

- Her gelen paket için *avg* hesaplanır.
- *pa* hesaplanır
- *minimum* < *avg* < *maksimum* ise paket *pa* olasılığı ile atılır.
- Eğer *avg maksimum* dan büyükse paket atılır

Bazı görüntüler bu algoritma ile ilgilidir. Ortalama uzunluk, önceki değerlerin formülde hesaplanması ile bulunur [4].

$$yeniAvg= (1-w)eskiAvg+w*q \qquad (2.7)$$

q, o andaki sıra ölçüsü, *w* de bir ağırlık faktörüdür. *w*'nın seçimi önemlidir çünkü eğer çok küçük olursa *avg* de değişimler çok yavaş olur ve sıkışmalar tespit edilemez. Eğer çok büyük olursa *avg* de değişimler çok hızlı olur ve patlayan trafikler (bursty traffic) sıkışma gibi düşünülür. Ortalama sıra uzunluğunun kullanılmasıyla, patlayan trafiklerin sıkışma

gibi tespit edilmesi önlenmiş olur. *avg* nin hesaplanmasında periyod da hesaba katılır, bu periyod da sıra boş olduğu zaman belli sayıda bir paketin iletildiği varsayılır [2].

pa sabit bir değer değildir, *avg* nin temel hesabında ve daha önceki *pa* değerleri kullanılır ve sıra uzunluğu arttıkça yavaş yavaş artar. Bu eş zamanlama dan kaçınmak için kullanışlı bir yöntemdir, sıkışıklığın düştüğü kadar uzar, az sayıda paket atılır; sıkışıklık arttığı zaman paketlerin düşme olasılığı daha fazladır. Sonuç olarak sıkışıklık çok yüksekse tüm paketler atılır [2].

Minimum ve *maksimum* seçimi tampon uzunluğuna ve elbetteki gelen trafiğin türüne bağlıdır. *Minimum* ve *maksimum* arasındaki fark eş zamanlılıktan kaçınmak için yeterince büyük seçilmelidir. Çünkü, eğer çok küçükse sistem tüm paketleri, sıkışma yoğun olmadan, erkenden atmaya başlar. Kural olarak önerilen *maksimum = 2*minimum* dır [4]. Ayrıca *minimum* yeterince geniş seçilmelidir ki bant genişliğinin kullanılması maksimuma çıkarılabilsin [2].

Sonunda, dikkat edersek RED dahili olarak yüksek yüklü bağlantıları ve düşük yüklü bağlantıları ayırmaktadır, aslında belli bir olasılıkla paketlerin atılmasıyla ağır yüklü bağlantılar için paketleri atanların sayısı hafif yüklü bağlantılarınkinden daha geniştir. Bu da kaynakları daha doğru ayırmamız konusunda yol gösterir [2].

2.6 Hızlı Tekrar İletim

Parçaların kaybolduğu yada yeniden gönderilmesini gerektiren durumların tespiti için TCP bir zamanlayıcı kullanır. Bu tekrar iletim zaman aşımı (RTO) zamanlayıcısı her zaman bir veri paketi gönderilecek diye ayarlanmıştır (eğer zamanlayıcı henüz ayarlanmamışsa). Yeni bir data için ACK alındığında RTO baştan başlar. Eğer yeni bir kayıt için beklenen ACK'den önce RTO'nun süresi dolarsa, en eski doğrulanmamış paketin kaybolduğu ve yeniden iletileceği düşünülür. RTO zamanlayıcısı RTT'nin 3-4 katına ayarlanır, böylece geciken paketler gereksiz yeniden iletimlere sebep olmazlar.

TCP Tahoe'de sıkışıklıkdan kaçınma eki ile RTO'nun süresi dolarsa, 1/2 *cwnd, sstresh* da saklanır ve *cwnd* 1 parçaya ayarlanır. Bu noktada *cwnd* < *sstresh,* bu bir zaman aşımı ile yavaş başlamaya dönmektir [3].

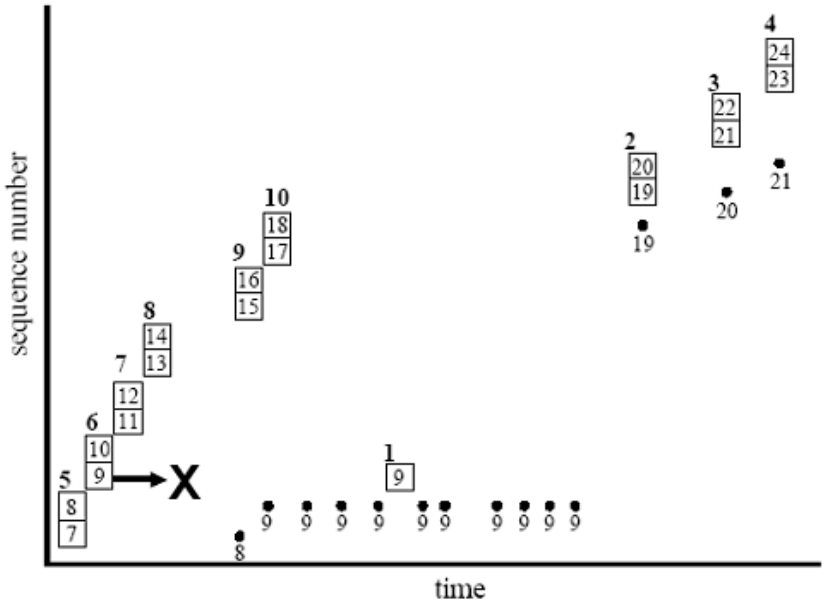
TCP Tahoe, paket kayıplarını tespit edebilmek için daha hızlı bir yol ekler ve bu *hızlı tekrar iletim (fast retransmit)* olarak adlandırılır. TCP Tahoe'den önce, parça kayıplarını tespit etmenin tek yolu RTO'nun süresinin dolmasıydı. Bir alıcı sıralı olmayan bir paket görür görmez, alınan son sıralı paket için bir ACK gönderir. Gönderici, sadece paketlerin yeniden sıralanacağından çok kayıp paket olduğunu ifade etmek için aynı ACK'nın üç tane tekrarlanan alındısını kullanır [3].

Şekil 2.7 TCP Tahoe hızlı tekrar iletimi göstermektedir. Bu şekil, şekil 2.3'ün devamıdır ve paket 9'un kaybolduğundaki durumu göstermektedir. Paket 7 için ACK geri döndüğünde, paket 15 ve 16 gönderilir ve *cwnd* 9'a çıkar. Paket 8 için ACK alınırsa, *cwnd* 10'a çıkar ve paket 17 ve 18 gönderilir. Ama paket 9 'un ACK sı alınmazsa, paket 10'un ACK'sı paket 8 in tekrarlanan ACK'sıdır. Sıkışma penceresi, tekrarlanan ACK'lar alınınca, değişmez. Tekrarlanan ACK'lar geri dönmeye devam eder. Üçüncü tekrarlanan alındığında, hızlı tekrar iletim girilir. Paket 9 kaybolduğu varsayılır ve hemen gönderilir. Bu noktada, *cwnd ,* 1 pakete düşürülür. Tekrarlanan ACK'lar alınmaya devam eder ama gönderilen data paketi olmadığından *cwnd* de değişiklik yapılmaz. Kaybolan paketin başarılı bir şekilde alındığına dair ACK gelirse, *cwnd* 2'ye çıkarılır ve paket 19 ve 20 gönderilir. Bu noktadan yavaş başlama normal olarak başlar [3].

Hızlı tekrar iletimi kullanmanın avantajı, kaybolan parçanın tespiti için ihtiyaç duyulan zamanı düşürmesidir. Hızlı tekrar iletim olmadan, kayıpları tespit etmek için RTO'nun süresinin dolması gerekmektedir. Akışlar için geniş sıkışma pencerelerine sahip olmak, üç tane tekrarlanan ACK'nın tetiklemesiyle, çoklu ACK'ların bir RTT içerisinde tipik olarak alınmasıdır. Kayıp bir parçanın tespit edilmesi bir RTT'den daha az süredir. Bu yolda, hızlı tekrar iletim, veri gönderilemediği sırada uzun zaman aşımlarından kaçınmaya izin verir [3].

2.7 AIMD

TCP Tahoe'nin TCP'ye en büyük katkısı AIMD (additive Increase / Multiplecative Decrease) pencere düzeltme algoritmasıdır. Eklemeli artış $w(t+1) = \alpha + w(t)$ olarak tanımlanır. $w(t)$, bir RTT zaman biriminde t zamanında parçalarda o andaki sıkışma penceresinin ölçüsüdür. Sıkışıklıkdan kaçınmada, $\alpha = 1$ *dir*. Her ACK alındığında, sıkışma penceresi $1 / w(t)$ kadar arttırılır, sonuçda bir RTT'de en fazla bir paket arttırabilir. Çarpımsal düşüş ise $w(t+1) = \beta w(t)$ olarak tanımlanır. TCP Tahoe'de, $\beta = 0.5$'dir, *ssthresh* de geri iletim sırasında ½ *cwnd* ye ayarlanır [3].



Şekil 2.7. *TCP Tahoe hızlı tekrar iletim*

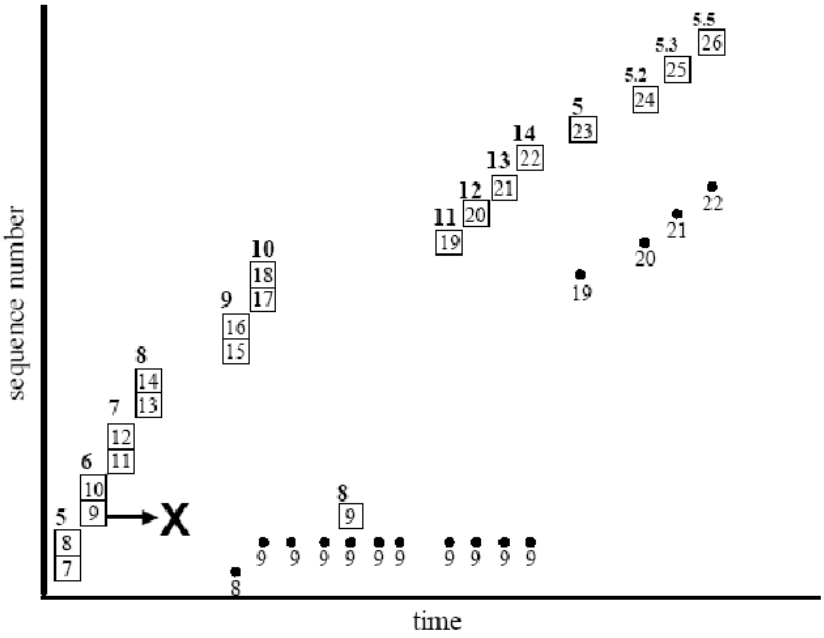
2.8 TCP Reno

1990 yılında Van Jacobsen TCP Tahoe'ye hızlı düzeltme (fast recovery) adı verilen yeni bir özellik ekledi. Hızlı düzeltme ile TCP Tahoe, TCP Reno olarak bilinir. Bu aslında Internet üzerinde TCP'nin standardı olarak kabul edilir [44].

2.8.1 Hızlı Düzeltme (Fast Recovery)

TCP Tahoe'de olduğu gibi, eğer parçanın tekrarlanan ACK'i üç kez alındıysa, o parçanın kaybolduğu varsayılır. Parçanın kaybolduğu üç kez tekrarlanan ACK yoluyla tespit edilirse, yavaş başlama yerine hızlı düzeltme (fast recovery) devreye girer. Hızlı düzeltmede, *ssthresh,* 1/2 *cwnd* ye ayarlanır (AIMD algoritmasına göre), ve *cwnd* ise, *ssthresh* + 3 olur. Her bir ek tekrarlı ACK alındığında, *cwnd,* yavaş başlamadaki gibi bir parça arttırılır. Yeni parçalar *cwnd* izin verdiği sürece gönderilebilir. Geri iletilen parça için ACK alındığında, *cwnd* tekrar *ssthresh'*e ayarlanır. Kayıp parça bir kez doğrulandığında, TCP hızlı düzeltmeden ayrılır ve sıkışmadan kaçınmaya (congestion avoidance) geri döner. Eğer *cwnd* önceden genişse, *cwnd* yarıya bölünür ve *cwnd* 1 parçaya düşürülmek yerine sıkışıklıkdan kaçınmaya geçilir ve yavaş başlamaya geri dönülür, göndericiye varolan bant genişliğini sıkıca araştırması için izin verir ve *cwnd* nin önceki değerine yaklaşma şansı ağ sıralarındaki taşmanın şansı, yavaş başlama kullanmakdan daha azdır.

Şekil 2.8 hızlı tekrar iletim ve hızlı düzeltmeleri göstermektedir. Bu durum şekil 2.7'ye çok benzemektedir. Parça 9 kaybolur, hızlı tekrar iletim de belirtildiği gibi yeniden iletilir. Hızlı düzeltmeye göre, *cwnd* 1'e düşürülmüyor ama 8'e değiştiriliyor (cwnd yarısına yani 5'e düşer ama sonra herbir tekrarlı ACK alındığında , 3 arttırılır). Hızlı tekrar iletimden sonra iki yada daha fazla tekrarlı ACK alınırsa, bunların herbirisi için, *cwnd* arttırılır. Bundan dolayı *cwnd* 10'dur ama gönderilecek ilave parça yoktur, çünkü TCP göndericisi, gönderilecek parçayı 10 olarak bilmektedir. Parça 15 alındığında, alındısı olarak ACK gönderildiğinde, *cwnd* tekrar arttırılır. Bundan dolayı *cwnd* şimdi 11'dir, daha fazla parça göndermek için boş yer yoktur ve parça 19 bırakılır. Bu parça 9'un geri iletilmesi doğrulanıncaya kadar devam eder. Bu olursa *cwnd* 5'e geri döner (kayıp tespit edildiğinde sıkışma penceresinin yarısı) ve sıkışıklıkdan kaçınma girilir ve normal olarak işletilir.



Şekil 2.8. *TCP Reno hızlı tekrar iletim ve hızlı düzeltme*

Yukarıdaki örnekte görüldüğü gibi, hızlı düzeltme ayrıca yavaş başlamadan, sıkışıklıkdan kaçınmaya bir geçis sağlar. Eğer bir gönderici yavaş başlamadaysa ve üç tekrarlı ACK ile kayıp parça tespit ederse, kayıp yerine koyulduktan sonra, sıkışmadan kaçınmaya girilir. TCP Tahoe'deki gibi, sıkışmadan kaçınma ayrıca *cwnd* > *ssthresh* olduğunda girilir. Bir çok durumda, *ssthresh'in* ilk değeri geniş bir değere ayarlanmasına rağmen parça kaybı genellikle sadece sıkışıklıkdan kaçınmaya geçisi tetikler [45].

TCP Reno, tek bir hızlı iletim ve hızlı düzeltme fazları boyunca sadece bir parçanın düzeltilmesi ile sınırlıdır. Aynı pencerede ek bir parça kaybı parçalar yeniden iletilmeden önce RTO'nun zamanının dolmasını gerektirebilir. Bu istisna, girilen hızlı düzeltme üzerinde *cwnd* 10 parçadan büyük olduğunda , göndericinin bir zaman aşımı tecrübesi olmadan düzeltmek için iki parçanın kaybına izin verir. Hızlı düzeltme sırasında, gönderilmiş olan yeni parçalardan birisi ilk hızlı düzeltme tamamlandıktan sonra kaybolabilir ve üç tekrarlı ACK ile tespit edilmiş olabilir. Bu durumda TCP Reno, yerine geçmede (succession) iki kez hızlı düzeltmenin girilmesiyle kayıp iki parçadan düzeltilebilir. Bu *cwnd* nin etkin bir şekilde iki RTO'da azaltılmasına sebep olur.

2.9 Seçici Alındı Bilgileri

TCP yürütme standartlarına son eklenenlerden biri seçici alındı bilgileridir (SACK). SACK, kayıpları kurtarmak ve göndericiye, pencerede çok sayıda kayıpların olması durumunda kısımların düşürülmesinde yardımcı olmak içindir. SACK opsiyonu, son zamanlarda alınan verilerin bitişik bloklarının özelleştirilmesini yapan 4'e kadar çıkabilen (yada RFC 1323 zaman sayacı opsiyonu kullanılırsa 3) SACK blokları içerir. Her bir SACK bloğu alıcıların tuttuğu verinin dizisini sınırlandıran iki sıra numaradan oluşmaktadır. Alıcı ACK opsiyonuna SACK opsiyonunu ekleyebilir. O da SACK'i seçilebilir hale getiren göndericiye tekrar geri gönderir. SACK bloklarındaki bilgiyi kullanarak gönderici hangi kısımların kaybolduğunu ifade edebilir (kaybolan kısımlardan 3'e kadarından bitişik olmayan blokları). SACK opsiyonu belirli bir işleyiş sırasının dışındaki kısımlar alındıktan sonra oluşan ACK üzerinde gönderilir. Bir SACK opsiyonuna 3 SACK bloğuna izin vermek, herbir SACK bloğunun en az 3 ACK içinde iletilmesini sağlar. Bu da ACK'ın yüzünde dinçliğin kaybolmasına sebep olur. SACK RFC sadece SACK opsiyonunu özelleştirir ve SACK opsiyonunda verilen bilginin gönderici tarafından nasıl kullanması gerektiğine karışmaz. Etkili veri kurtarma [42] için SACK opsiyonunu kullanarak bir metod sunulmuştur. Bu metod Fall ve Floyd [43] tarafından SACK'in yürütülmesi üzerine kuruludur. SACK kurtarma algoritması sadece 3 çift ACK alıcısından girilerek sadece bir kere hızlı bir şekilde işletir. SACK kullanımı gönderdiği kısımdan tekrar ilettiğinde TCP'ye bildirileri çözmesi için izin verir. Bunu yapmak için SACK TCP'ye iki değişken ekler, skor tahtası (göndermesi gereken kısımlar) ve hat bant genişliği (kısımları göndereceği zaman).

Skor tahtası, SACK tabanlı bilginin hangi kısımlarının kaybedildiğini kayıt eder. Skor tahtasındaki kısımlar en yüksek toplam değeri geçen bütün sıradaki numaralara sahiptir. Hızlı kurtarma boyunca, borudaki veri'nin büyük bir çoğunluğu onaylanmamıştır. Her seferinde kısım gönderilir, boru artırılır. Çift ACK, yeni bir verinin alındığını söyleyen SACK bloğu ile birlikte yerine ulaşır ulaşmaz borunun değeri azaltılır. $cwnd - boru \geq 1$ olduğu durumda, gönderici hem tekrar ileti gönderir hemde yeni bir veri iletir. Göndericiye veri göndermesi için verildiği zaman, ilk önce skor tahtasına bakar ve

alıcıdaki boşlukların doldurulmasında gerekli olan kısımları gönderir. Gerekli olan kısımlar yoksa gönderici yeni veri iletimi yapar. Hızlı kurtarmanın başındaki onaylanmamış veriler onaylandığı zaman gönderici hızlı kurtarmayı bırakır [3].

2.10 TCP NewReno

Hızlı kurtarma süresince, TCP Reno zaman aşımına uğramadan sadece kaybolan bir kısımdan kurtarma işlemi yapar. Çift ACK tekrar oluştuğu sürece, gönderici ağ'a yeni kısımlar gönderir fakat hızlı kurtarma kaybolan kısım için ACK alınana kadar olmaz. Sadece yeniden ileti herbir kurtarma periyodu boyunca gönderilir. RTO süre ölçerin süresinin dolmasıyla çok sayıda yeni ileti tetiklenmiş olur. TCP NewReno, göndericinin bir kısım ACK [46,47] alındıktan sonra hızlı kurtarmaya devam ettiği yerde, non-SACK-enabled TCP Reno olarak değişir. Bir kısım ACK alındı bilgileri, kısımlar kaybolmadan tespit edilir. Bir kısım ACK alındısı ile, gönderici alıcının beklediği diğer bir kısmın kaybolduğunu ifade eder. TCP NewReno tek bir hızlı kurtarma boyunca göndericinin birden fazla kısmı göndermesine izin verir. Fakat sadece bir kayıp kısım her bir RTT tarafından yeniden iletilebilir.

Yakın zamanda yapılmış bir çalışma, TCP NewReno'nun Internet Web sunucuları [48] örneği için en popüler TCP versiyonun olduğunu belirtmektedir.

2.11 TCP Eşleştirmesi

TCP Reno'daki ağın tıkanıklığında tek gösterge TCP'nin en sık uygulaması olan kısım kaybıdır. TCP host'larında kayıp kısımlar açıkca yönlendiriciler tarafından belirtilmezler. Fakat, kayıpları belirtmek için zaman aşımına ve çift alındı bilgisine güvenmelidir. TCP tıkanıklık kontrolündeki tek problem, kısım kaybı olduktan sonra gönderme oranını düşürmesidir. Araştırma mekanizması olan *cwnd*'yi veri kaybı

oluşuncaya kadar artırmakla, TCP ek bant aralığı ararken sıranın taşmasına sebep olabilir. Ek olarak, TCP'nin sıkışıklık sinyali çift koddur. Hem ACK döner ve TCP *cwnd*'yi artırır hemde kısım kaybı tespit edilir ve *cwnd* büyük ölçüde azaltılır. Böylece TCP tıkanıklık kontrolü veri kurtarma için kendi mekanizmasına bağlanır. İdeal sıkışıklık kontrol algoritması sıkışıklığı tespit edebilmelidir ve kısım kaybı olmadan reaksiyon vermelidir.

Yönlendirici tampon bellekleri taşmadan önce tıkanıklığı tespit etmek için iki ana yaklaşım vardır, uç uca metodu ve yönlendirici tabanlı mekanizmayı kullanmak. Aktif sıra yönetimi(AQM) gibi yönlendirici tabanlı mekanizmalar, drop-tail yönlendiriciler problemdir fikriyle ortaya çıkmıştır. Bu mekanizmalar paketler düşürülmeden önce sıkışıklık olduğunda yönlendiricilere değişiklik yaparlar ve böylece göndericileri uyarırlar. Uç uca yaklaşımlar drop-tail sıralama mekanizmasından daha çok TCP Reno'da değişiklik yapmak üzerine yoğunlaşmışlardır. Bu tür bir çok yaklaşım uç uca ölçümleri kullanıp ağ'ı görüntüleyerek TCP Reno'dan daha önce tıkanıklığı tespit etmeye ve reaksiyon vermeye çalışır (örneğin, kısım kaybolması olmadan önce). Teorik olarak sıkışık yönlendiriciler, sıkışıklık olduğu zaman sıkışıklığı anlamak için en iyi pozisyonda olduklarından, AQM en iyi performansı vermektedir. Karmaşıklık içeren AQM metodları kullanmak ve ağda yönlendiricileri değiştirmeye ihtiyaç duymak sakıncalardır. Yaklaşımım yönlendirici tabanlı mekanizmalara performans kolaylığı sağlamaya çalışan uç uca metodlarına bakmaktır.

Sıkışıklıkları erkenden tespit ederek ve kısım kayıplarından kaçınarak akışın tek-yol iletim sayısı bilgisi TCP sıkışıklık kontrolünde kullanılabilir. Bağlantının ileri doğru aldığı yolda OTT göndericiden alıcıya bütün bağlantıları döndüren ve yayılma ve dizi gecikmesi olan bir zamandır. Diziler taşmadan önce yönlendiriciler de toplanırlar, artan OTT ile sonuçlanırlar. Bütün göndericiler OTT'lerdeki değişiklikleri doğrudan ölçerlerse ve OTT sıkışıklığın oluştuğunu belirttiğinde düşerse, tıkanıklık hafifleyebilir.

BÖLÜM 3

AKTİF SIRA YÖNETİMİ

Aktif sıra yönetimi *(Active Queue Management - AQM),* bir yönlendirici tabanlı sıkışma kontrol mekanizmasıdır, burada bir yönlendirici sıra uzunluğunu görüntüler ve paketlerin sıraya nasıl alınacağına karar verir. Geleneksel yönlendiriciler bir drop-tail politikası kullanırlar, bu mekanizmada eğer sıra dolu değilse paketler sıraya alınırlar. Bundan dolayı bir drop-tail yönlendirici sadece sıranın dolu olup olmadığına bakar. AQM yönlendiricileri, sıra dolmadan önce potansiyel olarak paketleri düşürür. Bu aksiyonlar, paketler düşürüldüğü zaman gönderme oranı azaltılan, TCP Reno gibi, ağ trafiğinin büyük kısmında kullanılan bir sıkışma kontrol algoritması ile yapılan temel varsayımlara dayanır. Birçok AQM algoritması nispeten daha küçük sırayı devam ettirmek için tasarlanır ama paketleri sıraya koymak için paketleri düşürmeden kısa patlamalara izin verir. Sırayı küçük tutmak amacıyla "erkenden" sıklıkla birçok paket düşürülür, yani sıra dolmadan önce. Küçük bir sıra, paketlerin düşmeden daha küçük gecikmelerle sonuçlandırır [3].

TCP'nin şimdiki versiyonları sıkışmanın göstergesi olarak kayıplara güvenir. Açıkca, eğer birisi kayıpların düşük seviyedeki kayıplarda ağı işletmek isterse, bu istenen bir durum değildir. Diğer taraftan, kayıplar sıkışmanın iyi bir göstergesidir ve az veya hiç sıkışma olmayacağına, sıkışma kontrol kararı için ağdan başka sinyallere ihtiyaç duyar. Son zamanlarda, ağdaki sıkışmanın yakın zamanda, kaynaklara erken bildirmek için kesin sıkışma bildirisi (Explicit Congestion Notification, ECN), önerilmiştir. ECN işaretlemesi, ağ hakkındaki böyle bilgileri kullanıcılara sağlamak için kullanılan bir mekanizmadır [8].

ECN işaretlerini sağlamak için, yönlendiriciler, ağın şimdiki durumu hakkındaki bilgiyi kullanıcılara taşıyan paketleri zekice işaretlemelidir. Böyle bilgileri taşımak için

yönlendiricilerde çalıştırılan algoritmalar *Aktif Sıra Yönetimi* (AQM) şemaları olarak adlandırılır. Bunlardan bazıları,

- Drop Tail
- RED
- REM
- BLUE
- GREEN
- PURPLE

Genel olarak, AQM şemaları, kontrol akışları ile sıkışmayı kontrol eder. Sıkışma ölçülür ve ona göre yapılacaklar belirlenir. Sıkışmayı ölçmek için esas olarak iki yaklaşım vardır.

1. Sıra Tabanlı
2. Akış Tabanlı

Sıra tabanlı AQM'lerin sıkışmaları sıra uzunluğu tarafından elde edilir. Bunun dezavantajı, kontrol mekanizmasının, sıra pozitif olduğu zaman, sıkışma olarak paketleri geciktirmesidir. Bu, gereksiz gecikme ve gecikme değişimlerine sebep olur. Diğer taraftan akış tabanlı AQM'ler, sıkışmaya karar verir ve paketlerin alınma oranına göre yapılacakları belirler. Böyle şemalar için, ihmal etme ve tüm zıt çıkarımlar kontrol mekanizması için gereksizdir [6].

Bir aktif sıra yönetim mekanizmasının amacı, aşağıdaki gibi özetlenmiştir [7].

1. Yönlendiricilerde düşürülen paketlerin sayısını azaltmak : Ortalama sıra uzunluğu küçük tutulur, bundan dolayı patlamalar için yeterli alan ayrılır.
2. Daha düşük etkileşim servisini destekler : Ortalama sıra uzunluğunun küçük tutulmasıyla, uçtan uca gecikmeler daha kısa olur.

3. Dışarda bırakma davranışından kaçınma : Düşük bant genişliği ve patlak akışlara karşı eğilimlerden kaçınmak. Tampon belleğe yeni gelen bir paketin hemen hemen her zaman yer bulabileceğini garanti etmektir.

3.1 DROP-TAIL

Yönlendiricilerdeki sıra uzunluğunu yönetmek için en bilindik ve geleneksel teknikdir. Bu teknikde her bir sıra için en fazla sıra uzunluğu (paket cinsinden) ayarlanır. Sıra en fazla uzunluğa ulaştığı zaman, sonra gelen paketleri reddeder yani düşürür, taki sıra uzunluğu düşünceye kadar. Çünkü, sıradaki bir paket iletilmiş olmalıdır. Bu teknik "*Drop Tail*" olarak bilinir, sıra dolu olduğu zaman son alınan paketlerin hemen hemen hepsi düşürülür. Bu metod internette uzun yıllar iyi bir şekilde kullanıldı, ama bu metodun iki önemli dezavantajı vardır [7].

1. **Dışarda Bırakma :** Bazı durumlarda, drop tail tek bir bağlantıya izin verir yada sıradaki yerleri diğer bağlantılardan korumak için bazı akışlar tekellerine alırlar. Bu "dışarda bırakma" olayı genellikle eşlemenin yada diğer zamanlama etkilerinin sonucudur.

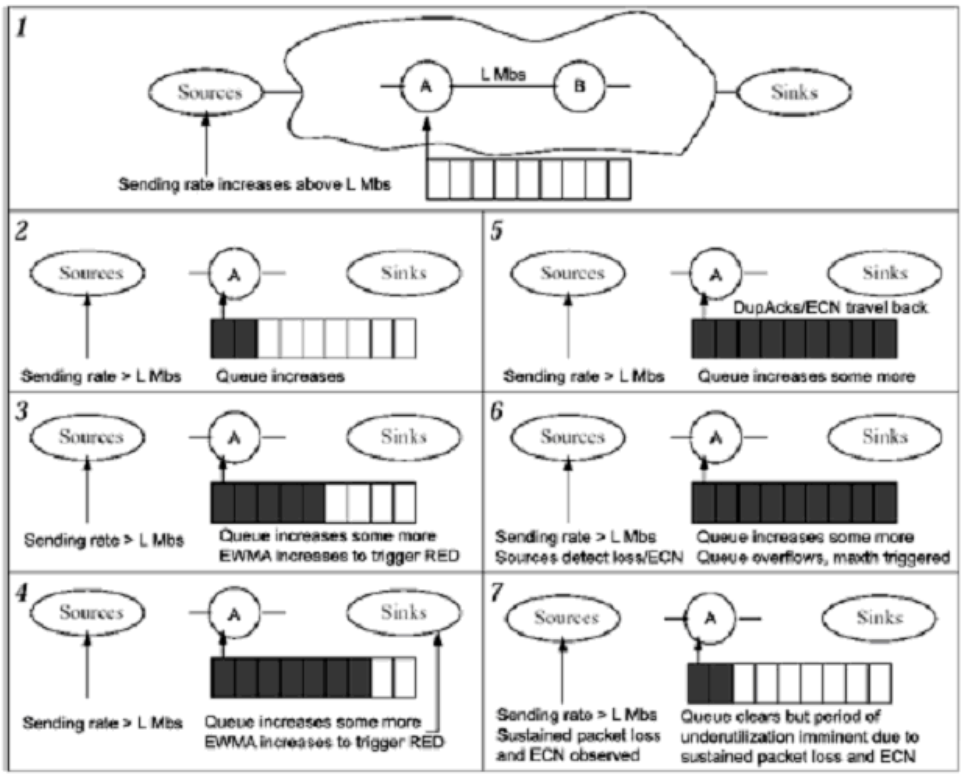
2. **Dolu Sıralar :** Drop tail disiplini, sıralara uzun zaman periyodu için dolu durumunda kalmasına izin verir, bundan dolayı sıkışma sinyali sadece sıra dolu olduğu zaman oluşur. Bu kararlı durum sıra uzunluğunu düşürmek için önemlidir ve bu sıra yönetiminin en önemli amacıdır .

Kısaca, drop tail etkin bir yönetim değildir. Ağdaki talepler arttığı zaman hatlar boyunca geçen verinin yönetilebilir miktarı daha uzun olamaz. Daha sonra ilk defa sıkışmayı kontrol etmek için " Rastgele Erken Tespit" (Random Early Detection, RED) olarak bilinen gerçek yeni bir AQM algoritması geliştirildi.[8]

3.2 RED

Drop tail sıralarının üzerindeki TCP'nin sıkışıklık kontrol algoritması ile en büyük problemlerinden birisi, kaynakların iletim oranlarını, sadece sıra taşması yüzünden oluşan paket düşmesinin ardından düşürmedir. Yönlendiricideki paketin düşmesi ile bunun kaynaklarda tespit edilmesi arasında büyük bir zaman geçtiğinden dolayı, ağın desteklemediği oranda devam eden iletim, bir çok paketin düşmesine neden olabilir. RED, henüz yeni başlayan sıkışmanın tespiti ve sıkışmanın uç noktalara bildirilmesiyle bu problemi yatıştırmıştır. Böylece sıradaki taşmalar oluşmadan, kaynakların iletim oranlarını düşürmelerine izin verir.

RED algoritmasını ilk defa 1993 Ağustos'unda Van Jacobson ve Sally Floyed tanıtmışlardır [4,13]. RED, paket düşmelerini ve sıra gecikmelerini en aza indirmek amacıyla kaynakların toplu eşlemelerinden kaçınmak, yüksek hat kullanımını sağlamak ve patlamalı kaynaklara karşı olan ön yargıları silmek için tasarlanmışdır.



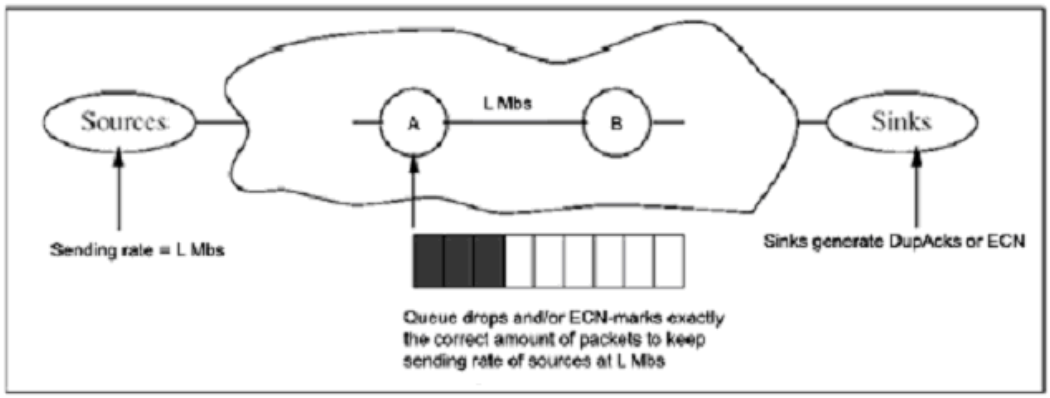
Şekil 3.1. *RED örneği [16]*

Geniş sayıda TCP kaynağının aktif olduğu ve daralan kısımda kullanılan tampon kapasitesinin küçük olduğunda oluşan sıkışma senaryosu şekil 3.1 ‘de gösterilmiştir. Şeklin gösterdiği gibi, t = 1 anında TCP yükündeki yeterli değişimler, TCP kaynaklarının iletim oranının, hattın daralan kısmındaki kapasiteyi aşmasına sebep olmuşlardır. t = 2 anında kapasite ve yük arasındaki eşleşme daralan kısımda bir sıranın oluşmasına sebep olur. t = 3 anında ortalama sıra uzunluğu min_{th} ‘ı aşar ve sıkışma kontrol mekanizmalarının tetiklenmesine sebep olurlar. Bu noktada, sıkışma bildirisi, sıra uzunluğuna ve işaretlenme olasılığı max_p ye bağlı olan oranlarda uç noktalara geri gönderilir. t = 4 anında TCP alıcıları ya paket kayıplarını tespit eder yada ECN biti ayarlanmış paketleri elde eder. Cevap olarak, aynı doğrulamayı ve varsa TCP tabanlı ECN sinyalini kaynaklara geri gönderir. t = 5 anında tekrarlanan doğrulamalar ve varsa ECN sinyalleri sıkışma sinyali için kaynaklara geri gönderilirler. t = 6 anında, kaynaklar sonunda sıkışmayı tespit eder ve iletim oranlarını düşürürler. Son olarak t = 7 anında, daralan kısımda önerilen yükde bir düşüş elde edilir. Toplu TCP kaynaklarının agresifliğine ve daralan kısımdaki tamponda mevcut olan boşluğun miktarına bağlı olarak, paket kayıplarının büyük miktarı ve varsa belirleyici ECN işaretlemesi oluşur. Böyle davranma sonuç olarak daralan hattın kullanım altında olmasını sağlar [16].

Bu problemi çözmenin bir yoluda, RED yönlendiricilerinde geniş miktarda tampon kullanılmasıdır. Örneğin, RED’in iyi çalışması amacıyla, bir orta yönlendirici, iki katı miktarda bant genişliği gecikmesi olan tampon alanı gerektirmektedir. Bu yaklaşım, aslında, sayıları artan yönlendirici satıcıları tarafından alınmıştır. Maalesef, geniş bant genişliği gecikmesi olan ağ ürünlerinde geniş miktarda tampon kullanımı, uçtan uca gecikme ve gecikme stresi ekler. Bu uygulamalar arasındaki çalışma yeteneğini şiddetli bir şekilde azaltır. Buna ek olarak, sınırlı hafıza kaynaklarına sahip hedeflendirilmiş yönlendiricilerin çokluğu, bu çözümü istenmeyen hale getirir [16].

Şekil 3.2’de ideal bir sıra yönetim algoritmasının nasıl çalıştığı gösterilmiştir. Bu şekilde, sıkışık yönlendiriciler, sıkışıklık bildirisini TCP’nin toplu iletim oranlarını tutan yada altında olan bir oranda dağıtır. RED bu ideal işletim noktasını başarabilirken, uygun

miktarda tampon bellek alanına sahip olduğu zaman ve doğru şekilde parametreleri ayarlandığında yapabilir [16].



Şekil 3.2. *İdeal senaryo [16]*

RED kapıları, üstel ağırlıklandırılmış taşıma ortalaması ile bir low-pass filtre kullanarak, ortalama sıra uzunluğunu hesaplar. Ortalama sıra uzunluğu en düşük ve en yüksek eşik değerleri ile karşılaştırılır [4]. Ortalama sıra uzunluğu en düşük eşik değerinden düşükse, hiç bir paket işaretlenmez. Ortalama sıra uzunluğu en büyük eşik değerinden büyükse, her alınan paket işaretlenir. Eğer işaretli paketler düşürülürse yada tüm kaynaklar işbirliği yaparsa, ortalama sıra uzunluğunun en büyük eşik değerini ciddi bir şekilde aşmaması sağlanır [4].

Ortalama sıra uzunluğu en düşük ve en yüksek eşik değerleri arasındaysa, her alınan paket *pa* olasılığı ile işaretlenir. Buradaki *pa* , ortalama sıra uzunluğu *avg* nin bir fonksiyonudur. Paketin işaretlendiği her zaman, olasılık, belli bir bağlantıdan işaretlenen paketin kabaca bağlantının kapıdaki bant genişliğinin paylaşımına uygun olmasıdır. Genel RED algoritması aşağıda verilmiştir [4].

alınan herbir paket için
ortalama sıra uzunluğu avg'yi hesapla
eğer min_th<avg<max_th ise
pa olasılığını hesapla

pa olasılığı ile:
alınan paketi işaretle
eğer max_th<avg
alınan paketi işaretle

RED, TCP ile etkileşir, kaynak oranının artmasıyla sıra uzunluğu büyür, daha fazla paket işaretlenir, kaynakların oranları düşürülür, devir tekrarlanır. AQM sıkışma ölçüsünün nasıl güncellendiğini belirtirken, TCP de kaynak oranlarının nasıl ayarlandığını belirtmektedir. RED için, sıkışma ölçüsü sıra uzunluğudur ve tampon işlemi tarafından otomatik olarak ayarlanır. Sonraki periyoddaki sıra uzunluğu, şimdiki sıra uzunluğu artı toplu giriş eksi çıkışdır [17].

$$b_l(t+1)=[b_l(t)+x_l(t)-c_l(t)]^+ \qquad (3.1)$$

$[z]^+ = max\{z,0\}$ dır. Burada $b_l(t)$, t periyodu içerisindeki l sırasında toplu sıra uzunluğudur. $x_l(t)$, t periyodu içerisindeki l sırasına toplu giriş oranıdır ve $c_l(t)$, t periyodu içerisindeki çıkış oranıdır [17].

3.3 BLUE

BLUE algoritması, RED'in problemlerinin bazılarını, sıkışma ölçü şemasının bir sıra ölçüsü ile beraber melez akış kontrol şemasının kullanımasıyla çözümler. Sıkışma bildirim oranını değiştirmek için akış ve sıra olaylarını kullanır. Bu oran iki faktör tarafından sağlanır, sıra sıkışmasından paket düşmesi ve hat kullanımı. BLUE algoritmasını RED'den ayıran anahtar farklılık, ortalama sıra uzunluğundan ziyade paket kayıplarının kullanılmasıdır [13].

BLUE paketleri işaretleme için tek bir olasılık sağlar, P_m. Eğer sıra bellek taşması yüzünden sürekli paketleri düşürüyorsa, BLUE P_m yi arttırır, bundan dolayı sıkışma bildirisini yada düşen paketleri geri bildirme oranı artar. Ters olarak, eğer sıra boş olursa yada hat kullanımda değilse, BLUE işaretleme oranını azaltır. Bu BLUE'ya etkin

olarak geriye gönderdiği sıkışma bildirisi veya düşen paketler için gerekli doğru oranı bulmasını "öğretir" [16].

BLUE'nın tipik parametreleri *d1,d2 ve donma zamanı* dır. *d1* , sıra taştığı zaman arttırılan P_m nin miktarını belirtir. *d2,* hat boş kaldığı zaman düşürülen P_m nin miktarını belirtir. Donma zamanı ise önemli bir parametredir ve P_m nin iki başarılı güncellenmesi arasındaki en küçük aralığı belirtir. Bu değer tekrar güncellenmeden önce etkisini almak için işaretleme olasılığının değişmesine izin verir. Bu parametrelere dayanaran temel blue algoritması şu şekildedir [16].

Hattın boşta kalmasına bağlı olay	Paket düşmesine bağlı olay:
Eğer((şimdi – son güncelleme) > donma zamanı) $P_m = P_m - d2$; sonGüncelleme = şimdi;	Eğer((şimdi – son güncelleme) > donma zamanı) $P_m = P_m + d1$; sonGüncelleme = şimdi;

Burada BLUE ile ilgili bazı problemler vardır [13].

- BLUE, cevaplanmayan akışlar için adresleme fonksiyonu kullanır. Bu cevaplanmayan akışların sayısının çok büyük olmadığı varsayılır. Bunun doğru olduğunu biliriz fakat bu varsayımın doğru olmadığı durumlarda olabilir.
- Cevaplanmayan akışların sayısı fazla olduğu zaman kutular kirlenebilir ve TCP akışları cevap olarak hatalı olabilir, sonuçda onları gereksizce cezalandırır.
- Bunun olabilmesi için bir çözümde, düzenli zaman aralıklarında adresleme fonksiyonunu değiştirmektir. Bu bazı cevaplayan akışlara kirlenmemiş kutular için yol gösterir.
- Diğer bir problem de, bir akış bir kere işaretlemişse, o daima kirletilmiştir. Eğer sonra akış kendi kendine engelliyorsa, BLUE hala paket düşmesi nedeniyle gönderme oranlarını düşürmeye çalışır.

3.4 REM

REM, basit ve kararlı biçimde hem yüksek kullanım hem de ihmal edilebilir kayıp ve gecikmeyi gerçekleştirmeyi amaçlar. Bunu başarmak için anahtar düşünce, sıkışma ölçüsünü, kayıp, sıra uzunluğu yada gecikme gibi performans ölçüsünden ayırmaktır. Sıkışma ölçüsü bant genişliği için talebi aşmayı belirtir ve kullanıcı sayısının kaydını tutmak zorundadır, performans ölçüsü ise kullanıcı sayısından bağımsız olarak hedefleri etrafında kararlı tutulabilmesidir.

REM aşağıdaki anahtar özelliklere sahiptir [17] .

1. **Oran eşleştirme temiz bellek** : Tampon bellek temizken (küçük bir hedef etrafında kararlı sıralar), kullanıcı sayısını önemsemeden, kullanıcı oranlarını ağ kapasitesine eşleştirmeye çalışır.

2. **Ücretlerin Toplanması** : Uçtan uca işaretleme (yada düşme) olasılığı, basit ve kesin bir tarzda, kullanıcının yolundaki tüm yönlendiricilerin üzerindeki toplanan hat ücretlerinin (sıkışma ölçüsü) toplamı tarafından dikkatle incelenir.

'Oran eşleştirme temiz bellek' özelliği, alışıldık bilimin aksine, yüksek kullanımın ağ da geniş geciktirilmiş işlerin tutulmasıyla başarılmadığını, ama kullanıcı için oranlarını ayarlayarak doğru geri besleme ile başarılabilmesini sağlar. REM'in ilk düşüncesi, hattı paylaşan kullanıcının sayısını önemsemeden, hem giriş oranı hat kapasitesi etrafında kararlıdır, hemde sıra küçük hedef etrafında kararlıdır. Herbir sıra çıkışı, sıkışma ölçüsü olarak REM'in devam ettirdiği ve 'ücret' olarak adlandırılan bir değişken gerçekleştirir. Bu değişken sonraki alt bölümde de açıklanacağı gibi işaretleme olasılığını elde etmek için kullanılır. Ücret, oran eşlemesine (yani giriş oranı ve hat kapasitesi arasındaki fark) ve sıra eşleşmesine (yani sıra uzunluğu ve hedef arasındaki fark) periyodik olarak ya da eş zamanlı olmadan güncellenir. Eğer bu eşleşmeyenlerin ağırlıklandırılmış toplamı pozitifse, ücret arttırılır aksi takdirde düşürülür. Giriş oranı hat kapasitesini geçerse yada temizlenmiş olmak için gecikmiş işler varsa ağırlıklandırılmış toplam pozitifdir, aksi takdirde negatifdir. Kaynakların sayısının artmasıyla, orandaki ve sıra büyümesindeki uygunsuzluk ücreti arttırır ve bundan dolayı işaretleme olasılığıda artar. Bu kaynaklara oranlarını düşürmesi için daha güçlü bir sıkışma sinyali gönderir.

Kaynak oranları çok küçük olduğu zaman, eşleşmeyenler negatif olur, ücret ve işaretleme oranı düşer ve kaynak oranı yükselir taki eşleşmeyenler sıfıra getirilinceye , yüksek kullanım ve dengedeki ihmal edilebilir kayıp ve gecikme kazanılıncaya kadar. Dengede, eğer hedef sıra sıfıra ayarlanmışsa, tampon bellek temizlenir [17].

Halbuki RED'de sıkışma ölçüsü (sıra uzunluğu), bellek işlemi tarafından (1)'e göre otomatik olarak güncellenir, REM açıkca ilk özelliğini getirmek için ücretini güncellemesini kontrol eder. Tam olarak, *l* sırası, *t* periyodundaki $p_l(t)$ ücreti için [17] 'ye göre güncellenir.

$$P_l(t+1)=[\ P_l(t)+\gamma(\alpha_l b_l(t)-b_l^*)+x_l(t)-c_l(t))]^+ \qquad (3.2)$$

$\gamma > 0$ *ve* $\alpha_l > 0$ küçük sabitlerdir ve $[z]^+ = max\{z,0\}$'dır. Burada $b_l(t)$, *t* periyodunda *l* sırasındaki toplu bellek işgalidir ve $b_l^* \geq 0$ hedef sıra uzunluğudur, $x_l(t)$, *t* periyodunda *l* sırasına toplu giriş oranıdır ve $c_l(t)$, *t* periyodundaki *l* sırası için mevcut bant genişliğidir. $x_l(t) - c_l(t)$ arasındaki fark eşleşmeyen oranı ölçer, $b_l(t) - b_l^*$ arasındaki fark ise eşleşmeyen sırayı ölçer. α_l sabiti herbir sıra tarafından kişisel olarak ayarlanabilir ve kullanımı ve sıra gecikmesi iletim sırasında değişebilir. γ sabiti REM'in ağ şartlarında değişimlere cevap vermesini kontrol eder. Bundan dolayı eşitlik (3.2) den, eğer oranın ve sıra uygunsuzluklarının ağırlıklı toplamı α_l tarafından ağırlıklandırılır, pozitifdir ve ücret arttırılır, aksi takdirde düşürülür. Dengede ücret kararlı ve ağırlıklı toplam sıfır olmak zorundadır. Yani, $\alpha_l (b_l - b_l^*) + x_l - c_l = 0$. Bu sadece giriş oranı kapasiteye eşitse ($x_l = c_l$) ve gecikme hedefe eşitse tutulabilir ($b_l = b_l^*$), bu ilk başta bahsedilen birinci özelliğe yol gösterir [17].

'Ücretlerin toplamı' özelliği kullanıcıların çoklu sıkışmış hat boyunca ilerlediği ağ içerisinde önemlidir. Kullanıcı tarafından dikkatle izlenen uçtan uca işaretleme (yada düşme) olasılığı içerisinde gömülü sıkışma bilgisinin anlamını açıklaştırır ve bu yüzden uyum oranının tasarımında kullanılır.

Sıra çıktısı, henüz işaretlenmemiş alınan herbir paketi, şimdiki ücretinin üstel artan bir olasılıkla sıranın akışına karşı işaretler, işaretleme olasılığının üstel formu geniş bir ağ

da önemlidir. Burada kaynakdan gidilecek yere çoklu sıkışmanın içinden geçen bir paket için uçtan uca işaretleme olasılığı yoldaki her hat da, hat işaretleme olasılığına bağlıdır. Sadece kişisel hat işaretleme olasılığı kendi hat ücreti içerisinde üstel olduğu zaman, bu uçtan uca işaretleme olasılığı, kendi yolunda tüm sıkışmış hatlarda hat ücretlerinin toplamı içerisinde üstel olarak artıyor olur. Bu toplam yoldaki sıkışmanın tam ölçüsüdür. Uçtan uca işaretleme olasılığına gömüldüğünden dolayı, yoldaki her hattadır. Bu kaynaklar tarafından işaretlenen paketlerinin parçasından kolayca tahmin edilebilir ve oranlarının ayarlanmasının tasarımı için kullanılır [17].

Bir paketin $l=1,2,....,L$ hatları içinde iletildiğini ve t periyodunda $p_l(t)$ ücretlerine sahip olduğunu varsayalım. Daha sonra işaretleme olasılığı $m_l(t)$, l sırasında t periyodunda [17].

$$m_l(t) = 1 - \phi^{-pl(t)} \qquad (3.3)$$

$\phi > 1$ bir sabittir. Daha sonra paket için uçtan uca işaretleme olasılığı şu şekildedir.

$$1 - \prod_{l=1}^{L} (1 - \pi_l(t)) = 1 - \phi^{-\sum_l pl(t)} \qquad (3.4)$$

Yani, yolun sıkışma ölçüsü, $\sum_l p_l(t)$, geniş olduğu zaman, uçtan uca işaretleme olasılığı yüksektir.

Hattın işaretleme olasılığı $m_l(t)$ küçük olduğu zaman, buna bağlı olarak hat ücretleri $p_l(t)$ küçüktür, eşitlik (3.5) de verilen uçtan uca işaretleme olasılığı aşağı yukarı yoldaki hat ücretlerinin toplamına uygundur.

$$\textit{Uçtan uca işaretleme olasılığı} \cong (\log_e \phi \sum_l pl(t)) \qquad (3.5)$$

3.5 GREEN

GREEN algoritması, cevapda akış tabanlı sıkışma ölçüsü olan $x_{est,}$ tahmin edilen veri alış oranı olan sıkışma bildirisinin oranını ayarlayan bir geri besleme kontrol fonksiyonudur. GREEN bir eşik değeri fonksiyonu temellidir. Eğer hattın tahmin edilen veri alış oranı x_{est} hedef hat kapasitesi c_l üzerinde ise, sıkışma bildirisinin oranı P, ΔP tarafından $1/\Delta T$ oranında arttırılır. Tersine eğer x_{est} , c_l altındysa P, ΔP tarafından $1/\Delta T$ oranında azaltılır. Bu algoritma gelen paketlerin olasılıklı işaretlerine P oranında, düşen paketler ya da ECN biti ayarlananlar tarafından uygulanır. Adım fonksiyonu $U(x)$ [6,18] tarafından tanımlanır.

$$U(x) = \begin{cases} +1 & x \geq 0 \\ -1 & x < 0 \end{cases} \qquad (3.6)$$

Bundan dolayı,

$$P = P + \Delta P.U(x_{est} - c_l) \qquad (3.7)$$

Hedef hat kapasitesi c_t, gerçek kapasite c olarak aşağıda atanmıştır, tipik olarak 0.97 c dir, böylece sıra büyüklüğü 0'a yaklaşır. Gelen veri oran tahmini, üstel ortalama kullanılarak gösterilmiştir.

$$x_{est} = (1-\exp(-Del/K))*(B/Del)+\exp(Del/K)* x_{est} \qquad (3.8)$$

Del paketler arası gecikmedir, B paket ölçüsü ve K zaman sabitidir. Diğer gelen oran tahmin teknikleri de başarılı bir şekilde kullanılabilir.

REM ve GREEN arasında bir ilişki vardır. Eğer eşitlik (3.3) doğrusalsa, $m = P$ dir ve üstel terim dikkate alınmaz. Dahası eğer bellek terimi $\alpha = 0$ ise, ve doğrusal sabit γ adım fonksiyonu ile yer değiştirirse, GREEN'in sıkışma bildirim oranı P, REM'in ücreti P_l ye eşit olur [6].

3.6 PURPLE

PURPLE yaklaşımı, diğerlerine karşın, tepki oluşturan protokollerin davranışı üzerinde kendi aksiyonlarının etkisini tahmin eder. Bundan dolayı kısa-dönem gelecek trafikdir [19]. PURPLE, ağdaki sıkışma durumu hakkındaki uçtan uca bilgiyi analiz ederek bunu başarır. PURPLE, ana AQM parametrelerinin, en azından yerel bir en iyiliğe doğru, daha hızlı birleşmesine izin verir, buna bağlı olarak düzleştirme ve küçültme hem sıkışma geri beslemesi hemde sıra işgalidir. Tahmini geliştirmek için [19]'da, bu pasif olarak gösterilen bilgi sıkışmanın miktarı ile ilgilidir. Ağda başka yerde akışlar bu yönlendiriciden geçiyormuş gibi görünür.

PURPLE paket işaretleme düzeltme ve parametreler ayarlanmadan sıra gecikmesi sağlar çünkü çevrim içi olarak iyileştirilmiştir. Kendi kendini idare eden davranış çevrim içi model tabanlı tahminlere güvenir. Bu ayrıca goodput, işlem hacmi ve ortalama gecikme arasında mükemmel bir denge sağlarsa drop-tail kayıplardan tamamen kaçınabilir. Bu, çok az güç ve durum bilgisi ve üç yeni mekanizma olan uçtan uca sıkışma analizi, ECN bilgisinin görüntülenmesi ve TCP model eşitliğinin kullanarak başarılmıştır. [19]'da çok çeşitli durumlar için simüle edilmiş ve PURPLE'dan çok iyi davranışlar alınmıştır.

3.7 ECN

Şimdiye kadar sıkışma olduğu zaman kapılardan paketlerin atılması hakkında konuştuk, bu şekilde son noktanın da sıkışmadan haberi oldu çünkü geri iletim zaman aşımı tecrübe edildi. Ama, örneğin RED algoritması ile tampon bellek gerçekten dolmadan önce paketler düşürülmeye başlandı. Bu geliştiricileri, paketlerin basitce nasıl düşürüldüğünden başka çözümler üzerinde düşündürmeye başladı. Örneğin sıkışmayı son noktaya sinyal olarak iletmek için paketleri kaybetmeden hala tampon bellek içerisinde dururken paketlerin bir bayrak ile işaretlenmesi gibi bir yaklaşım geliştirildi ve bunun

ismine erken sıkışma bildirisi anlamına gelen (ECN) Early Congestion Notification adını verdiler [20].

Bu öneride, QoS tanımlamasında (quaility of service) Diffserv tekniği kullanıldığında sıkışma başladığı zamanki sinyal için, IPv4 başlığının ToS alanındaki 2 biti kullanıldı. (servis sınıfını belirtmek için 8 bitin sadece 6 sı kullanılıyor, diğer kalan 2 bit kullanılabilir). Bu iki bitten birisi CE (Congestion Experienced) oldu, diğeri ise bir hostun ECN olabileceğini bilmek için ayarlandı (ECT) [20].

Bu düşünce çok anlaşılırdır, bağlantının başında, gönderici ECT bitini kapasitesini bildirmek için ayarlar. Eğer alıcınında kapasitesi varsa, TCP başlığındaki bir bayrakla geri gönderir. Alıcı, işaretli bir paket aldığı zaman, sıkışmayı işaret etmek için göndericiye bir ACK gönderir ve gönderici kullanılan sıkışıklıkdan kaçınma tarafından uygun yolla bir karşılık verir [20].

Tüm sistemi tamamlayabilmek için, TCP/IP protokolünde çok küçük değişikliklere ihtiyaç vardır. IP başlığında bulunan iki bite değer atanması (gönderici tarafından EC'yi işaret etmek için ve ECN kapasitesini işaret etmek için), TCP başlığındaki iki bitin atanması. Bu ikisi bir EC paketi için geri göndermeleri tekrarlatır ve göndericiye, pencerenin düşürüldüğünü ve tekrarlamaları kesmesini alıcıya işaret edebilmesine izin verir (CWR biti); bu bit ayrıca sinyale bağlantı sağlanırken, alıcı sinyale, kapasitesini ACK içerisinde bir SYN paketi için ayarlamasına izin verir [20] .

Bu yaklaşımla bazı problemler ortaya çıkabilir, birisi, örneğin kapasitesi yeterli olmayan bir kullanıcı, kapasitesini ayarlayabilir. Böylelikle EC biti ayarlanmış bir paket aldığında, sıkışmanın kullanıcısına ilan etmeyecek. Ama, eğer bir host düşen bir paket için sıkışmadan kaçınma ile cevap vermezse, aynı problemler olabileceğinden, yazarlar yeniden cevap verirler. Bazı kişilerde düşen paketlerin yüksek yük periyodlarında trafiği düşürmenin bir yolu olduğunu söylediler, ECN ağ hala paketleri düşürmektedir. Diğer bir konuda düşen EC paketlerinin olabilmesidir. Bu durumda gönderici sanki bir ECN ağının içinde değilmiş gibi herhangi bir yolla sıkışıklıkdan kaçınma ile cevap verir. Sonuç olarak IPsec tüneli ile ilgili problemler, herhangi bir şifreli hesaplamada ECN biti yeterince uzun

bir şekilde içerilemez. IPsec, ECN nin değerini değiştirmeye çalışan bir düşmana karşı herhangi bir koruma geliştirmemiştir [20].

BÖLÜM 4

ADAPTE EDİLMİŞ RED(Adaptive RED) ALGORİTMASI

İnternette, sıkışıklık çökmelerinden korunmak için genellikle uçtan uca sıkışıklık kontrolü kullanılır. Bunun yanında, veri trafiğinin doğası gereği patlaklı olmasından dolayı, yönlendiriciler patlakları içine alabilmesi ve yüksek hattan yararlanmayı sürdürmek için oldukça büyük tampon bellekler ile tedarik edilir. Bu geniş belleklerin dezavantajı, eğer bilindik drop-tail yönetimleri kullanılırsa, sıkışık yönlendiricilerde yüksek sıra geciklemeri olur. Bundan dolayı drop-tail bellek yönetimi ağ yöneticilerini geniş bellek gerektiren, yüksek kullanım yada küçük bellek gerektiren düşük gecikmeden birisini seçmeye zorlar.

RED tampon bellek yönetim algoritması, paketleri, ortalama sıra uzunluğunun artması olarak, artan olasılıkla rastgele düşürerek sıra daha aktif, anlamlı olarak yönetilir; paketlerin düşme oranı ortalama sıra uzunluğu minimum eşik değerli(min_{th} olarak gösterilir) RED parametrelerindeyse, ortalama sıra uzunluğu maksimum eşik değere (max_{th}) ulaştığında düşme oranı sıfırdan itibaren doğrusal olarak artar. RED'in ana amaçlarından bir tanesi, sıra uzunluğu algoritması ve erken sıkışma bildirimi kombinasyonunu kullanarak, düşük ortalama sıra gecikmesi ve yüksek işlem hacmini birarada başarmakdır. RED'in benzetme denemeleri ve işlemsel deneyler bu konuda oldukça başarılıdır.

Bunların yanında RED'in esas zayıf noktası, sıkışma seviyesinde ve parametre ayarlarında ortalama sıra uzunluğu çeşitlidir. Hat biraz sıkıştığında ve/veya max_p yüksektir, ortalama sıra uzunluğu neredeyse min_{th} dır. Hat ağır bir şekilde sıkıştığı zaman ve max_p düşükse, ortalama sıra uzunluğu max_{th} a yakın hatta üzerindedir. Sonuç olarak ortalama sıra gecikmesi RED de trafik yüküne ve parametrele duyarlıdır, ve bundan dolayı ilerde tahmin edilebilir değildir. Gecikme, müşterilere dağıtılan servisin kalitesi için esas bileşendir. Ağ yöneticilerinin doğal olarak, sıkışmış ağlarda kabaca bir ortalama

gecikme tahmini için öncelikleri vardır. RED ile böyle ortalama gecikmeleri önceden söyleyebilmek için varolan trafik ayarlarının sağlanması için sabit RED parametrelerinde ayarlamalar yapılması gerekmektedir.

RED'in ikinci bir zayıf noktası, işlem hacminin de trafik yüküne ve RED parametrelerine duyarlı olmasıdır. Özellikle, ortalama sıra max_{th} dan büyük olursa, RED sık sık rolünü yerine getiremez, sonuç olarak da önemli şekilde işlem hacmi düşer ve düşen paketlerin oranı artar. Bu yönetimden kaçınmak için RED parametrelerinde tekrar sabit değişiklikler yapmak gerekebilir.

Bu tür problemlerden kaçınmak için aktif sıra yönetimi ile ilgili olarak birçok öneriler vardır. Revize edilmiş öneri bu bölümde gerçekleştirilmiştir ve NS simülatöründe simüle edilmiştir. Bu yeni versiyon, senaryolarda geniş değişiklikler yaparak ve RED'in diğer faydalarından feragat ederek hedeflenen ortalama sıra uzunluğunu başarmıştır. Bu sadece ortalama sıra gecikmelerini daha tahmin edilebilir yapmakla kalmaz, max_{th} nında aşılma olasılığını düşürür; bundan dolayı Adaptive RED, hem paket kaybetme oranını hemde sıra gecikmelerindeki değişimi düşürür .

4.1 Metrikler and Senaryolar

RED'in yada aktif sıra yönetiminin esas amacı genel olarak, düşük ortalama sıra gecikmesi ve yüksek işlem hacminin sağlanmasıdır. Bu nedenle biz öncelikle ortalama sıra gecikmesi ve işlem hacmi metriklerine odaklanıyoruz. RED 'in ikincil bir amacıda, drop-tail sıra yönetim algoritmasında verilenlerin doğruluklarını bir dereceye kadar geliştirmek ve verilen ortalama sıra uzunluğunu, paket düşme ve işaretleme oranını en aza indirmektir. Biz adaptive RED'in güzel davranışlarını tartışmayacağız, zaten RED'in güzel davranışlarına benzerdir. Sadece Adaptive RED ve RED'in düşme oranına (drop rate) kısaca değineceğiz. Çünkü genellikle alçaltılmış düşme oranı davranışını, alçaltılmış işlem hacmi yansıtır [5].

Öncelikle, tüm metrikler yönlendirici tabanlıdır. Dosya transfer zamanı, paket gecikmesi gibi son kullanıcı metrikleri, algoritmanın geçerliliği için önemli ölçülerdir. Adaptive RED ile ilgili son kullanıcı metrikleri yönlendirici tabanlı metriklerde kolayca elde edilebilir. Ayrıca yönlendirici tabanlı metrikler (AQM) dinamiklerine daha doğrudan bakış sağlamaktadırlar [5].

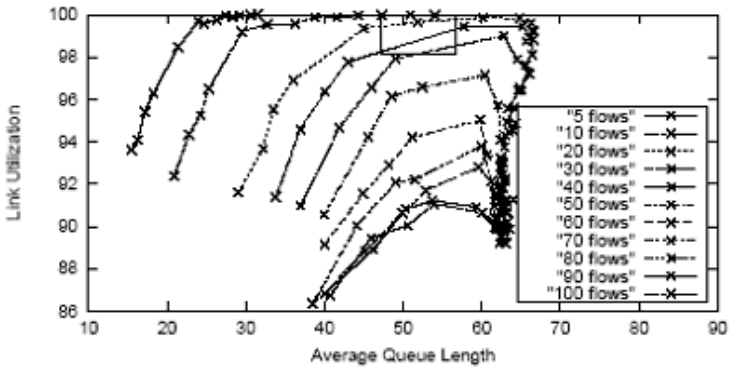
İkinci olarak en kötü durumla ilgili metriklerini dikkate almıyoruz, çünkü bu tür sıra gecikmelerini kontrol etmek AQM'lerin amacı değildir, bu tür en kötü durum sıra gecikmelerinin genişletilmesi için yönlendiricilerde sıraların bellek ölçülerinin konfigüre edilmesi ile doğrudan kontrol edilebilir [5].

Üçüncü olarak, sıra uzunluklarının salınım ölçüleriyle ilgili metrikleri doğrudan dikkate almayacağız [49,25]. Böyle salınımların, ortalama sıra gecikmesini arttırmadığı ve işlem hacmini düşürmediği sürece zararlı olduğunu düşünmüyoruz [5]. Salınımların etkisini ana metriklerimiz tarafından ilerleyen bölümlerde tartışacağız.

Adaptive RED'in gelişimi sırasında, Adaptive RED parametrelerinin hassasiyetini bulabilmek için çok sayıda trafik senaryosu inceledik. Sağlamlığı sağlamak için, iş yüklerinin sınıflandırılmasında, istatistiksel çoğullamaların seviyesinde ve sıkışmaların seviyesinde performansı düşündük. İş yükleri, ters yol trafiği boyunca, uzun yaşamlı akışları içerir. Ters yoldaki veri trafiğinin varlığı, ACK sıkışmasını ve paketlerinin kaybolduğunu bildirir, o nedenle ileriki yollarda veri trafiğinin patlaması artar. Ters yol trafiği aynı zamanda, ileriki yollarda paket trafiğinin sıralanmasına zorlarlar. Buradaki ileriki yol (forward path) veri ve ACK arasında paylaşılan şimdiki yoldur. Ayrıca simlasyonun sıkışma seviyeleri ve iş yükleri üzerindeki değişim senaryolarını araştırdık. ECN bildirimiyle (explicit congestion notification) ve ECNsiz baktık [5]. Son olarak, geniş pencere bildirileri ve farklı veri paket ölçülerini dikkate aldık [5].

4.2 Adaptive RED'e Alışmak

İlerleyen bölümler de belirtilen, Adaptive RED'in tasarım ve performans ayrıntılarına girmeden önce, RED'in parametrelerdeki hassasiyetini gösteren bazı benzetmeleri tekrar ettik [5], ve bu gösterdiki buradaki problem için Adaptive RED gerçek adres. Bu bölümde, RED'in bilenen karakteristik özellikleri olan ortalama sıra ölçüsü ve performans çeşitliliği, RED'in max_p ve w_q parametrelerinin bir fonksiyonu olarak, benzetmeleri şekillerle gösterilir. Bu bölüm aynı zamanda, min_{th} ve max_{th} arasındaki hedef sıra ölçüsünü yakalamak için max_p nin adapte edilmesini gösteren, Adaptive REM ile yapılmış benzetme sonuçlarını da gösterir. Benzetme senaryosunda da ayarlanmış bir max_p değeri ile de aynı sonucu alabilmeniz mümkündür. Bir başka deyişle Adaptive RED, çeşitli RED parametrelerini güvenli iyi sonuçlarla "otomatik ayarlamışdır".

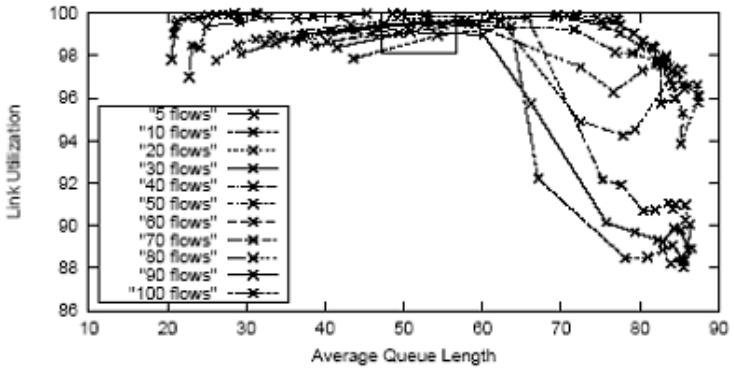


Şekil 4.1. *RED ile Gecikme – kullanım değişimi,* w_q*=0.002.*

Şekil 4.1 deki benzetmede, RED NS'in varsayılan değeri w_q =0.002 *ve* max_p = 0.1 ve min_{th} ve max_{th} sırasıyla 20 ve 80 paket olarak ayarlanarak kullanılmıştır, tüm bu benzetmelerde RED yavaş modda çalışmaktadır. Her bir çarpı, x ekseninde, 100-saniye benzetmesinin ikinci yarısı üzerindeki paketlerdeki ortalama sıra gecikmesini, y eksenide, benzetmenin ikinsi yarısı üzerindeki hat kullanma sayısını gösteren tek bir benzetmeden sonuçlardır. Herbir çizgi N akış ile benzetmeden sonuçlar göstermektedir, çizgilerde N değeri 5 ile 100 arasında sıralanmıştır. Herbir çizgi üzerindeki çarpılar, 0,5 soldan ve 0.02 sağdan max_p ile benzetme sonuçlarını göstermektedir. Benzetmelerdeki düşme oranı, sıfıra yakınken %8'e kadar çıkmaktadır. Şekil 4.1 de gösterildiği gibi, akışların sayısı ve max_p ile performans çeşitlilik gösterir, bu benzetmelerde daha düşük işlem hacmi ile daha

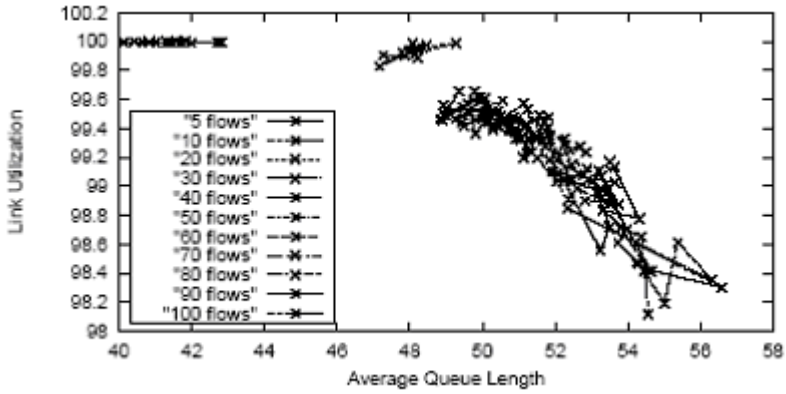
geniş sayıda uzun yaşamlı akış vardır. Bu benzetmelerde artan akış sayısı hat kullanımını düşürür ve artan ve max_p daha düşük sıra uzunluklarına liderlik eder. max_p nin düşük değerleri için kullanımdaki azalmanın yansımaları zamanın büyük bir bölümünde ortalama sıra uzunluğu max_{th} ı geçip gittiği durumlara yansır [5].

Daha sonraki bölümlerde göreceğimiz gibi, 15 mbps bant genişliğinde bir hat için 0,002 sıra ağırlığı çok geniştir, bu yüzden tipik 100ms RTT'ın bir parçası üzerindeki sıra uzunluğu ortalanır. Şekil 4.2 deki benzetmede şekil 4.1 dekinden sadece w_q , 0.002 yerine 0.0026 ye ayarlanmıştır. Bu sonraki bölümlerde detaylıca tartışılacaktır. RED'in performansı, ortalama sıra uzunluğu küçük çeşitli RTT'ler üzerinde yapıldığı zaman en iyidir. Tek bir RTT'nin bir bölümü üzerinde iyi değildir. Şekil 4.1 ve 4.2 RED'in performansının w_q parametresi üzerindeki hassasiytetinin bir kanıtıdır. RED'le iyi bir işlem hacmi ve kabul edilebilir ortalama sıra uzunluğu elde edebilmek için, w_q ve max_p nin iyi ayarlanması gereklidir. Adaptive RED, bu iyi ayarlanması gereken değerleri otomatik olarak ayarlamaktadır [5].



Şekil 4.2. *RED ile Gecikme – kullanım değişimi, w_q=0.00026.*

Adaptive RED algoritmasının ayrıntıları anlatılacaktır, yalnız adaptive RED'i genel olarak basitce w_q nun otomatik olarak ayarlanması (hat hızına göre) ve max_p nin cevapda sıra uzunluğuna göre düzenlenmesi olarak özetlenebilir. Şimdi Adaptive RED'i gerçekten öneren RED parametrelerini iyi ayarlamak için gerekirse silen bazı benzetmeler göstereceğiz [5].



Şekil 4.3. *RED ile Gecikme – kullanım değişimi,* w_q*=0.00026.*

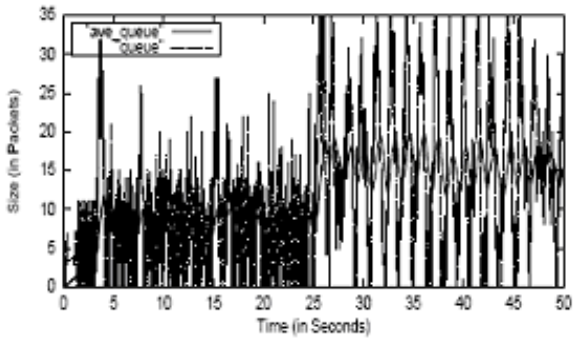
Şekil 4.3 adaptive RED ile aynı benzetmeyi gösterir; bu şekilde çeşitli değerler max_p nin ilk değerleridir, Adaptive RED, ölçülmüş davranışlarını sağlar. Şekil 4.3 deki x ve y eksenleri şekil 4.1 ve 4.2 deki ile eşleşmez, şekil 4.1 ve 4.2, şekil 4.3 için alanı gösteren bir kutu içerir. Şekil 4.1ve 4.2 nin "*iyi*" performans bölgesinde küçük bir alan meşgul eden tüm alan şekil 4.3 de resmedilmiştir. Önceki grafiklerde olduğu gibi, 100 saniyedir. Benzetmenin ikinci yarısındaki sonuçları göstermektedir; verilen eğrideki noktaların bir araya toplanması, esasen max_p nin ilk değerinin bağımsız olduğu sonucunu göstermektedir [5].

Bu benzetmeler Adaptive RED'in , w_q nun otomatik ayarlanması ve max_p nin o anki şartlara göre cevapta uyarlanması, ortalama sıra uzunluğunu hedeflenen aralıkda tutmasıyla yüksek işlem hacminin başarılmasında etkin olduğunu göstermektedir. Bu aralık, bölüm 4.4 de anlatılacağı gibi daha önceden tanımlı bir aralık çevresinde (min_{th}+ max_{th})*/2* ortalama sıra uzunluğunu sağlanması için gerekli algoritmalara uyumludur. Bu benzetmeler daha az paket düşürme, daha küçük ortalama sıra ve tam hat kullanımına sahiptirler [5].

4.2.1 Çeşitli Sıra Uzunlukları ile RED'in Gösterilmesi

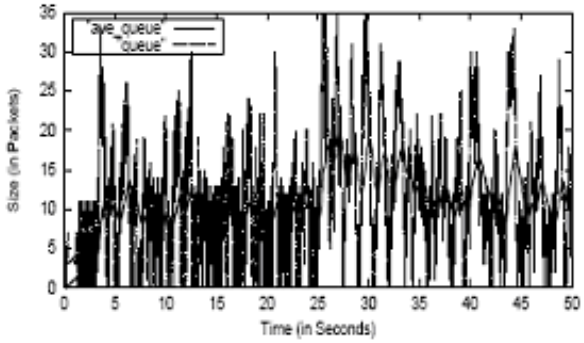
Daha önceki benzetmeler RED ve Adaptive RED'in istikrarlı performanslarını gösterdi. Şimdi RED ve Adaptive RED'in sıkışma seviyesinde hızlı değişim için nasıl cevap verdiklerini inceleyeceğiz. Sunulan benzetmeler, sıkışıklık seviyesi ile değişen ortalama sıra uzunluğunun iyi anlaşılmış dinamiklerini sunmaktadır [5]. Sonuçlar RED'in sabit ortalama sıra uzunluğundan paket düşürme olasılığı tablosundandır. Adaptive RED için, bu benzetmeler bir sıkışma seviyesinden diğerine iletime odaklanmıştır.

Bu benzetmeler, 1,5 Mbps'lık sıkışmış bir hat ile basit bir halter topolojisini kullanırlar. Tampon bellek 1500 byte paket için 35 paket yer ayırır. Tüm benzetmelerde w_q *0.0027 y*e, min_{th} da 5 pakete ve max_{th} da 15 pakete ayarlanır.



Şekil 4.4. *Sıkışıklıkda artışla RED.*

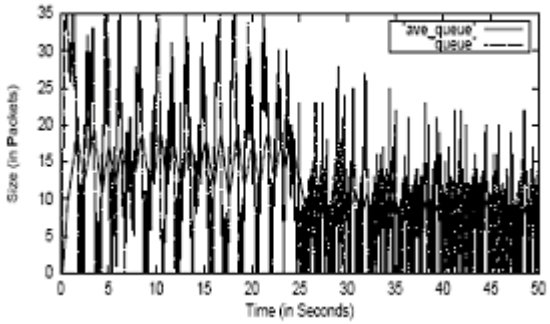
Şekil 4.4 daki benzetmede, ileri giden trafik uzun yaşamlı iki akış içerir, geri dönen trafikde uzun yaşamlı bir akış içerir. 25 inci zamanda, her 0,1 sn de bir, 20 yeni akış başlar. Herbiri 20 paketin maksimum penceresi ile oluşur. Bu gerçekci yükü modellemek için değil de, sıkışma seviyesindeki keskin değişimin etkisini basitçe, daha iyi gösterebilmek içindir. Şekil 4.4 daki grafik uyarlanmamış olan RED'in ortalama sıra büyüklüğündeki değişimi, paket düşme oranının bir fonksiyonu olarak gösterir. Koyu çizgi, RED tarafından tahmin edilen ortalama sıra büyüklüğünü göstermektedir. Kesikli çizgiler anlık sırayı göstermektedir. Paket düşme oranı, benzetmenin ilk yarısının üzerinde , %1 den, ikinci yarısının üzerinde %12.6 ya değişir. Buna karşılık gelen ortalama sıra uzunluğu ile değişir.



Şekil 4.5. *Sıkışıklıkda artışla Adaptive RED.*

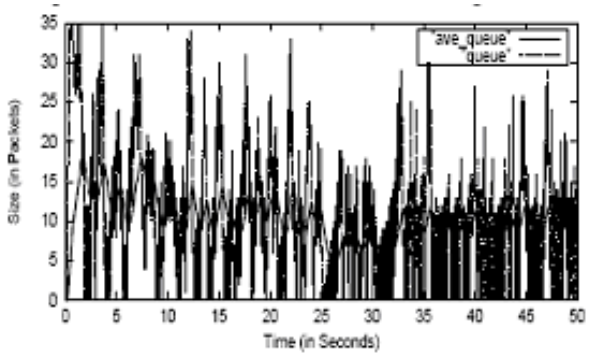
Şekil 4.5 deki grafik aynı benzetmenin Adaptive RED kullanılmışını gösterir. Uyarlanmış RED'de yine aynı 25. zamanda ortalama sıra uzunluğunda keskin değişim gösterir. Bunun yanında, kabaca 10 sn sonra, Adaptive RED ortalama sıra uzunluğunu hedeflenen 9 ile 11 paket arasındaki aralığa geri düşürür. Adaptive RED ile olan benzetmeler, adaptive olmayan RED'le yapılanlara göre biraz daha fazla işlem hacmine sahiptir. (% 93.1 yerine % 95.1), ortalama sıra uzunluğunda tamamında biraz daha düşüktür (13.4 paket yerine 11.5 paket) ve daha küçük bir paket düşürme oranı vardır. Adaptive RED'le yapılan benzetmelerde, ortalama sıra uzunluğu ve paket düşme olasılığı arasındaki ilişkinin max_p nin uyarlanmasıyla gösterilmesi mümkündür ve bundan dolayı trafik dinamiklerinde hazır bulunan ortalama sıra uzunluğununun korunmasını sağlar.

Şekil 4.6, 0. zamanda başlayan ve 25. zamanda biten yirmi yeni akışla ilgili benzetmeyi gösterir. Şekil 4.6 de uyarlanmamış RED ile yapılmış benzetme, ortalama sıra uzunluğundaki düşüş 25. zamandaki değişimin sıkışmasının seviyesi olarak gösterilir. Bu zamanda, paket düşme oranı uyarlanmamış RED'le benzetmenin ilk yarısının üzerinde % 9.7 ve ikinci yarısında % 8 dir.



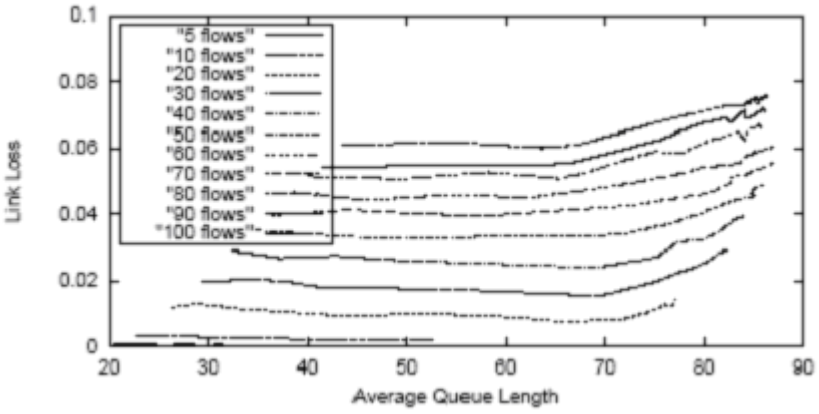
Şekil 4.6. *Sıkışıklıkda azalma ile RED.*

Adaptive RED ile yapılan benzetmede ortalama sıra uzunluğundaki benzer değişim şekil 4.7 da vardır, ama uyarlanmış RED ortalama sıra uzunluğunu hedeflenen seviyeye 10 sn içinde geri getirir. Adaptive RED le yapılan benzetmede diğer uyarlanmamış RED'le yapılan benzetmedeki işlem hacmi benzerdir (% 92.7 yerine % 93) , ortalama sıra uzunluğu da çok az daha küçüktür (12.4 paket yerine 11.1 paket)

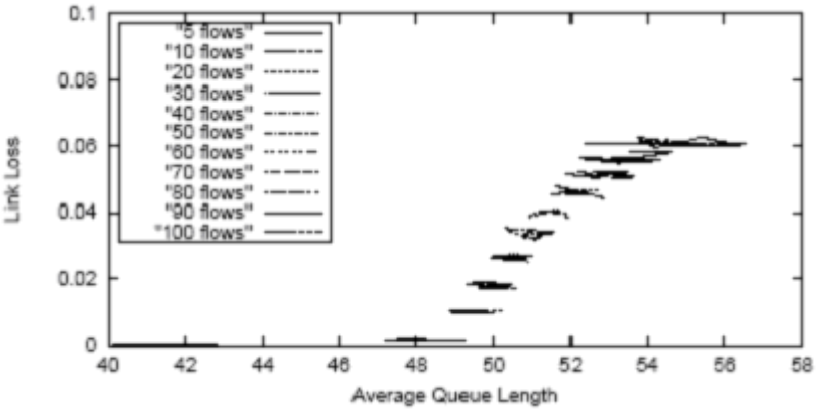


Şekil 4.7. *Sıkışıklıkda azalma ile Adaptive RED.*

Adaptive Red'le tüm benzetmeler, %98'den (100 akış ile) %100'e (5 akış ile) çıkan yüksek işlem hacmine sahiptir. Her bir akış sayısı, Adaptive RED'le aynı performansda, Adaptive RED olmayan sabit max_p seçebilir. max_p için bu sabit(static) ayarlama, benzetme senaryosunun bir fonksiyonu olabilir. Örneğin, 20 akışlı bir benzetme için Adaptive RED'in performansı, kabaca max_p değeri 0.07 ye ayarlanmış 100 akışlı Adaptive RED olmayan bir benzetmenin performansına karşılık gelebilir. Adaptive RED'in performansı, max_p si 0.2 ye ayarlanmış Adaptive olmayan RED'in performansına karşılık gelebilir [5].



Şekil 4.8. *RED ile gecikme – kayıp değişimi,* w_q*=0.00026.*



Şekil 4.9. *Adaptive RED ile gecikme – kayıp değişimi.*

Şekil 4.8 ve 4.9, şekil 4.1 ve 4.2 deki benzetmelerin paket düşme oranlarını göstermektedir. Belli bir akış kümesi için, kabaca RED ve Adaptive RED, aynı paket düşürme oranına sahip olduklarını gösterir, Adaptive RED, ortalama sıra uzunluğunu koruyarak max_{th} dan uzaklaşır, RED ortalama sıra uzunluğu max_{th} etrafında olduğu zaman daha yüksek oranda paket kaybetmekten kaçınır. Benzetmelerde, RED ve Adaptive RED'in doğruluk özelliklerinin benzer olduklarını gördük [5].

Bu benzetmeleri, belli bir sıradaki hat bant genişliklerini ve karışık web trafiklerini içeren belli bir sıradaki benzetmelerde , ECN ile ve ECN siz, byte ve paket birimlerinde ölçülmüş sıralarla, byte modundaki ve paket modundaki RED ile (paketi düşürüp düşürmeyeceğine karar verirken paketin büyüklüğünü byte olarak dikkate alır) keşfettik [5]. Tüm bu benzetmelerde Adaptive RED 'den aynı iyi performansı gördük.

4.3 Adaptive RED Algoritması

Burada gerçekleştirilen Adaptive RED ile ilgili kılavuz esas Adaptive RED ile aynıdır, ortalama sıra uzunluğunu min_{th} ve max_{th} arasında korumak için max_p uyarlanır. Buradaki yaklaşımın esas Adaptive RED den 4 farklılığı vardır.

- max_p nin uyarlanması sadece ortalama sıra uzunluğunu min_{th} ve max_{th} arasında korumak için değildir, ortalama sıra uzunluğunu yarım yol belirlenen hedef aralığı içinde min_{th} ve max_{th} arasında tutmak içindir.
- max_p, zaman çizelgesi üzerinde tipik bir RTT'dan daha büyüktür ve küçük adımlarla yavaşça uyum sağlar.
- max_p , [0.01 , 0.5] aralığında kalması için sınırlandırılmıştır.(yada eşiti [% 1 ,% 50])
- çarpansal olarak artan ve azalan max_p yerine AIMD (Addive-Incrase Multiplecative-Decrese) politikası kullanılır.

Adaptive RED algoritması şu şekildedir.

Her *aralık*da
  if (*avg* > *hedef ve* $max_p \leq 0.5$)
    max_p *artar:* $max_p \leftarrow max_p + \alpha$;
  elseif (*avg* < *hedef ve* $max_p \geq 0.01$)
    max_p *azalır:* $max_p \leftarrow max_p * \beta$;

Değişkenler :
avg: ortalama sıra uzunluğu

Sabit parametreler :
aralık : zaman ; 0.5 sn.
hedef : *avg* için hedef;
$[min_{th} + 0.4 * (max_{th} - min_{th}) , min_{th} + 0.6 * (max_{th} - min_{th})]$.
α : artış ; $min(0.01 , max_p / 4)$
β : azalma faktörü ; *0.9*

max_p nin adapte edilme kılavuzu yavaşça ve nadiren RED'in dinamiklerine izin verir - ortalama sıra uzunluğundaki değişimler için cevaplamada paket düşme olasılığının adapte edilmesi – daha küçük zaman çizelgesine hakim olmak için. max_p nin uyarlanması sadece uzun zaman çizelgelerinde ihtiyaç olduğunda çağrılır.

Adaptive RED'in sağlamlığı yavaş ve nadiren max_p nin ayarlanmasından gelir. Bu yavaş değişimin bedeli, şekil 4.5 ve 4.7 deki gibi, sıkışma seviyesindeki keskin bir değişimden sonra, max_p nin yeni değerine değişmesinden önce bazen on yada yirmi saniye almasıdır. Adaptive RED'in performansının sağlanması için iletim periyodu esnasında gereğinden fazla alçaltma yapılmayacaktır, üçüncü kılavuzumuz max_p yi sınırlayarak, [0.01 , 0.5] aralığında kalmasını sağlar.Bunlar, ortalama sıra uzunluğunun hedeflenen aralıkda olamaması ve ortalama gecikme yada işlem hacmi yavaşça kötüye gitmesi bile, RED'in tüm performansının, iletim periyodu sırasında hala kabul edilebilir olmasını sağlar [5].

Adaptive RED algoritması için en iyi veya en ,iyiye yakın demek istemiyoruz, fakat senaryoların geniş bir aralığında iyi çalışıyor görünüyor, ve bizde internetteki RED gerçekleştiriminde güvenle deploy edilebileceğine inanıyoruz [5]. max_p nin yavaş adaptasyonun sonucu olarak, Adaptive RED'in tasarımı, geniş alanlarda güçlü sonuçlar verir. Yukarıda belirtildiği gibi, iletim periyodunun bu yavaş adaptasyonun bedeli,

ortalama sıra uzunluğu hedeflenen bölgede olmadığı zaman sıkışma seviyesindeki keskin bir değişim sonrası. Adaptive RED bundan dolayı dikkatli bir şekilde pozisyonlandırılmıştır, AQM mekanizmasının spektrumunun sonunda sağlam, daha ince ayarlanmadan kaçınma amacı ile, spektrumun sonunda daha agresif daha kırılgandır [5].

Adaptive RED algoritması max_p yi adapte edebilmek için AIMD kullanır. MIMD(Multiplicative Increase Multiplicative Decrease) gibi diğer doğrusal kontrollerle de denedik, ama AIMD yaklaşımının daha daha güçlü olduğunu gördük [5].

Bu genel Adaptive RED algoritmasının tanımını tamamlar. Bu algoritmada gömülü seçenekler çeşitli parametreler için ayrıntılandırılmıştır. Şimdi bu seçeneklerin gerekçelerini kısaca anlatacağız.

4.3.1 max_p'nin Sınıflandırılması

max_p nin 0.5 üst sınırı iki taban üzerinde haklı çıkarılabilir. Birincisi, paket düşme oranını %50 den daha büyük RED'i optimize etmeye çalışmıyoruz. Ayrıca, çünkü RED'i hafif modda kullanıyoruz, bu ortalama sıra uzunluğunun min_{th} dan max_{th} a kadar değiştiğininde, paket düşme oranının 1 den max_p ye kadar değiştiğini, ve ortalama sıra uzunluğunun max_{th} dan max_{th} ın 2 katına kadar değiştiğinde, paket düşme oranının max_p den 1'e kadar değiştiği anlamına gelmektedir. Bundan dolayı max_p nin 0.5 e ayarlanması ile ortalama sıra uzunluğunun max_{th} dan max_{th} ın 2 katına kadar değiştiğinde, paket düşme oranı 0 dan 1 'e kadar çeşitlenir. Bu, paket düşme oranı % 50 nin üzerinde bile olsa bir dereceye kadar güçlü performans vermelidir [5].

max_p nin 0.01 alt sınırı, max_p nin arzulanan limit aralığında harekete geçirilir. Çok küçük paket düşme oranlı senaryolar için, max_p nin 0.01'e ayarlanmasıyla RED'in dürüstçe kuvvetini göstereceğine ve hiç birinin benzer şekilde ortalama sıra uzunluğunu hedeflenenden daha küçük nesneleştiremeyeceğine inanıyoruz [5].

4.3.2 α ve β Parametreleri

max_p için 0.01 den 0.50 ye artmaları için minimum 0.49/α aralıklarında aldığını not ettik; bu α ve *aralık* varsayılan paremetreleri için 24.5 sn dir. Benzer şekilde 0.50 den 0.01 e düşmeleri için max_p minimum *log 0.02/log β* aralıklarında alır; bu varsayılan parametrelemiz ile 20.1 sn dir. Verilen, bir sıkışma seviyesinden diğerine keskin bir değişim, 25 sn, bundan dolayı, ortalama sıra uzunluğunun hedeflenen aralığın dışında kalabildiği sırada, aralık üzerindeki üst sınır ve AQM in performansı bir dereceye kadar düşürülmüş olabilir [5].

α ve β nın önerilen değerlerinde, normal şartlar altında, max_p nin tek değişimi, ortalama sıra uzunluğunu hedeflenen aralığın üzerinden, altına taşıdığı yada tersi sonucunu çıkarmaz. Şimdi basitleştirmek için max_p, istikrarlı duruma uyarlandığı zaman, paket düşme olasılığı aynı kalır ve ortalama sıra uzunluğu *avg* basitce max_p nin yeni değerlerini eşleştirmek için kayar. Bundan dolayı, $p < max_p$ olduğunu varsayarsak, max_p α tarafından arttırıldığı zaman *avg*'nin $min_{th} + p/max_p\ (max_{th} - min_{th})$ dan $min_{th} + p/(max_p + \alpha)\ (max_{th} - min_{th})$ a düşmesi beklenir.

$$\frac{\alpha}{(\max_p + \alpha)} \frac{p}{\max_p} (\max_{th} - \min_{th}) \qquad (4.1)$$

Eğer sadece $0.2(max_{th} - min_{th})$ den daha düşükse, ortalama sıra uzunluğu hedef aralığının üstünden, altına tek bir aralıkda değişmemelidir. Bu $\frac{\alpha}{(\max_p + \alpha)} < 0.2$ seçimini önerir yada eşit olarak $\alpha < 0.25\ max_p$ .algoritmada gösterilen α nın varsayılan ayarı bu sınırlamaya uymak zorundadır.

Benzer şekilde, max_p .nin çarpansal düşüşünü (multiplicative decrease), ortalama sıra uzunluğunu hedeflenen aralığın altından üstüne, max_p.nin tek bir düzeltmesi sonrasında götürmesine sebep olmamasını kontrol etmeliyiz. Benzer bir analizde α,

$$\frac{p(1-\beta)}{(\max_p + \alpha)}(\max_{th} - \min_{th}) < 0.2(\max_{th} - \min_{th}) \qquad (4.2)$$

olduğu sürece ortalama sıra uzunluğu tek bir aralıkda hedeflenen aralığın altından hedeflenen aralığın üstüne değişmemelidir. Bu *(1-β)/p < 0.2* seçmeyi yada eşiti *β > 0.83*'ü seçmeyi önerir. Bu sınırlama *β* için 0.9 varsayılan değerimiz için sağlanır.

4.3.3 RED Parametreleri max_{th} ve w_q'nun Ayarlanması

Yukarıda tanımlandığı gibi Adaptive RED, RED'in max_p parametresi üzerindeki bağımlılığını kaldırır. RED için ihtiyaç duyulan parametre ayarlamalarını azaltmak için, max_{th} and w_q parametrelerinin otomatik ayarlanması için prosedürler tanımlar [5].

Otomatik modda, max_{th} , min_{th} ın üç katı olarak ayarlanır. Bu durumda hedeflenen ortalama sıra uzunluğu 2 x min_{th} etrafında merkezlenir, ve sadece RED'in min_{th} parametresi ile elde edilir.

Orjinal RED tanımında [4] verilen w_q ayarları için kılavuz, iletim sıra uzunluğu cinsinden RED tarafından birbirine uygun hale getirilir, ve cevaplamada şimdiki sıra uzunluğundaki değişme adımında tahmin edici tarafından zamana ihtiyaç vardır. [4] den, eğer sıra uzunluğu bir değerden diğerine değişirse, ortalama sıra için yeni değerin % 63 üne ulaşmada, $-1 < ln(1- w_q)$ paket ulaşır. Bundan dolayı $-1 < ln(1- w_q)$, ortalama sıra uzunluğu için tahmin edicinin "*zaman sabiti*" olarak atarız, bu zaman sabiti, kendi kendine değil, paket gelişlerinde tanımlanır.

NS simülatöründe varsayılan w_q 0.002 ye ayarlanır edilir; bu 500 paketin gelişinde bir zaman sabitine tekabül eder. Bunun yanında, 1 Gbps lık bir hat için, 500 byte'lık paketlerle, 500 paketin gelişi RTT'nin çok küçük bir parçasına karşılık gelir (100 ms'nin varsayılan rtt sinin 1/50.). Açıkca yüksek hızdaki hatlar için daha küçük w_q

gereklidir, böylece zaman sabiti RTT'nin dışında kalır. Aşağıdaki yaklaşımlarda [50,51] otomatik modda, w_q , hat bant genişliğinin bir fonksiyonu olarak ayarlanır.

Otomatik moddaki RED için, w_q verilen bir zaman sabitine bir saniyenin ortalama sıra uzunluğu tahmin edicisi için ayarlanır; bu on RTT'ye eşittir, RTT 100 ms olarak varsayılmıştır. Bundan dolayı, w_q yu,

$$w_q = 1 - \exp(-1/C) \qquad (4.3)$$

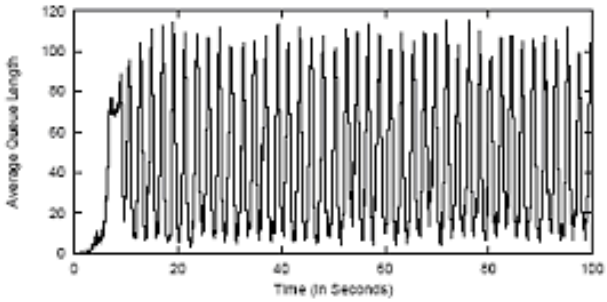
olarak ayarlanmıştır. C, paket/saniye olarak hat kapasitesidir, belirtilen varsayılan ölçünün paketleri olarak hesaplanır.

4.4 Benzetmeler

Bölüm 4.2 deki benzetmeler Adaptive RED'i önerir, çeşitli şartlarda, yüksek işlem hacmine ve düşük ortalama sıra gecikmesine ulaşmak için max_p nin sürekli uyarlanması ve w_q nun ayarlanması otomatik olarak yapılır. Bu bölümde Adaptive RED'in davranışındaki üç maddeyi daha yakından ele alacağız, salınımlar (oscillations), etkiler (effects) ve yönlendirme dinamiklerine cevap (response to routing dynamics).

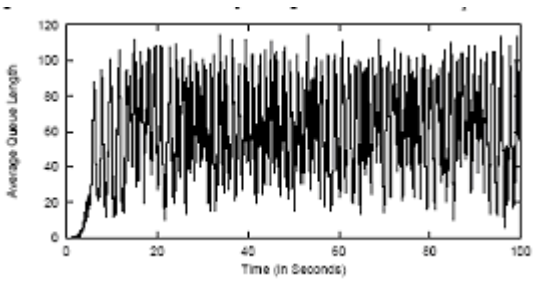
4.4.1 Salınımların Araştırılması

TCP'nin sıkışıklık kontrolünün geri besleme doğasından dolayı, sıra uzunluklarındaki salınımlar çok ortakdır. Bazı salınımlar çok zararlıdır, tüm işlem hacmini azaltır ve sıra gecikmesindeki değişimi arttırır; diğer salınımlar iyi huyludur ve işlem hacmine ve gecikmeye anlamlı etkileri yoktur. Şekil 11 den 14e kadar, herbiri ortalama sıra uzunluğunu gösterir, bir benzetme 100 uzun yaşamlı akış ile , her biri 250 ms lik RTT'lerle ve 15 Mbps'lık sıkışmış bir hat ile gösterilmiştir. Tüm akışlar, ECN ve 1000-byte data paketi kullanır. RED sıra yönetimi, $min_{th} = 20$ ve $max_{th} = 80$ e sahiptir.



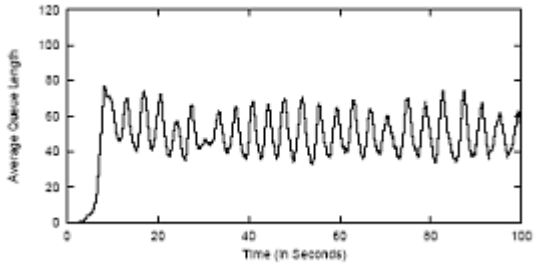
Şekil 4.10. *RED,tek yönlü uzun yaşamlı trafik, w_q=0.002 [5].*

Şekil 4.10 daki benzetme, RED'i kullanır, herbiri sıra uzunluğundaki salınımı cesaretlendiren üç faktörü vardır, (1) max_p için sabit (aşırı derecede küçük) bir değer; eşitlik (4.2) w_q için yüksek bir değer; ve eşitlik (4.3) uzun yaşamlı akışların tek yönlü trafiklerinin karışımı için basit bir trafik. Şekil 4.10, ortalama sıra uzunluğunda dramatik bir salınım gösterir, herbir salınımda, ortalama sıra uzunluğu min_{th} 'ın altına ve max_{th} 'ın üstüne gider. Bu, yüksek paket düşme oranını periyodu ile hiç paket düşmeyenin periyodu arasında salınımlara ve sonuç da azalmış işlem hacmi ve sıra gecikmesinde yüksek değişimlere liderlik eder. max_{th} nin aşılması, geniş paket düşmenin formunda doğrusal olmamaya maruz bırakır, kullanımda düşmeye karşılık gelir ve bu durumda min_{th} ın altında ortalama sıra uzunluğunu keskince düşürür. Ama ortalama sıra uzunluğu min_{th} 'ın altına düşerse, ortalama paket düşme olasılığı sıfır olur, ve akışlar yeniden, sonraki az RTT'leri üzerinde sıkışma pencerelerine yayılır, bu münasebetle salınımları devamlı tutarlar. Bu durumda RED % 90 hat kullanımına ve sıra gecikmesinde yüksek değişimlerin üstesinden gelir. Paket düşme oranı ECN kullanılsa bile % 3.5 civarındadır [5].



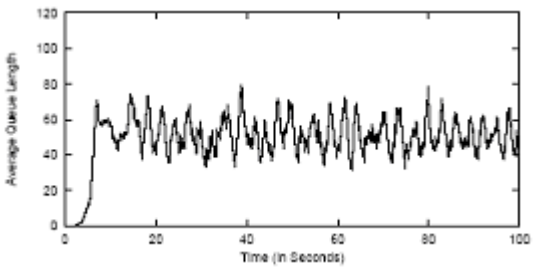
Şekil 4.11. *RED, zenginleştirilmiş trafik, w_q=0.002 [5].*

Şekil 4.11, böyle kötü huylu salınımların, ters yol trafiği ve web trafiği içeren daha gerçekci trafik karışımları ile titreşimi ciddi bir şekilde azaltır; kötü ayarlanmış ve Adaptive olmayan RED'le bile, salınımların en kötü etkilerinin çoğunluğu, hafifce daha gerçekci trafik kullanıldığında düşürülür[5].



Şekil 4.12. *Adaptive RED,tek yönlü uzun yaşamlı trafik, w_q=0.002 [5].*

Şimdi Adaptive RED'in daha düşük w_q değerleriyle max_p nin otomatik olarak adaptasyonunu nasıl sağladığını ve bu iki trafik senaryosundaki yolculuk ücretlerini dikkate alacağız. Şekil 4.12 de tek yönlü ve uzun yaşamlı trafik karışımının basit bir trafiğidir, Adaptive RED, kötü huylu salınımları ortadan kaldırmaya ve bunları iyi huylu salınımlara dönüştürmeye çalışır. 250 ms lik sabit RTT olmasına karşın, ne ters trafik nede web trafiği, Adaptive RED kullanımın % 96.8'e ulaşmasına ve ortalama sıra uzunluğu salınımını hedeflenen aralık içinde tutmaya çalışır, kayıp oranını ihmal eder (trafik ECN kullanır). Kötü huylu salınımlar, Adaptive RED ile eğer w_q için 0.002 nin daha geniş değerleri kullanılırsa kalıcıdırlar; max_p nin adapte edilmesi ve w_q için iyi bir değer seçilmesi, bu benzetmede kötü huylu salınımların ortadan kalkması için gereklidir [5].



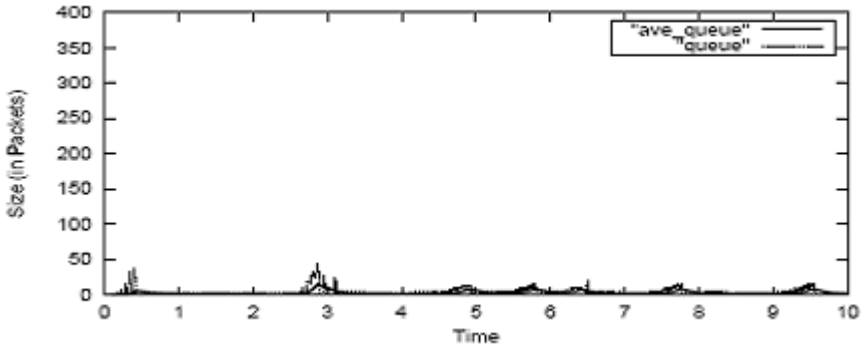
Şekil 4.13. *Adaptive RED, zenginleştirilmiş trafik, w_q=0.002 [5].*

Şekil 4.13 de, şekil 4.12'nin iyi huylu salınımlarını, web trafik ve ters trafik, daha gerçekci karışımı ile göstermektedir. Burada önceki şekle göre paketlerin hedef aralıkları içindeki ortalama sıra uzunluğunun değişimi daha düzensiz olarak gösterilmiştir. Bu durumda kullanım şekil 4.12'ye göre biraz daha yüksektir.

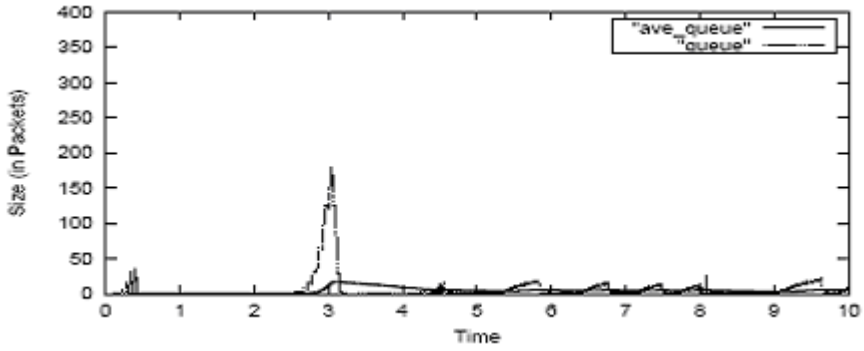
4.4.2 Sıra Ağırlığının Etkileri

Şekil 4.1, azalan işlem hacmi cinsinden w_q nun çok geniş değerleri için performans değerini gösterirken, bu bölüm w_q nun çok küçük değerleri için, artan ortalama gecikme cinsinden değerini gösterir.

Şekil 15'den 17'ye kadar, iki uzun yaşamlı TCP akışı ile basit bir benzetmenin sonucunu göstermektedir. Herbirinin RTT'si 45 ms civarındadır, 15 Mbps'lık bir hat üzerinde rekabet ederler. İkinci TCP akışı, 10 sn lik benzetme içerisinde 2.5 de başlar. İki TCP akışıyla ortalama sıkışma penceresi 85 paket civarında olmalıdır. Her üç benzetme de Adaptive olmayan RED kullanmıştır, ve sadece w_q değerleri farklıdır. Bu benzetmeler aynı zamanda w_q nun çok küçük değerleri için bedelinin birini gösterir, anlık sıra içerisinde geniş bir cevap için güçlendirilmiş olarak artış yavaşlamaya başlar.

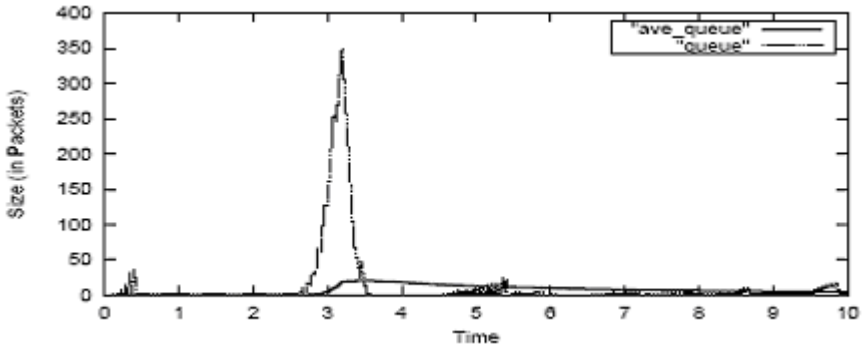


Şekil 4.14. *RED, 0.002'de çok geniş w_q, iki akış [5].*



Şekil 4.15. *RED, 0.00027'de w_q için otomatik ayarlar [5].*

Şekil 4.14'deki benzetmede, w_q 'nun 0.002 ve geniş değerleri için RED kullanmıştır. Benzetmeleri tamamında min_{th} *ve* max_{th} 'ı otomatik ayarlar, sonuçta min_{th} 19 pakete max_{th} da 3 min_{th} a ayarlanır. Şekil 4.14 o andaki ortalama sıra uzunluğunu gösterir. Ek olarak bu RED tarafından tahmin edilir. Her ne kadar ikinci TCP, yavaş başlaması istenen sıkışıklık penceresine ulaşmadan yavaşca keser (şekil 4.14), şekil 4.14 ve 4.15, bu senaryo için kabul edilebilir uygun iyi performansı gösterirler.



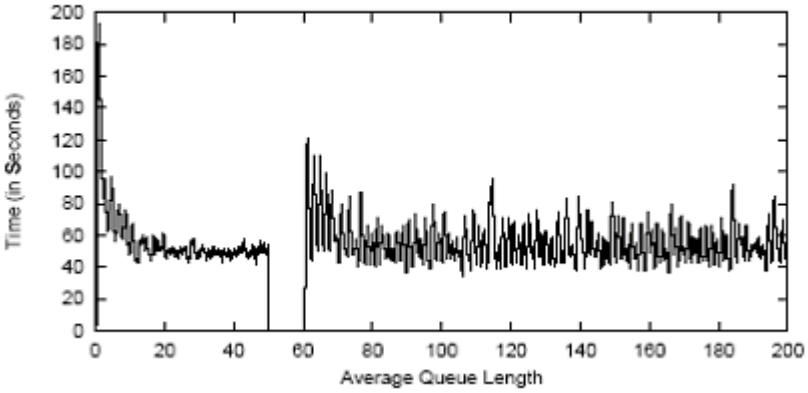
Şekil 4.16. *RED, 0.0001'de w_q çok küçük [5].*

Buna karşın, şekil 4.16, w_q 'nun çok küçük değerlere ayarlanmasının bedelini göstermektedir. Bu benzetmede w_q = 0.0001, sonuçda da sıkışmadaki ani artışların tespitinde RED yavaştır, ve 2.5. zamanda olan sıkışmayı, sırada 350 paket oluncaya kadar tespit edememektedir. Bu benzetmede, sıradaki keskin artış, tek bir yüksek bant genişliğinin yavaş başlamasından olmaktadır, ama artış, ani kalabalık yüzünden olabilmektedir, bir yönlendirme başarızlığı yada bir denial of service saldırısı vb. Bu benzetme geniş bir tampon bellek size çalışır, ve 350 paketin depolanmasına izin verir.

Eğer tampon bellek size daha küçük olsaydı, benzetme basitce tipik drop-tail sıra yönetimi davranışına dönebilirdi,(bir verinin penceresinden fazla sayıda paketin düşebildiği). Bir dizi senaryo keşfettik ve hemen hemen tüm durumları, hatta kararlılık-durum senaryolarını bile inceledik, w_q 'nun eşitlik (4.1) de önerilenden daha küçük bir değerini kullandığımızda hat kullanımı zarar görmektedir.

4.4.3 Yönlendirme Değişimlerinin Benzetmesi

Bu bölüm, kısaca Adaptive RED'in iletim davranışını yönlendirme değişimleri yüzünden, yükünde keskin değişimler olan çevreleri inceler. Şekil 18, benzetmede ortalama sıra uzunluğunu zamanın bir fonksiyonu olarak gösterir, 50 sn den 60 sn ye çıkış hattı mevcut olmayabilir. Benzetme topolojisi, daha düşük öncelikli ama sadece yarım hat kapasitesi içeren alternatif bir yol içerir. Böylece TCP bağlantıları hat kesintisi sırasında da paket göndermeye devam edebilir. Hat geri geldiği zaman, tüm yük esas hatta geri kaydırılır. Hat kullanımı 10 sn lik periyodda %88.3 e ulaşır, hemen tamir eder ve sonraki 10 sn içerisinde de % 96.1'e ulaşır. Bundan dolayı, bu senaryo Adaptive RED'in iyi dinamik özelliğini gösterir [5].



Şekil 4.17. *Yönlendirme değişikliği ile ortalama sıra uzunluğu değişimi [5].*

4.5 İşlem Hacmi ve Gecikme Arasındaki Değişimler

Verilen Adaptive RED algoritması ve max_{th} ve w_q nun otomatik olarak ayarlanması bu bölümde daha önce tanımlandı. Geriye sadece tanımlanacak kritik parametre olarak hedeflenen ortalama sıra uzunluğu kaldı. Adaptive RED ortalama sıra uzunluğunu min_{th} ‘ın iki katı olarak sürdürür; bundan dolayı verilen bir hedef için ortalama sıra uzunluğu, min_{th} ‘ın ayarlanması doğrudur. Zor kısmı ise istenen ortalama sıra uzunluğunun elde edilmesidir [5].

Bir yönlendirici için en iyi ortalama sıra uzunluğu, işlem hacmi ve gecikme arasındaki takas (trade off) oranın bir fonksiyonudur. Bu takas, politikanın gerekli bir sorusudur. Bunun yanında, işlem hacmi ve gecikme arasındaki bu takaslar, kümelenmiş trafiğin karakteristiğinin bir fonksiyonudur. Bundan dolayı tek yönlü trafikli senaryolar, uzun yaşamlı akışlar, kısa RTT’ler ve istatistiksel çoklamanın yüksek seviyesi, hem çok yüksek işlem hacmine hemde çok düşük gecikmeye izin verir, senaryolar daha yüksek patlamalarla iken, sonuçlar iki yönlü trafikden ve web senaryosundan istatistiksel çoklamanın düşük seviyeleri ile işlem hacmi ve gecikme arasında biraz daha zor takas gerektirir [5].

En iyi olma sorununun arkasında ayrılırken, sıradaki Jacobsen [50] de ve benzetme scriptleri [52] de, otomatik modda, min_{th} ı hat bant genişliğinin bir fonksiyonu olarak ayarlarız. Yavaş ve makul hat hızları için, min_{th} ın beş pakete ayarlanmasının iyi çalıştığını bulduk, böylece bunu, otomatik modda min_{th} ın alt sınır olarak kullanabiliriz. Yüksek hızdaki bir hat için, on paketlik ortalama sıra uzunluğu, gecikme bantgenişliği ürününe oranla çok küçüktür, ve sonuçta işlem hacmi çok şiddetli kaybolmaktadır.

İşlem hacmi ve gecikme arasındaki güvenilir bir takas için yönlendirici sıra gecikmesi, varsayılan değeri olarak 100 ms kullanılan, uçtan uca RTT’nin sadece küçük bir parçası olabilir. $min_{th} = delay_{target}C / 2$ ayarlanması hedeflenen ortalama sıra gecikmesi, $delay_{target}$ ı sn cinsinden verir. C paket/sn cinsinden hat kapasitesidir. Otomatik

modda $delay_{target}$ ı 5 ms olarak kullanıyoruz, min_{th} ı $Max[5, delay_{target}C / 2]$ paket olarak ayarlıyoruz bu dönüşüm mintreshi 100Mbps bir hat için 12.5 pakete ayarlar.

BÖLÜM 5

RED DİNAMİKLERİ VE KARARLILIK KONTROLÜ

Bilindiği gibi TCP/RED çılgın gibi salınım yapabilir ve bu salınımı RED parametrelerini ayarlayarak düşürmek oldukça zordur [21,22]. AIMD stratejisi TCP Reno tarafından çalıştırılır (ve onun değiştirilmiş hali olan NewReno ve SACK) ve gürültü trafik gibi etkin bir şekilde TCP tarafından kontrol edilemez. Şüphesiz bu salınıma katkısı vardır. Son çıkan modeller örneğin [10],[49], bunun, protokolün kaçınılmaz bir sonucu olduğunu belirtmektedirler. Bu bölümde, daha çok kanıtla bu görüş desteklenmiştir. TCP/RED salınımlarının sadece AIMD araştırmalarından ve trafik gürültüsünden kaynaklanmadığını ama temel olarak kararsızlığından kaynaklandığını kanıtlamaya çalışılmıştır. Salınımın AIMD bileşenin dışındaki düzeltmeden sonra ortalama davranışı küçük rastgele dalgalanmalarla salınmaz olabilir (protokol kararlı olduğu zaman), yada rastgele dalgalanmalardan daha geniş genliğin sınır devirlerini gösterir (kararsız olduğu zaman). Dahası bu niteliksel davranış, geniş miktarda gürültülü trafik olduğu zaman bile ve hatta kaynaklar farklı gecikmelere sahip olduklarında bile kalıcıdır. Sonuç olarak kararlılığı büyük ölçüde TCP/RED 'in dinamiklerinden elde edilir.

Bu TCP/RED'in kararlılık karakterini motive eder. İlerleyen bölümlerde genel bir TCP/RED'in doğrusal olmayan bir modelini geliştirilmiştir. Bu modelin dengeli yapısı kaynak [25]'de sıkışıklık kontrolünün toplu kaynak özelliklerini en yükseğe çıkarabilmek için çeşitli TCP/AQM'lerin başlıca ikili algoritmaların internet üzerinde dışarıya taşınmasıyla analiz edilmiştir. Burada, model etrafında dengeli bir şekilde doğrusallaştırılmasıyla yerel kararlılığı gösterilmiştir. Bu doğrusal model tek hattı kaynak [26]'nın kaynak modeli olarak genelleştirir. Bu modelin geçerliliği benzetmelerle ve TCP/RED'in kararlı bölgeside şekillerle gösterilmiştir. Karışık kaynaklarla tek bir hattın özel durumu için yeterli kararlılık şartlarını elde ettik. Bu gecikmeler artarsa yada hat kapasitesindeki artış daha fazla göze batarsa TCP/RED'in kararsız olduğunu gösterir.

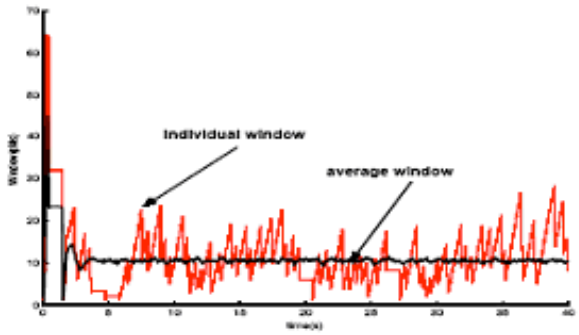
TCP tarafından tanıtılan kazanç, tek bir hat durumunda benzer kaynaklar tarafından paylaşılır, bant genişliği gecikme ürününün karesine orantılıdır ve tersine kaynakların sayısıyla orantılıdır. Böyle bir yüksek kazanç, gecikme yada kapasite yüksek olduğu zaman kararsızlığa sebep olur ve oldukça zor RED tarafından yeri doldurulur. RED parametreleri kararlılığı arttırmak için ayarlanabilir ama dinamik olarak ayarlansa bile sadece geniş sıra değerinde olmalıdır.

Bu önerilere göre gelecekteki ağlar için kapasite geniş olacağından tam uygun değildir. Bu bölümde, kaynak [27]'de geliştirilmiş basit bir sıkışıklık kontrol algoritması sunulmuştur, bu merkezsiz bir şekilde kaynaklar ve hatlar tarafından belirtilmiştir ve kararlıdır. Bu, rastgele seçilen gecikme, kapasite, yük ve yönlendirme için doğrusal kararlılığın devamını sağlar. Dahası, ihmal edilebilir sıralarla, dengeli bir şekilde, yüksek ağ kullanımını başarır. İletim cevabı gibi performans kaybı olmadan başarılan, bunların yararlarını gösteren temel benzetmeleri sunulmuştur.

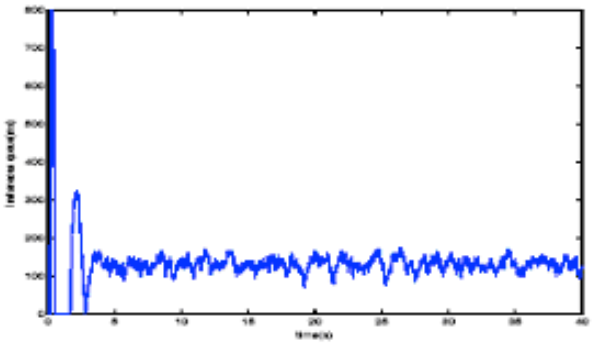
5.1 TCP/RED Salınımları

AIMD'nin, trafik gürültüsünün ve gecikmelerin karışık olmasının ortalama pencere ve anlık sıra üzerindeki etkisi nedir ? Bu bölümde protokol kararsızlıkları ile etki sınırlarını karşılaştırmalı olarak göstereceğiz.

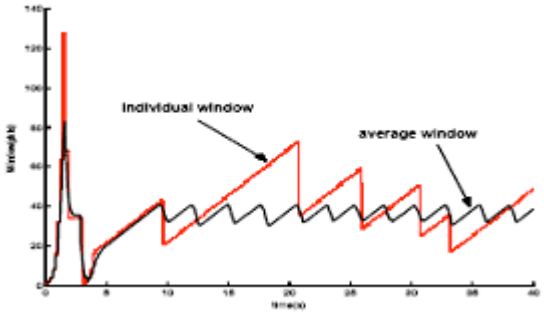
ns-2 simülatöründe, daralan kısım 9 paket/ms olarak simüle edilmiştir (sabit paket ölçüsü = 1000byte). Hat 'byte' modunda ECN işaretlemeli (örneğin doğrulanan paketler ihmal edilebilir bir olasılıkla işaretlenir) RED çalıştırır. RED parametreleri, max_p = 0.1, min_{th} = 50 paket, max_{th} = 550 paket, ağırlık sıra ortalaması için $\alpha = 10^{-4}$ dür. Hat 50 devamlı FTP kaynağı tarafından paylaşılmaktadır. Benzetme hem tek yönlü hemde iki yönlü trafikle çalıştırıldığında, davranışı çok benzerdir. Şekil 5.1 ve 5.2 çift yönlü trafiği göstermekte, şekil 5.3 ise tek yönlü trafiği göstermektedir. İnternetteki ölçülerin % 85'inde RTT 15-500ms aralığındadır. Benzetmeler gecikmelerle bu aralıkda gösterilmiştir.



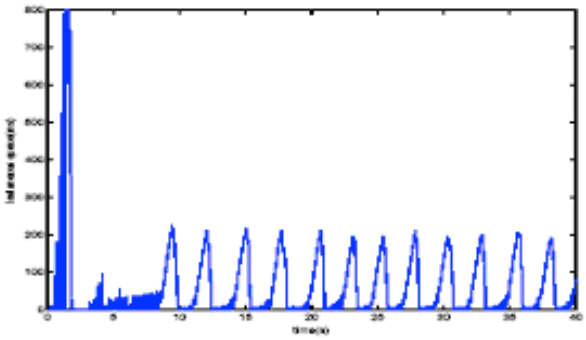
(a)Pencere (Gecikme = 40ms)



(b)Sıra (Gecikme = 40ms)



(c)Pencere (Gecikme = 200ms)



(d)Sıra (Gecikme = 200ms)

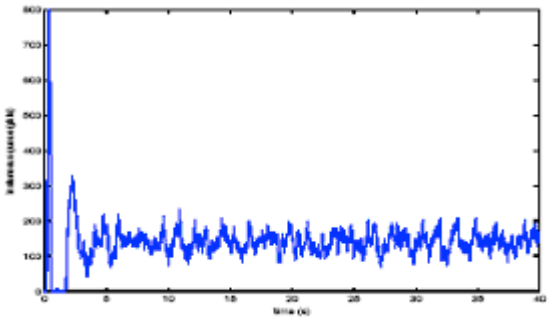
Şekil 5.1. *Gürültüsüz pencere ve sıra izleri. Benzetme parametreleri : 50 kaynak, kapasite = 9 paket/ms, RED = (0.1,50,550,10^{-4}) , byte mod ile işaretleme, iki yönlü trafik.*

Şekil 5.1 iki durumun sonucunu vermektedir, bunlar bağlantıların benzer gidiş dönüş gecikme yayılmasına sahip olduklarını ve her iki yönde de trafik oluşturulmasıdır. Şekil 5.1(a) kişisel pencere (açık eğri) ve ortalama pencere (koyu eğri) ortalama 50 kaynak üzerinde, her iksinide zamanın fonksiyonu olarak göstermektedir. Gidiş dönüş gecikme yayılması küçük (bu durumda 40ms) olduğu zaman bunlar tipik izleridir. Reno'nun AIMD'si yüzünden oluşan salınımlar kişisel pencerede göze çarpar ama ortalama pencerede görünmez. Beklendiği gibi sıranın kişisel pencereyi ortalamasından, ayrıca rastgele küçük dalgalanmalarla düzelmiş bir iz görünür, şekil 5.1(b) de görüldüğü gibi. Protokolün ortalama davranışını düşündüğümüzde, bu durum için kararlıdır (salınımsız).

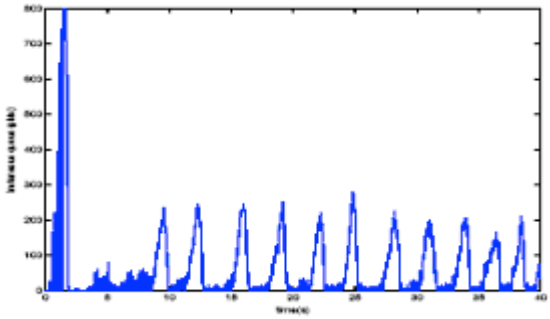
Şekil 5.1(c) ve (d), gidiş dönüş gecikme yayılması (round trip propagation delay) 200ms'ye çıkarıldığı zaman ki karşılık gelen pencereleri ve sırayı göstermektedir. Burada kişisel pencerenin daha geniş bir genlikle salınımından daha önemli ortalaması deterministik devir sınırlarını göstermektedir. Bu ayrıca sıra izinide göstermektedir. Protokolün kararsız bir usulde olduğunu söyleyebiliriz.

TCP/RED tarafından etkin bir şekilde kontrol edilemeyen gürültü gibi gecikme ve kayıplara karşı duyarlı trafiklerinin etkisi nedir ? Niteliksel olarak anlayabilmek için, iki yönlü 50 devamlı FTP bağlantısına kısa http kaynakları ekleriz. Her bir http kaynağı

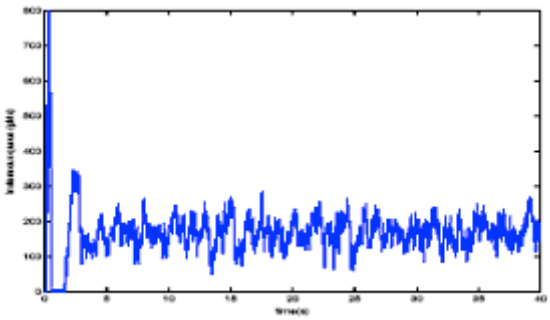
gideceği yere tek bir paket isteği gönderir, daha sonra çarpansal olarak dağıtılmış (demek istediğimiz 12 tane 1KB paket) büyüklükteki yanıtlanır. Kaynak tamamen veriyi aldıktan sonra rastgele bir süre bekler, 500msec ile çarpansal dağıtılır ve işlemi tekrar eder. İstek ve cevapda TCP bağlantısı üzerinden taşınmıştır. İki sümülasyon kümesi yürütüldü, birincisi 60 http kaynağı ile % 10 gürültü oluşturuldu (örneğin devamlı FTP kaynakları daralan hat kapasitesinin % 90nını kapladı), ikinci küme ile 180 http kaynağı %30 gürültü oluşturdu. Sıra izleri, yayılma gecikmesi 40ms ise kararlı, 200ms ise kararsızdır, bunlar %10 luk bir gürültü yoğunluğu ile sırasıyla şekil 5.2(a) ve (b) de gösterilmiştir. Şekil 5.2(c) ve (d) ise gürültü yoğunluğu %30'dur. Sıra ve ortalama pencere'nin davranışları protokolün kararlılığı ile baskınlaştırılmıştır. Kararlı yönetimde (40 ms gecikme), gürültü trafiği ortalama sıra uzunluğunu yavaşça arttırır. Bu işaretleme olasılığını arttırır ve FTP kaynağının ortalama penceresini düşürür.



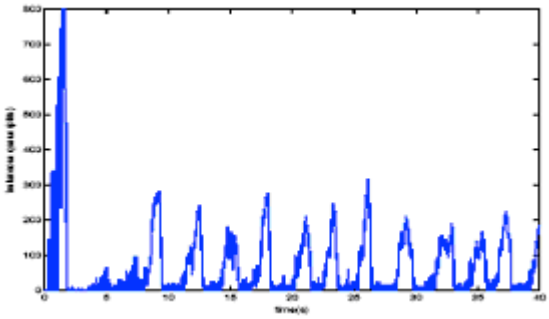
(a) Sıra (gecikme=40ms, %10 gürültü)



(b) Sıra (gecikme=200ms,%10 gürültü)



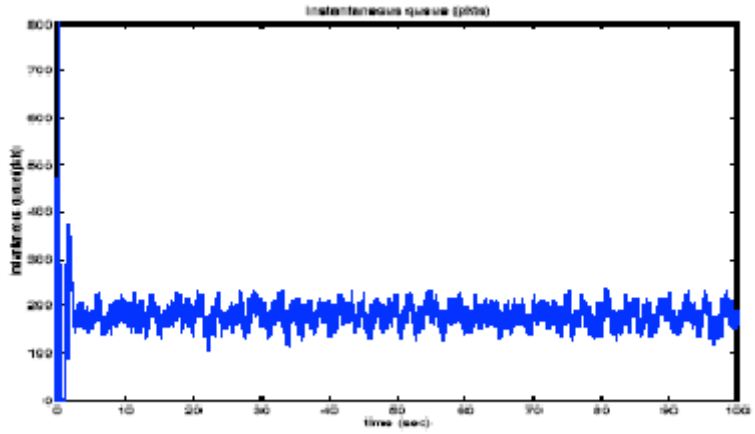
(c) Sıra (Gecikme=40ms, %30 gürültü)



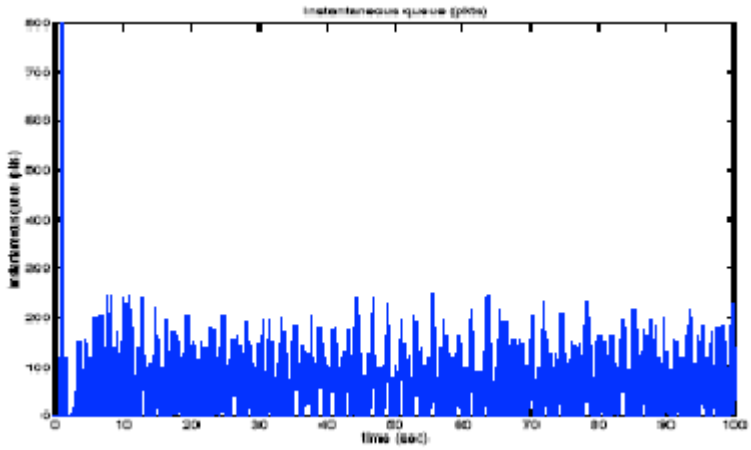
(d) Sıra (Gecikme=200ms, %30 gürültü)

Şekil 5.2. *Gürültüsüz sıra izleri. Benzetme parametreleri : 50 kaynak, kapasite = 9 paket/ms, RED = (0.1,50,550,10^{-4}) , byte mod ile işaretleme, iki yönlü trafik.*

Önceki tüm benzetmeler, benzer yayılma gecikmeli kaynaklar içindir. Kaynaklar farklı gecikmelere sahip olduğu zaman, dinamik davranışları çok değişir mi ? Önceki deneyleri gürültüsüz ve gecikme aralıkları 1ms artışda 40ms'den 64 ms'ye kadar olan, 50 devamlı tek yönlü bağlantılar ile tekrarlayacağız. Tüm gecikmeler geniş bir aralıkda aşağı veya yukarı ayarlandığı zaman ki dinamik davranışlarını öğreneceğiz. Gecikmelerin benzemesi durumunda davranışlarda, daha fazla sıra salınımları ile niteliksel olarak benzer. Şekil 5.3(a) anlık sırayı göstermektedir. Ayarlama(scaling) faktörü 0.3 (gecikme aralığı 0.3ms'den(40) 0.3ms'ye (64)),ortalama gecikme 15.6ms dir. Şekil 5.3(b) ayarlama faktörü 4 ve ortalama gecikme 208ms olduğundaki sırayı göstermektedir.



(a) Sıra (gecikme 12'den 19'a kadar)



(b) Sıra (gecikme 160'den 254'e kadar)

Şekil 5.3. *Karışık gecikmelerle sıra izleri. Benzetme parametreleri : 50 kaynak, kapasite = 9 paket/ms, RED = (0.1,50,550,10^{-4}) , byte mod ile işaretleme, tek yönlü trafik.*

Kararsızlık üç potansiyel probleme sebep olur. Birincisi, kaynak oranında ve gecikmedeki stresi arttırır ve bazı uygulamalar için zararlı olabilir. İkincisi, kısa süreçli bağlantıları hükmüne alır, bu bağlantılar tipik olarak gecikme ve kayıp, gereksiz gecikme ve kayıp için hassasdır. Son olarak eğer sıralar boş ve dolu arasında sıçrama yaparsa, hatların kullanımına yol gösterir.

Bundan dolayı protokol kararlılığı, TCP/RED'in dinamiklerini büyük ölçüde belirler. Şimdi TCP/RED kararlı olduğu zamanı karakterize edeceğiz.

5.2 Dinamik Model ve Kararlılık

Bu bölümde kararsızlığın başlangıcını tahmin etmek için geliştirilen TCP/RED'in bir modeli kullanılmıştır. Doğrusal olmayan bir modelle başladık ve denge özelliği hakkında bazı görüşler belirttik, sonra denge etrafında modeli doğrusallaştırdık. Doğrusal modelin ns-2 simülatörü ile doğruluğunu sağladık ve TCP/RED'in kararlı bölgelerini şekillerle gösterdik. Son olarak karışık kaynaklarla tek bir hattın özel durumu için kararlılık şartı elde edeceğiz.

5.2.1 TCP/RED'in Doğrusal Olmayan Modeli

Bir ağ L hatlarının (sınırlı kaynaklar) kümesi olarak sınırlı kapasitelerle $c = (c_l, l \in L)$ modellenir. i ile indislenmiş I kümesindeki, N kaynağın kümesi tarafından paylaştırılmıştır. Her bir kaynak i, L_i hatlarının $C = L$ kümesini kullanır. L_i kümesi LxN yönlendirme matrisini belirtir.

$$R_{li} = \begin{cases} 1 & l \in L_i \\ 0 & aksitakdirde \end{cases}$$

Her bir l hattıyla ilişkilendirilmiş olan işaretlenme olasılığıdır $p_l(t)$ t zamanında ve her bir s kaynağı ve penceresi $w_i(t)$ t zamanındadır. TCP Reno $w_i(t)$ nin nasıl ayarlacağını ve AQM'de $p_l(t)$ nin nasıl güncellendiğini tavsiye eder. Birlikte geciken geri besleme sisteminin bir formudurlar ve internet üzerinde azami dereceye çıkarma problemini çözmek için dağıtılmış ikili esas (primal-dual) algoritma dışarı taşınır [26,30].

i kaynağının t zamanında bir RTT tanımlanır.

$$\tau_i(t) = d_i + \sum_l R_{li} \frac{b_l(t)}{c_l} \qquad (5.1)$$

d_i gidiş dönüş yayılma gecikmesidir ve $b_l(t)$, l linkinde t zamanındaki geciktirmedir. kaynak i'nin oranı $x_i(t)$ t zamanında ,

$$x_i(t) := \frac{w_i(t)}{\tau_i(t)} \qquad (5.2)$$

l hattındaki tüm akış oranı

$$yl(t) = \sum_i R_{li} x_i(t - \tau_{li}^f(t)) \qquad (5.3)$$

$\tau_{li}^f(t)$ kaynak i den l hattına ilerlemiş gecikmedir. Uçtan uca işaretleme olasılığı i kaynağında $q_i(t) = 1 - \prod_{l \in L}(1 - pl(t - \tau_{li}^f(t)))$ olarak gözlemlenir. $\tau_{li}^b(t)$, l hattından i kaynağına geriye doğru olan gecikmedir. Tüm t ler için $p_l(t)$ küçük olarak düşünülür, böylece uçtan uca olasılık yaklaşık olarak şöyledir.

$$q_i(t) := \sum_i R_{li} pl(t - \tau_{li}^b(t)) \qquad (5.4)$$

Geciken hat olasılıklarının toplamıdır. İleriye ve geriye doğru olan gecikmeler RTT boyunca,

$$\tau_i(t) = \tau_{li}^f(t) + \tau_{li}^b(t) \qquad (5.5)$$

ile ilişkilidir, her $l \in L_i$ dir.

Şimdi TCP Reno ve RED modellerine bakacağız. TCP Reno'nun AIMD algoritmasına odaklanıyoruz. *t* zamanında, *i* kaynağının iletim oranı $x_i(t)$ paket/sn; bundan dolayı ACK'leri $x_i(t - \tau_i(t))$ oranında alır, her paketin doğrulandığı varsayılır. Bu ACK'lerin bir parçası $(1 - q_i(t))$ pozitifdir, her bir artış pencereyi $w_i(t)$, $1 / w_i(t)$ kadar arttırır; bundan dolayı $w_i(t)$ penceresi artar, ortalamada,

$$x_i(t - \tau_i(t))\ (1 - q_i(t))/w_i(t)$$

oranındadır. Benzer şekilde negatif ACK 'ler

$$x_i(t - \tau_i(t))\ q_i(t)$$

ortalama oranında alınırlar, herbiri pencereyi yarıya indirir ve bundan dolayı $w_i(t)$ penceresi $x_i(t - \tau_i(t))\ q_i(t)w_i(t)/2$ oranında düşer. Bundan dolayı pencere Reno altında

$$\dot{w}_i(t) = x_i(t-\tau_i(t))(1-q_i(t))\frac{1}{w_i(t)} - x_i(t-\tau_i(t))q_i(t)\frac{w_i(t)}{2} \qquad (5.6)$$

formülüne göre gelişir. $q_i(t)$ eşitlik (5.4) de verilendir.

RED'i modellemek için $b_l(t)$ anlık sıra uzunluğunu t zamanında $b_l(t) > 0$ olduğu zaman gelişir.

$$\dot{b}_l(t) = y_l(t) - c_l \qquad (5.7)$$

$y_l(t)$, eşitlik (5.3) de verilen akış oranıdır ve c_l hat kapasitesidir. Ortalama sıra uzunluğu $r_l(t)$ olarak belirtilir. bu şu formüle göre güncellenir,

$$\dot{r}_l(t) = -\alpha_l c_l\left(r_l(t) - b_l(t)\right) \qquad (5.8)$$

bazı sabitler için $0 < \alpha_l < 1$ dir. Verilen ortalama sıra uzunluğu $r_l(t)$ işaretleme olasılığışu şekilde verilir,

$$p_l(t) = \begin{cases} 0 & r_l(t) \le \underline{b}_l \\ \rho_l r_l(t) - \rho_l \underline{b}_l & \underline{b}_l < r_l(t) < \bar{b}_l \\ \eta_l r_l(t) - (1 - 2\bar{p}_l) & \bar{b}_l(t) \le r_l(t) < 2\bar{b}_l \\ 1 & r_l(t) \ge 2\bar{b}_l \end{cases} \qquad (5.9)$$

$\underline{b}_l, \bar{b}_l, ve, \bar{p}_l$ RED parametreleridir, ve

$$\rho_l := \frac{\bar{p}_l}{\bar{b}_l - \underline{b}_l} \quad \text{ve} \quad \eta_l := \frac{1 - \bar{p}_l}{\bar{b}_l}$$

Özetle, TCP/RED eşitlik (5.6 - 5.9) tarafından modellenmektedir ve ağ boyunca birbirlerine bağlı olmaları eşitlik (5.3 - 5.4) tarafından modellenmektedir.

Hatırlatmalar :

1. Kaynak [5] ve kaynak [7] den, TCP/RED modelini eşitlik (5.6 - 5.9) yorumladık ve diğer TCP/AQM modelleri , toplu kaynak özelliğini internet üzerinde en fazla yapabilmek için dağıtılmış esas-ikili algoritmaları dışarıya taşındılar. Kaynak oranı $x_i(t)$ yi, TCP tarafından tekrarlanan esas değişkenler olarak ele alırsak ve işaretleme olasılığı $p_l(t)$ yi AQM tarafından tekrarlanan ikili değişkenler (Lagrange multipliers) olarak ele alabiliriz. Farklı protokoller, farklı güncelleme kurallarına karşılık gelir ve farklı araç fonksiyonu U yu en fazla yapar. TCP Reno nun araç fonksiyonu şöyle elde edilir.

$$U_i(x_i) = \frac{\sqrt{2}}{\tau_i} \tan^{-1}\left(\frac{\tau_i x_i}{\sqrt{2}}\right)$$

buna karşılık TCP Vegas [31] ise,

$$U_i(x_i) = \alpha \log x_i$$

Verilen ağ topolojjisi R , hat kapasitesi c , ve TCP aracı (utility) U_i, buradan basit bir konveks programın çözülmesiyle, ilgisi olan her denklik özelliğini elde edebiliriz. Bunlar işlem hacmi, kayıp, gecikme farklı TCP protokollerinin etkileşimi ve ayrılmış denge oranının doğruluğudur.

2. Reno'nun birçok gerçekleştirimi yada değişik biçimleri, herbir RTT'de en az bir kere pencereyi ikiye böler. Bu durumda, eşitlik (5.5) deki çarpımsal azalma (multiplicative decrease) terimi $-q_i(t)w_i(t) / 2\tau(t)$ ile yer değiştirilir. Buradaki tüm benzetmelerde, işaretleme olasılığı çok küçüktür , bir RTT'de birden çok işaretlenme olasılığı ihmal edilmiştir. Bundan dolayı, çarpansal azalmanın iki modeli arasındaki farklılık ihmal edilebilir.

5.2.2 TCP/RED'in Doğrusal Modeli

TCP/RED'i eşitlik (5.6 -5.9)'da denklik etrafında kararlılığı ile ilgili çalışabilmek için doğrusallaştırdık. Birçok basitleştirici varsayım yaptık. Birincisi yönlendirme matrisi R yi tüm satır sıralarını dolu olarak varsaydık böylece tek bir denklik kaybolma olasılığı vektörü p vardır (Lagrange multiplier). İkinci olarak, denklik işaretleme olasılığını tam olarak pozitif olan sadece daralan(bottleneck) hatların, modelde içerildiğini varsaydık. Dahası sistemin $\underline{b}_l < r_l(t) < b_l$, bölgesinde işlediği düşünülür, böylece işaretleme olasılığı ortalama sıra uzunluğunda afine edilmiş olur, $p_l(t) = p_l(r_l(t) - \underline{b}_l)$. Son olarak , gidiş dönüş gecikmelerindeki değişikliklerde, zamanda anahtar varsayım yapmıştık.

Gidiş dönüş gecikmeleri iki yerde görülür; birincisi pencere $w_i(t)$ ve oran $x_i(t)$ arasındaki ilişkide, eşitlik (5.2) de belirtildiği gibi ve ikinci olarak, akış oranı $y_l(t)$ nin zaman argümanında, eşitlik (5.3) de ifade edildiği gibi ve uçtan uca işaretleme olasılığı $q_i(t)$ de, eşitlik (5.4) de ifade edildiği gibi. ilk üründe anlık sıra gecikmesinin dahil olması, eğer sıra gecikmesi yok sayılırsa veya sabit varsayılırsa niteliksel olarak farklı bir

modeldir. Anlamı şudur ki, sıra bir toplayıcı değildir ama daha karmaşık dinamikleri vardır; eşitlik (5.11)'e bakınız. Teorem 2'nin ispatı, bu dinamiğin TCP/RED'in kararlılığı için kritik olduğunu gösterir. Sonuçdaki doğrusal model benzetmeyle eşleşir, eğer sıra gecikmesi sabit varsayılırsa, farkedilir bir şekilde daha iyidir. İkincideki zaman değişimli gecikmeler doğrusallaşmayı zorlaştırır, ve denk değeri(denklik sıra gecikmesini içerir) ile yer değiştirilir. Bundan dolayı zaman değişimli gecikmeleri eşitlik (5.1) ve eşitlik (5.2) de kullanırız, ama yaklaşık gecikmeler $\tau_i(t), \tau_{li}^f(t), \tau_{li}^b(t)$ eşitlik (5.3) ve eşitlik (5.4) deki denk değerleridir.

Bu varsayımlarla, Reno/RED'i tek denklik etrafında doğrusallaştırdık. Eşitlik (5.5)'den Reno,

$$\dot{w}_i(t) = \left(1 - \sum_l R_{li} p_l(t - \tau_{li}^b)\right) \frac{w_i(t-\tau_i)}{\tau_i(t-\tau_i)} \frac{1}{w_i(t)} - \frac{1}{2} \sum_l R_{li} p_l(t - \tau_{li}^b) \frac{w_i(t-\tau_i) w_i(t)}{\tau_i(t-\tau_i)}$$

olur. Doğrusallaşma sonrası ürünler,

$$\dot{w}_i(t) = -\frac{1}{\tau_i q_i^*} \sum_l R_{li} p_l(t - \tau_{li}^b) - \frac{q_i^* w_i^*}{\tau_i} w_i(t)$$

burada $q_i^* = \sum_l R_{li} p_l^*$ uçtan uca olasılığın dengesidir, ve $w_i^* = x_i^* \tau_i$ denge penceresidir.

Denge etrafında, tampon işlemi RED altında gelişir,

$$\dot{b}(t) = \sum_l R_{li} \frac{w_i(t - \tau_{li}^f)}{\tau_i(t - \tau_{li}^f)} - c_l$$

$$= \sum_l R_{li} \frac{w_i(t - \tau_{li}^f)}{d_i + \sum_k R_{ki} b_k \tau_i(t - \tau_{li}^f)/c_k} - c_l$$

$\tau_i = d_i + \sum_k R_{ki} b_k^* / c_k$ dengeli RTT'dir (sıra gecikmelerini içerir). Doğrusallaşma sonrası, şunlara sahibiz,

$$\dot{b}(t) = \sum_i R_{li} \frac{w_i(t-\tau_{li}^f)}{\tau_i} - \sum_k \sum_i R_{li} R_{ki} \frac{w_i^*}{\tau_i^2 c_k} b_k(t-\tau_{li}^f)$$

eğer RTT'de ihmal edilmiş veya sabit varsayılmış sıra gecikmesi varsa yukarıdaki ikinci terim yok sayılabilir. Çift toplama işareti, her kaynak *i* ile hat *l* yi paylaşan tüm *k* hatları üzerini toplar. Bu ağdaki hat dinamiklerinin paylaşılan kaynak boyunca birleştiğini söylemektedir. $\frac{w_i^*}{\tau_i c_k} b_k(t-\tau_{li}^f)$ terimi, kaynak *i* nin paketleri yüzünden *k* hattında, FIFO sırasının altında kabaca gecikmiştir. Bundan dolayı gecikme $b_l(t)$, *l* hattında, diğer *k* hattında paylaşılan kaynak *i* nin gecikmesiyle uygun oranda azaltır. Kaynak *i* nin yolundaki gecikme *l* hattında paket alan kaynak *i* nin oranını azaltır, $b_l(t)$ azalır.

Herşeyi bir araya koyarsak, Reno/RED, Laplace domain'de şu şekilde belirtilir.

$$w(s) = -(sI + D_1)^{-1} D_2 R_b^T(s) p(s)$$

$$p(s) = (sI + D_3)^{-1} D_4 b(s)$$

$$b(s) = (sI + R_f(s) D_5 R^T D_6)^{-1} R_f(s) D_\tau w(s)$$

$D_1 = diag\left(\frac{q_i^* w_i^*}{\tau_i}\right)$, $D_2 = diag\left(\frac{1}{\tau_i q_i^*}\right)$, $D_3 = diag(\alpha_l c_l)$, $D_4 = diag(\alpha_l c_l \rho_l)$,
$D_5 = diag\left(\frac{w_i^*}{\tau_i^2}\right)$, $D_6 = diag\left(\frac{1}{c_l}\right)$, $D_7 = diag\left(\frac{1}{\tau_l}\right)$, diagonal matrisleri ve $R_f(s)$ *ve* $R_b(s)$ geciken ileri ve geri doğru yönlendirme matrisleridir,şu şekilde tanımlanır.

$$\left[R_f(s)\right]_{li} = \begin{cases} e^{-\tau_{li}^f s} & l \in L_i \\ 0 & aksitakdirde \end{cases} \qquad (5.10)$$

$$[R_b(s)]_{li} = \begin{cases} e^{-\tau_{li}^b s} & l \in L_i \\ 0 & aksitakdirde \end{cases} \qquad (5.11)$$

kaynak [25]'in benzer-kaynak modelini tek hattan, birden çok hatta, karışık kaynaklarla genelleştirir.

5.2.3 Geçerlilik ve Kararlılık Bölgesi

Sistem kararlı olduğunda, doğrusal modelimizin geçerliliğini ve kararlılık bölgesini sayısal olarak şekillerle, bir seri deneyle sunduk.

Kapasitesi *c* paket/ms olan tek bir hat ve bu hattın *N* kaynak tarafından benzer gidiş dönüş yayılma gecikmesi *d* ms ile paylaşıldığını düşünelim. *N= 20,30,.....,60* için kapasite *c=8,9,.....,15* pkt/ms ve yayılma gecikmesi *d = 50,55,.....,100 ms,* geri besleme sistemi *(L(jw))* nin (11)'de, döngüsel kazancının Nyquist planını inceliyoruz. Her bir *(N,c)* ikilisi için, gerçek ekseni -1'e en yakın olan ile Nyquist planın en küçük kesintisinde 5ms'lik artışda gecikme $d_m(N,c)$ elde edilir. Bu, sistem *(N,c)* de, doğrusal modele göre kararlılıkdan kararsızlığa geçiş gecikmesidir. Bu gecikme için, *L(jw)* 'nin fazında $f_m(N, c)$ kritik frekansı $-\Pi$ olarak hesaplanır. *L(jw)* nin hesaplanmasında, dengeli RTT τ , yayılma gecikmesinin toplamı $d_m(N,c)$ ve dengeli sıra gecikmesi gereklidir. Sıra gecikmesi [26]'deki duality modelden hesaplanmaktadır. Bundan dolayı, her bir *(N,C)* ikilisi, 50ms ve 100ms arasındaki gecikmede ancak kararsız olur, Kritik (yayılma) gecikmesi $d_m(N,c)$, ve kritik frekans $f_m(N,c)$ analitik modelden elde edilirler. Tüm deneyler için, bazı parametreler sabit tutulmuştur, $\alpha = 10^{-4}$, $\rho=0.1 / (540-40)=0.0002$, ve $\beta=0.5$.

Bu deneyler ns-2'de devamlı FTP kaynakları ve ECN işaretlemeli RED ile tekrarlanmıştır. RED parametreleri (0.1 , 40paket , 540 paket , 10^{-4}) olarak α ve ρ değerlerine tekabül eder. Her bir *(N,c)* ikilisi için, sistem kararlılıkdan kararsızlığa geçtiğinde, kritik gecikme $d_{ns}(N,C)$'yi elde etmek için sıra ve pencere eğrilerini inceleriz. Kritik frekans $f_{ns}(N,C)$ yi, sıra eğrisinin sıra salınımının temel frekansı olan FFT'den

ölçeriz. Bundan dolayı, doğrusal modele karşılık gelen, benzetmelerden kritik gecikme $d_{ns}(N,C)$ *ve* frekans $f_{ns}(N,c)$*'yi* elde edilir.

Model tahminini benzetme ile karşılaştırırız. Şekil 4.4(a) , doğrusal modelden hesaplanan $d_m(N,C)$ kritik gecikmeye karşın ns-2 simülatöründen $d_{ns}(N,C)$ kritik gecikmesini çizimini göstermektedir. Her bir data noktası özel bir (N,c) ikilisine karşılık gelmektedir. Şekil 4.4(b) kritik frekans $f_{ns}(N,c)$ *ye* karşılık gelen $f_m(N,c)$ çizimini vermektedir. Model ve benzetme arasındaki arasındaki anlaşma oldukça mantıklı görünmektedir.

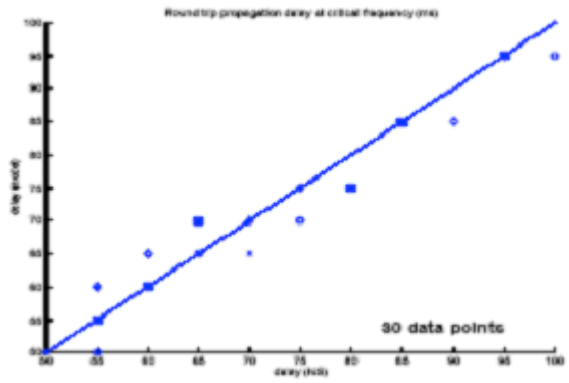
Statik bir hat modeli düşünelim ve işaretleme olasılığı hat akış oranının bir fonksiyonu olsun,

$$P_l(t) = f_l(y_l(t))$$

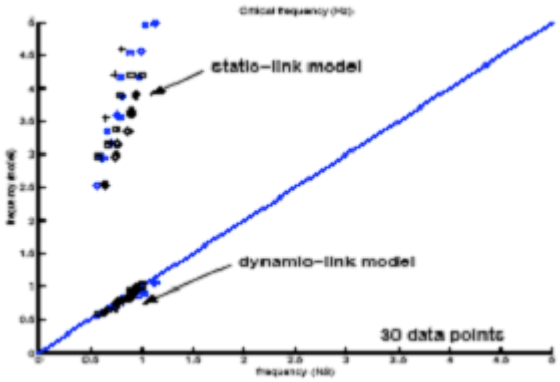
Sonra, doğrusallaşan model,

$$p_l(t) = f_l^1(y_l^*)y_l(t)$$

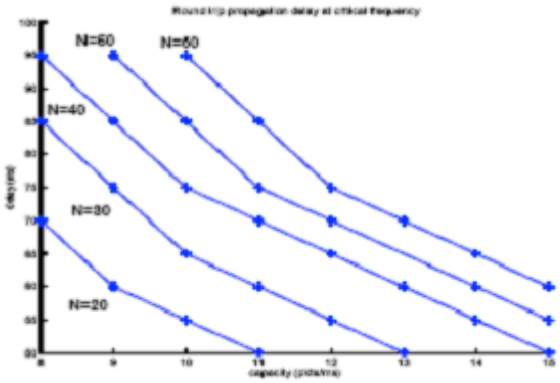
burada, $f_l^1(y_l^*)$, f_l nin türevidir ve dengeden elde edilir. Ayrıca şekil 4.4(b) de görülen kritik frekans, bu statik-hat modelinden tahmin edilmiştir. ($f_l^1(y_l^*) = \rho$ ile, bu kritik frekansı etkilemez), yukarıda tanımlanan aynı Nyquist çizim metodu ile kullanılır. İlgili zaman skalasında sıra dinamiklerini anlamlı olarak göstermiştir.



(a) Kritik gecikme (ms)



(b)Kritik Frekans (Hz)



(c)Kararlı bölge

Şekil 5.4. *Onaylama ve kararlı bölge. Her bir N için, eğrinin üstündeki bölge kararsızdır ve altı kararlıdır.*

Şekil 4.4(c), doğrusal model tarafından anlatılan kararlılığı göstermektedir. Her bir *N* için, kapasite *c* ye karşılık kritik gecikme $d_m(N,c)$ nin grafiğini gösterir. Eğri kararlı (alt) ve kararsız (üst) bölgeleri ayırmaktadır. Negatif eğim, gecikme veya kapasite çok büyük olduğunda TCP/RED'in kararsız olduğunu göstermektedir. *N* artarsa kararlı bölge genişler, örneğin küçük yük kararsızlığa neden olur. sezgisel olarak, daha geniş bir gecikme yada kapasite yada daha küçük bir yük, daha geniş dengeli pencerelere liderlik eder; bu da TCP'nin geniş bir pencere ölçüsünde rahatsız olduğunu doğrular.

5.2.4 Kararlılık : Tek Hatlı Karışık Kaynaklar

Şimdi *N* karışık kaynakla tek bir hat durumunda kararlı bölgeyi tanımlayacağız. Son alt bölümün doğrusal modelini bu duruma özelleştirirsek, ileri gecikmeyi RTT nin bir parçası olarak $\beta \in (0,1)$, $\tau_i^f = \beta\tau_i$ *ve* hat düşmesi alt simgesi *l* , döngü kazancı olarak ,

$$L(s) = \sum_i \frac{1}{\tau_i p^*(\tau_i s + p^* w_i^*)} \cdot \frac{\alpha c \rho}{s + \alpha c} \cdot \frac{1}{s + \frac{1}{c}\sum_n \frac{x_n^*}{\tau_n} e^{-\beta\tau_n s}} \cdot e^{-\tau_i s} \qquad (5.12)$$

elde edilir.

İlk terim, TCP dinamiklerini , ikinci terim RED ortalamalarını üçüncü terim tampon bellek işlemini ve son terim de ağ gecikmesini tanımlar. Tüm kaynaklar benzer RTT'lere sahiptir, $\tau_i = \tau$, ve ileri gecikmelerin sıfır, $\beta = 0$, olduğu özel durumda kaynak [25]'de analiz edilir. Kapalı-döngü durumu için yeterli şartları sağlarlar ve bunları α *ve* ρ RED parametrelerini ayarlamak için kullanırlar.

Aşağıda kullandığımız bazı denge özelliklerini toplayan bir yardımcı önerme ile başlayabiliriz. Eşitlik (5-8)'in sabit noktalarından doğrudan kanıtlanmıştır; yada kaynak [26]'ya bakınız. $\bar{\tau} := \max_i \tau_i$, $\underline{\tau} := \min_i \tau_i$ ve $\hat{\tau} := \left(\sum_i \frac{1}{\tau_i}\right)^{-1}$.

Yardımcı Önerme 1 *:* p^* denge düşme olasılığı, w_i^* ve x_i^* denge penceresi ve oranıdır. Sonra $p^* = 2 / (2 + (c\text{^}r)^2$, her kaynak *i* için $w_i^* = c\text{^}r$, $x_i^* = w_i^*/c = 1$.
Şimdi

$$\theta_0 := \arctan\frac{\pi(1-\beta)}{2\beta} \in \left(0, \frac{\pi}{2}\right) \qquad (5.13)$$

$$h_{red}(\upsilon,\theta) := \frac{e^{-j(v+\theta)}}{v(jv+\alpha c\underline{\tau})(jv+p^{*}w_1^{*})} \quad , \quad 0 \le \theta \le \pi - \theta_0 \qquad (5.14)$$

$\theta = \pi - \theta_0$ da $h_{red}(\upsilon,\theta)$ nın Nyquist çiziminin TCP/RED'in kararlılığını elde ettiğni göstereceğiz. $h_{red}(v,\pi-\theta_0) = -\pi$ fazında v_0 açı olsun.

***Teorem 2*:** Denge penceresi $w_i^{*} >= \sqrt{2}$ olsun. Sonra kapalı döngü sistemi eşitlik (5.12) tarafından tanımlanır ve eğer

$$c^3\bar{\tau}^3.\left|h_{red}(v_0,\pi-\theta_0)\right| \le \frac{1-\beta}{\alpha\rho}$$

ise kararlıdır. Burada $\bar{\tau} := \max_i \tau_i$ dir.

İspat(Taslak). Kapalı döngü sistemi kararlıdır eğer ki *L(s),* karmaşık düzlemde (-1,0) boyunca geçmiyorsa, burada *s* sağ yarım düzlemde değer alır. Bunu göstermek için, eşitlik (5.12)'yi yeniden yazarız.

$$L(s) = \frac{c^2\alpha\rho}{p^{*}}\sum_i \frac{w_i^{*}/\tau_i}{c}\frac{z_i(s)}{w_i^{*}}$$

burada

$$z_i(s) = \frac{e^{-\tau_i s}}{(s+\alpha c)(\tau_i s + p^{*}w_i^{*})}\frac{1}{(s+\sum_n \frac{x_n^{*}}{c\tau_n}e^{-\beta\tau_n s})} \qquad (5.15)$$

yardımcı önerme 1 şunu içerir,

$$L(s) = \frac{c\alpha\rho}{\hat{q}p^*} \sum_i \frac{x_i^*}{c} z_i(s) \qquad (5.16)$$

L(s), karmaşık düzlemde *N* 'in belirttiği $z_i(s)$ tarafından conveks gövde içinde tanımlanır. Bu karmaşık gövdenin (-1,0) dan uzakta sınırlandırıldığını, iki adımda göstereceğiz [21].

Birincisi, büyüklü ve eşitlik (5.15) deki son terimin sınırlandırılmasıdır. Şunu gösterebiliriz,

$$L(jw) \in \frac{\bar{\tau}^2 c\alpha\rho}{(1-\beta)\hat{q}p^*} .co\{h_{red}(v,\theta) : v \geq 0, 0 \leq \theta \leq \pi - \theta_0\} \qquad (5.17)$$

eşitlik (5.15) de tanımlanan conveks gövde eşitlik (5.14) de tanımlanan $h_{red}(v,\theta)$ nın daha geniş karmaşık gövde içerisine yerleştirilmiştir.

Sonra teorem sınırlarının,bu küme (-1,0)'dan uzaktır, hipotezini gösterir. $h_{red}(v,\theta)$ nun yörüngesi eğridir.

$$C(v) := \frac{e^{-jv}}{v(jv + \alpha c\underline{\tau})(jv + p^* w_1^*)}$$

negatif yönde θ kadar döndürülmüştür. Bundan dolayı $|C(v)|$ büyüklüğü, *v*'de düşer, eşitlik (5.17)'da conveks gövdenin sol sınırı $\theta = \Pi - \theta_0$ da $h_{red}(v,\theta)$ tarafından tanımlanır. düzlemsel eğri $h_{red}(v,\Pi - \theta_0)$ ın eğrileştirilmesinin incelenmesiyle, *v,* 0 dan $+\infty$'a değişiklik gösterir [32], bu kümenin sınırlarının $|h_{red}(v,\Pi - \theta_0)|$ da gerçek eksende kesintiği gösterilebilir. Eğer ki,

$$\frac{\bar{\tau}^2 c\alpha\rho}{(1-\beta)\hat{q}p^*} .\left|h_{red}(v_0, \pi - \theta_0)\right| < 1$$

Bu koşul teoremin hipotezinin altında tutulduğu daha sonra gösterilebilir. $h_{red}(v,\theta)$'nun conveks gövdede $L(jw)$ Nyquist çizimini sınırlanması düşüncesi, [33]'ün farklı bir algoritma tarafından ispat edilmesinden esinlenilmiştir.

5.3 RED Parametre Ayarları

Teorem 2 de kararlılık şartının RHS üzerindeki $\alpha \in (0,1]$ ve $\rho > 0$ parametreleri RED'in sıra uzunluğunun çarpansal ortalaması ve işaretleme olasılığının eğimidir. Kararlılık için *ürünleri* küçük olmalıdır. Küçük α yavaş cevaba neden olur çünkü anlık sıra uzunluğundaki data geri beslemeye çok yavaşca dahil edilmiştir. Küçük ρ ise geniş gecikme meydana getirir, dengedeki ortalama sıra uzunluğu r, $r = \underline{b}_l + 2/\rho(2+(c\hat{r})^2)$ *dir.* Doğrusu kaynaklar benzer olduğu zaman $\tau = \bar{\tau} = \underline{\tau} = \hat{t}N$ *dir.* Kararlılık şartının LHS'si $\frac{c^3\tau^3}{2N}|h|$ olur. gecikme τ yada kapasite c arttığı zaman, TCP/RED kararsız olur, son alt bölümdeki benzetme sonuçlarının onaylanmasıdır. Kabaca, c çiftse, denge oranı çifttir ve bundan dolayı iki katı frekansda iki katı büyüklükle pencere yarıya indirilir, sonuç da kontrol kazancında ikinci dereceden artış sistemi kararsız yapar.

Kaynak [36] da RED parametresi $\max_p$ nin dinamik ayarlanması önerilmiştir, $\max_p$ nin düşürülmesi N azalır ve aksi takdirde yükselir. $\max_p$ nin yükseltilmesi yada $\max_{th}$ nin – $\min_{th}$ nin düşürülmesi, teorem 2'de kararlılık şartını içeren yönde arttırmak için eşittir. $\rho\ (= max_p / (max_{th} - min_{th}))$. Teorem 2, verilen N, c, τ *(ve* α*)* ile ρ üzerindeki üst sınırı ayarlar, bundan dolayı alt sınırıda kararlılığı sağlamak için dengedeki sıra uzunluğu üzerindedir. RED parametrelerinin ayarlanması kararlılık ve performans arasındaki kaçınılmaz seçeneği korumaz, sıranın kararlılığı için ya ρ geniş bir değer etrafında küçük ayarlanır ya da alternatif olarak, şiddetli salınımının harcanmasında,sırayı azaltmak için geniş ayarlanır.

Aynı kararlılık analizi sanal sıra [37,38,39] ve REM/PI [27,40] gibi diğer AQM'lere de uygulandı ve AQM'in rolünü aydınlattılar. Kararlılığın ispatı bir takım formun

$$K.co\{h(v,\theta)\}$$

(-1 ,0) ın sağına sınırlanmasına güvenir. Kazanç K ve yörünge h AQM de olduğu gibi TCP'ye bağlıdır. Örneğin, c kapasiteli N benzer kaynak tarafından paylaşılan τ gecikmeli, tek bir hat için TCP ve ağ gecikmesi yörünge h'a bir parça katkı olarak,

$$h_{tcp} = \frac{e^{-jv}}{jv + p^* w_1^*}$$

ve kazanç K'ya

$$K_{tcp} = \frac{c^2\tau^2}{2N} \qquad (5.18)$$

küçük bir katkıda bulunur. Denge penceresi geniş varsayılır böylece $p^* = 2/w_i^2 = 2N/c\tau$ olur. bu yüksek kazanç eşitlik (5.18) , esas olarak yüksek gecikme, yüksek kapasite yada düşük yükde kararsızlıkdan sorumludur. AQM, bu etkiler için h şekillendirerek ve K yı düşürerek telafi eder. RED'le, örneğin,

$$h(v,\theta) = \frac{1}{jv + \alpha c\tau}\frac{e^{-j\theta}}{v}.h_{tcp}$$

$$K = \frac{c\tau\alpha\rho}{1-\beta}.K_{tcp}$$

h da ilk terim RED ortalaması yüzünden, ikinci terimde sıra dinamikleri yüzünden $\theta \leq \pi - \theta_0$ sınırlanır. Bundan dolayı sıra ve RED'in her ikiside faz geri kalmasını h 'a ekler. Daha önemlisi , RED başka $c\tau$ yi kazanç K 'ya ekler, kararlılık için küçük $\alpha\rho$ zorunlu

kılınır ve ağır cevap ve geniş dengeli sıraya sebep olur. K daki $\tau / (1 - \beta)$ parçası sıradan gelir.

5.4 Kararlılık Kontrolü

Kaynak [29]'daki ölçüler şimdiki internette gecikmenin hala geniş olduğunu göstermektedir (RTT ölçülerinin %85'i 15-500 ms aralığındadır). Önceki bölümlerin sonuçları şimdiki protokolün böyle çevreler için kötü düzenlendiğini belirtmektedir. Dahası, gelecekte ağ kapasitelerinin genişlemesi ile bu durum daha da kötüleşecektir. TCP tarafından belirtilen yüksek kazancı telafi etmek için AQM'lerin tasarlanması da zor görünmektedir. Bu bölümde, [28]'de geliştirilen, kaynaklar ve hatlar tarafından dağıtılmış merkezi olmayan bir yolla gerçekleştirilen ve kararlı olan bir protokol tanımlayacağız, bu rastgle seçilen gecikme, kapasite, yük ve yönlendirme için doğrusal kararlılığı devam ettirir. Dahası, küçük sıralar ile yüksek ağ kullanımının dengesini sağlamayı başarır. Bu gereksinimler doğrusal dinamikler üzerinde bazı sınırlamaları kabul ettirir. Hatlarda bütünleşme ve kaynaklarda ve hatlarda kazanç şartlarıdır.

5.4.1 Algoritma

Kaynak [28] deki sıkışıklık kontrol algoritmasını özetlersek, bu statik kaynak algoritmasını ve birinci dereceden dinamik hat algoritmasını içerir. Buradaki ana fikir, kaynaklardaki gecikmeyi, bireysel RTT'lerle oranlardaki kazançları küçülterek telafi etmek ve kapasite ve yönlendirme tarafından belirtilen döngü kazançlarını, varolan oranlarıyla kaynaklarda büyüterek ve kapasiteleriyle kontrol kazancının küçültülmesiyle telafi etmektir. Diğer bir deyişle, eğer gidiş dönüş gecikmesi genişse yada oranı küçükse kaynak daha da yavaşlayarak tepki verir; eğer hat daha geniş bir kapasiteye sahipse sıkışma ölçüsünü (price , ücret) daha yavaş olarak günceller. Ağ gecikmesi sadece açık – döngü parametresidir, kontrolümüz altında değildir ve sistem cevabının zaman skalasını ayarlamalıdır.

RED'in doğrusal olmayan modelinde tanımlanan ağ modelini düşünelim, $p_l(t)$, t zamanında l hattında ücret olsun ve c_l sanal kapasite (gerçek kapasiteden daha az) olsun. Her bir l hattı kendi ücretini giriş oranını $y_l(t)=\sum_s R_{ls}x_s(t)$ kullanarak ayarlar.

$$\dot{p}_l(t) = \begin{cases} \dfrac{y_l c_l}{c_l} & p_l > 0 \\ \max\{0, \dfrac{y_l c_l}{c_l}\} & p_l(t) = 0 \end{cases} \qquad (5.19)$$

Bundan dolayı ücretler fazlalık olan kapasiteleri normalleştirilmiş yolla bütünleştirir ve her zaman negatif olmamaya doyurulur. Dengede, sıfır olmayan ücretlerle dar geçitler $y_l^*=c_l$, sahip olurlar, yüksek kullanım verirler. $y_l^* < c_l$ ile dar olmayan geçitlerin ücretleri sıfır olacaktır. c_l gerçek kapasiteden küçük olduğundan , dengede sıra ihmal edilebilir. Eğer c_l gerçek kapasiteyse, $p_l(t)$ gerçek sıra gecikmesi olur, TCP Vegas'da [30] bir sıkışma sinyali kullanılır.

$x_i(t)$ t zamanında i kaynağının oranı olsun, τ_i RTT'si ve M_i yolunda sıkışan hat sayısıdır (yada üst sınır). Verilen toplu ücret $q_i(t) =\sum_l R_{li}p_l(t)$, kaynak i, $q_i(t)$ *'de* oranını çarpansal olarak ayarlar,

$$x_i(t) = x_{\max,i} e^{-\frac{\alpha_i q_i(t)}{M_i \tau_i}} \qquad (5.20)$$

burada $x_{max,i}$ maksimum oran parametresidir, ve $\alpha \in (0,1)$. Araç (utility) fonksiyonu kaynak kontrolüne karşılık gelir, o da,

$$U_i(x) = \frac{M_i \tau_i}{\alpha_i}\left[1 - \log\left(\frac{x}{x_{\max,i}}\right)\right], \quad her\ x \le x_{max,i};$$

Yönlendirme matrisi R nin tam satır sırasına sahip olduğu varsayılır. sonra tek bir denge oranı ve ücret vektörü (x^*,p^*) vardır. Denge etrafında doğrusallaştırılmış sistem şu şekilde tanımlanır,

$$\dot{p}_l(t) = \frac{y_l(t)}{c_l}, \quad her\,l \quad için \tag{5.21}$$

$$\dot{x}_i(t) = -\frac{\alpha_i x_i^*}{M_i \tau_i}, \quad her\,s \quad için \tag{5.22}$$

burada kaynak oranları *x(t)* ve hat ücretleri *p(t)* , eşitlik (5.10 – 5.11) de tanımlanan geciken yönlendirme matrisi tarafından birbirlerine bağlanmıştır.

Aşağıdaki teorem kaynak [28] de ispatlanmıştır, rastgele seçilen bir gecikme, kapasite ve yükde ağ büyüdüğü zaman algoritmanın kararlılığını garanti eder.

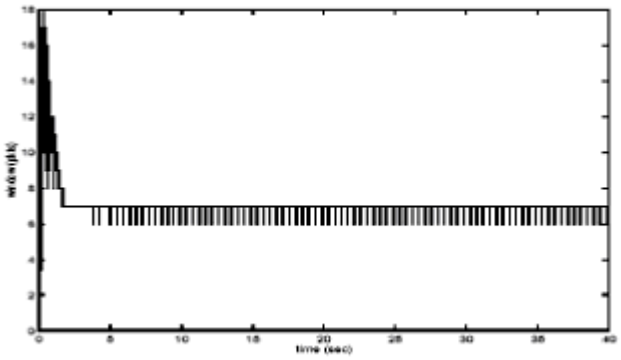
Teorem 3([28]): *R* de içerilen tüm hatların dar geçit olduğunu varsayalım, Örneğin dengede $c = Rx^*$ ve R tam dolu satır sıralarına sahiptir. Daha sonra eşitlik (20-21) tarafından tanımlanan kapalı-döngü sistemi ve eşitlik (5.10 – 5.11), rastgele seçilen gecikmeler τ_i ve hat kapasiteleri c_l için doğrusal olarak kararlıdır.

5.4.2 Gerçekleştirim ve Performans

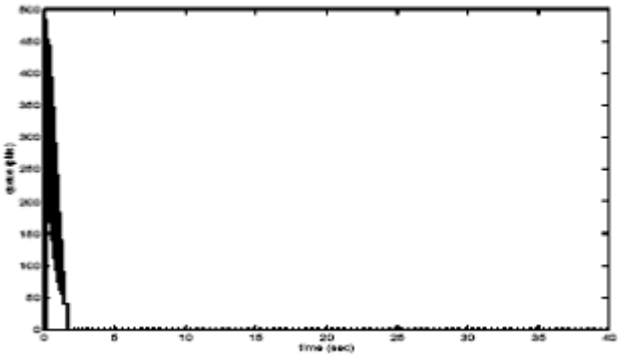
Eşitlik (5.19) deki hat algoritmasını gerçekleştirmek için basit bir yol, alınan paketlerle arttırılan ve sanal kapasite oranında azaltılan "sanal sıra" sayacının devamını sağlamaktır. Sonra ücretler sanal kapasite tarafından sayacın bölünmesiyle elde edilir.

Kaynaklar kendilerinin RTT τ_i 'ni ölçerler. Hedef durum boş sıralarla dengede olduğundan, τ_i yayılma gecikmesidir; bundan dolayı, kaynak güncellenmesinde (5.20) d_i nin bir tahmininin (tipik olarak elde edilen en küçük RTT) kullanılması önerilir, bu Vegas'da yapılmaktadır. Bu, gerçek sıranın geçici turları RTT aracılığıyla azaltıcı etkiye sahip olma olasığından kaçınmaktadır. Ayrıca kaynaklar ağdan iki parametre almak zorundadır, bu parametreler toplu ücret q_i , ve dar geçitlerin sayısı M_i 'dir. İletişim kurmak için q_i, rastgele çarpansal işaretleme tekniği kullanılabilir. Burada bir paket, her bir l hattında $1-\phi^{-p_l}, \phi > 1$ olasılığı ile işaretlenmiş olur. Bağımsızlığı varsayalım, bir paketin

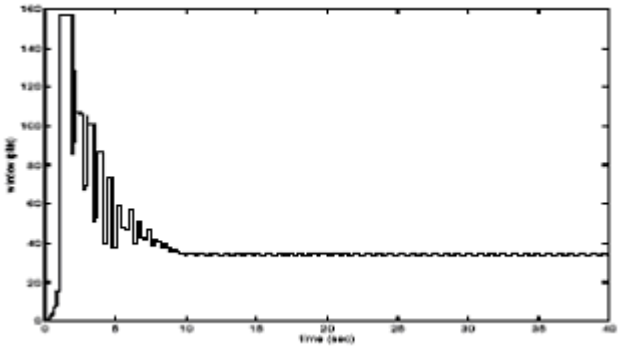
kaynak i den işatlenmesinin tüm olasılığı $1-\phi^{-q_i}$ dir, bundan dolayı q_i işaretlenme istatistiklerinde tahmin edilebilir. Bu, global bir sabiti bir öncelik ayarlayan ϕ'nın öz bilgisini gerektirir. M_i hakkında ise, en basit gerçekleştirmede basitce bir üst sınırı kullanabilir. paket seviyesi gerçekleştiriminin hazırlığı paralel simülator Parsec [41], kullanılarak yapılır. Bu benzetme, pencere yönetimi, hat sırası ve gecikmesini içerir, ama bu noktada işaretlemeyi içermez ; ücretler, ondalıklı sayılar olarak ifade edilirler. *N, c, d* 'nin geniş bir sınıfı için aynı şekil 5.1 deki parametrelerle, tek bir hattı simüle ederiz. Şekil 5.5, kişisel pencere ve sırayı göstermektedir, beklendiği gibi kişisel pencere ve sıra, gecikme ne olursa olsun yakınlaşırlar. Kapalı döngü davranışı için daha uzun gecikmeler, daha uzun zaman skalası ayarlar.



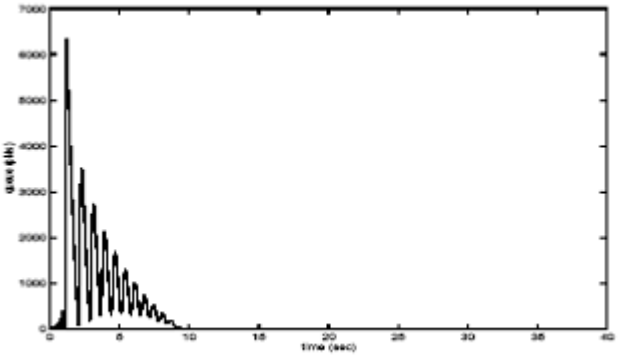
(a) Kişisel pencere (gecikme = 40ms)



(b) Sıra (gecikme = 40ms)



(c) Kişisel pencere (gecikme = 200ms)



(d) sıra (gecikme = 200ms)

Şekil 5.5. *Kişisel pencere ve sıra izleri. Benzetme parametreleri : 50 kaynak, kapasite = 9 paket/ms, α=0.8, sanal kapasite = %95.*

Bu noktada, performansın harcanmasında elde edilen protokolün kararlılık durumu ne olursa olsun, bir şaşkınlık olabilir. Örneğin cevap zamanını çok yavaşlatması. Bununla beraber şekil 5.1 deki karşılaştırma bu durum değildir. Burada Reno ve yeni protokolün kararlı duruma ulaşabilmesi için yaklaşık 50 RTT'ye ihtiyacı vardır. Mesela 200ms'lik gecikme durumunda, Reno'nun limit sayıya ulaşabilmesi yaklaşık 10 saniye alır ve bizim protokolümüz içinde boş sıra ile dengeye ulaşabilmesi için aynı miktar zamana ihtiyaç vardır.

BÖLÜM 6

RED'İN KONTROL TEORİSİ ANALİZİ

Kaynak [7], Ağ topluluğundaki araştırmalara liderlik etmiştir, AQM'ler için IP yönlendiricilerinde RED'in gerçekleştirimini önerilmiştir. RED'in akışların eş zamanlılıkları ile ilgili problemleri hafiflettiğine ve ayrıca zeki düşürme tarafından servisin kalitesinin bazı kavramlarını sağladığına inanılmıştır. RED'in analizi birçok ilginç yazıdan genelleştirilmiştir. RED parametrelerinin ayarlanması bazı zamanlar için hatalı olmaktadır, kullanılan RED'e karşı ayarlanmasının zorluğundan dolayı [10,22], birçok araştırma savunulmuştur. Sayısız RED değişimleri önerilmiştir. Kaynak [12-14],[36], belkide RED'in dinamiklerini tamamen anlayabilmenin zorluğundan motive olmuştur [25].

Kaynak [49]'da yazar RED parametrelerinin önerilerinin sorunlarını keşfetti ve göz atılan kuralları ve seçimleri için bir kılavuz verdi. Bu bölümde kontrol teoriği bakımından, kaynak [25] tarafından keşfedilen benzer şekilde daha ciddi problemler incelenmiştir. TCP ve RED dinamiklerinin daha önce geliştirilmiş bir modelini, analizimizin başlangıç noktasını göstermek için kullandık. Kalıtsal olarak sunulan doğrusal olmayan model, doğrusallaştırma (linearization) tekniği aracılığıyla doğrusal bir sisteme dönüştürülmüştür ve klasik doğrusal geri besleme kontrol teorisinde sonradan gelen iyi geliştirilmiş araçlara uyguladık. Doğrusal kararlı sistemlerin tasarımında, olduğu gibi doğrusal sistemin kararlılığı ve güçlülüğünü belirten, sağlanan metrikleri veren bir kılavuz verebiliriz. Analizimiz ayrıca çeşitli parametre seçeneklerinin takası ile ilgilidir. Bu analizde doğrusal olmayan benzetmeler aracılığıyla iyi bilinen ns-2 simülatörünü kullanılmıştır [25].

6.1 MODEL

Tartışmamıza AQM'in ilk önce TCP'nin sıkışıklıkdan kaçınma modu için bir dinamik modelin tanıtılmasıyla başlıyoruz.

6.1.1 TCP davranışının bir akıcı-akış modeli

Kaynak [54]'de, TCP davranışının bir dinamik modeli, akıcı-akış ve tahmini diferansiyel eşitlik analizleri kullanarak geliştirildi. Benzetme sonuçları gösterilmiştir, bu model hatasız olarak ele geçirilen TCP'nin dinamikleridir. Bu bölümde, bu modelin TCP zaman aşımı mekanizması yok sayılarak basitleştirilmiş bir versiyonu kullanılmıştır. Bu model anahtar ağ değişkenlerinin ortalama değeri ile ilgilidir ve aşağıdaki birleştirilmiş doğrusal olmayan diferansiyel denklemlerle tanımlanır.

$$\dot{W}(t) = \frac{1}{R(t)} - \frac{W(t)W(t-R(t))}{2R(t-R(t))} p(t-R(t))$$

$$\dot{q}(t) = \frac{W(t)}{R(t)} N(t) - C \qquad (6.1)$$

Burada $\dot{x}$, x 'in zaman türevini belirtmektedir. Ve

W	=	beklenen TCP pencere ölçüsü (paket);
q	=	beklenen sıra uzunluğu(paket);
R	=	gidiş dönüş zamanı (RTT) = q/C + T_p (saniye);
C	=	hat kapasitesi(paket/sn);
T_p	=	yayılma gecikmesi (sn);
N	=	yük faktörü (TCP oturumlarının sayısı);
p	=	paketlerin işaretlenme/düşme olasılığıdır.

Sıra uzunluğu q ve pencere büyüklüğü W pozitif ve sınırlı büyüklüklerdir; yani, $q \in [0,\bar{q}]$ ve $W \in [0,\overline{W}]$, burada $\bar{q}$ *ve* $\overline{W}$ tampon bellek kapasitesi ve en büyük pencere ölçüsüdür. Ayrıca işaretleme olasılığı p de sadece [0,1] aralığında değerler alabilir. Bu diferansiyel denklemleri blok diyagram olarak TCP pencere-kontrol ve sıra dinamiklerine dikkat çeken şekil 6.1'de gösterdik. Şimdi bu dinamikler küçük sinyal

doğrusallaşmalarıyla, AQM geri besleme kontrol sistemlerinin önerilerinde iç yüzünü anlamak için yaklaşık bir işletim noktası civarındadırlar.

6.1.2 Doğrusallaştırma

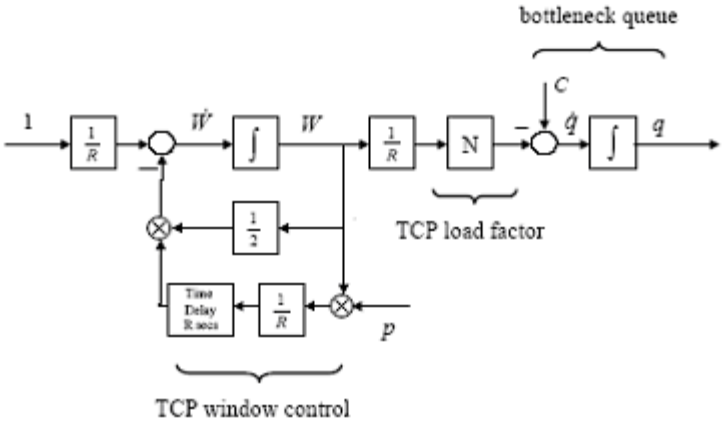
(W,q) durum olarak ve p giriş olarak alınır, işletim noktası (W_0,q_0,p_0), $\dot{W}=0$ *ve* $\dot{q}=0$ tarafından tanımlanır. Böylece

$$\dot{W}=0 \quad \Rightarrow W_0^2 p_0 = 2$$

$$\dot{q}=0 \quad \Rightarrow W_0 = \frac{R_0 C}{N} \qquad (6.2)$$

Burada,

$$R_0 \doteq \frac{q_0}{C} + T_p$$



Şekil 6.1. *TCP sıkışıklıkdan kaçınma akış kontrol modunun blok diyagramı [25].*

$N(t) \equiv N$ *ve* $R(t) \equiv R_0$ sabitler olarak varsayalım, işletim noktası etrafında elde edebilmek için doğrusallaştıralım.

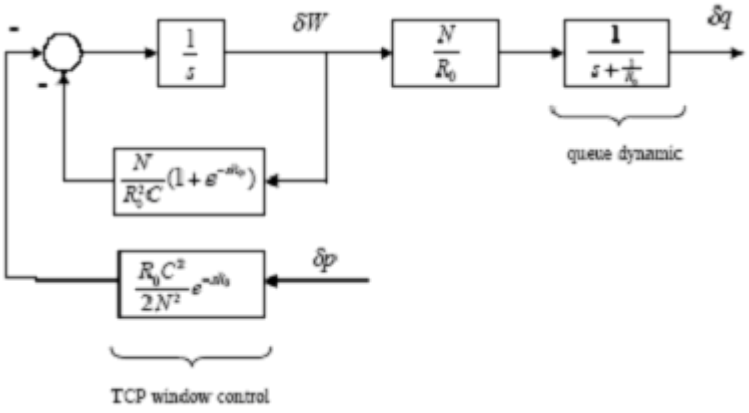
$$\delta\dot{W}(t) = -\frac{N}{R_0^2 C}(\delta W(t) + \delta W(t - R_0)) - \frac{R_0 C^2}{2N^2}\delta p(t - R_0)$$

$$\dot{\delta q}(t) = \frac{N}{R_0}\delta W(t) - \frac{1}{R_0}\delta q(t) \qquad (6.3)$$

Burada

$$\begin{aligned} \delta W &\doteq W - W_0; \\ \delta q &\doteq q - q_0; \\ \delta p &\doteq p - p_0. \end{aligned}$$

Diferansiyel eşitlikler üzerinde Laplace dönüşümünü yapan, şekil 6.2'de gösterilen doğrusallaştırılmış dinamiklerdir.



Şekil 6.2. *Doğrusallaştırılmış TCP bağlantılarının block diyagramı* [25].

Hatırlatmalar 1 [25]:

1. Kaynak [56]'da, TCP'nin pencere kontrol mekanizması için bir model geliştirildi ve ispat edildi, ve Smith düzenleyici yapısına dahil edildi [57]. Bununla beraber, bizim modelimiz, eşitlik (6.3) ve şekil 6.2 bu açıklamayı desteklemez. Hakikaten,

TCP pencere kontrolü için şekil 6.2'nin Smith düzenleyici yapısı gibi davranması için $\frac{N}{R_0^2 C} e^{-sR_0}$ teriminin $-\frac{C^2}{2N} \frac{1}{s+\frac{1}{R_0}} e^{-sR_0}$ ile yer değiştirmelidir.

2. Kaynak [58]'de TCP'nin pencere kontrol mekanizması için bir model geliştirildi bu eşitlik (6.3)'dekine benzemektedir. Bunun yanında bu model bir sıra dinamiği içermez. Dinamik sistem düşünülür bundan dolayı bizimkinden küçük bir farkı vardır, ve analiz ve sonuçları ile bizim ulaştığımız sonuca katılmaz.

3. Gecikme terimi e^{-sR_0} 'ı TCP pencere kontrol dinamiğinde şekil 6.2 de gösterdik ve aşağıdaki durumda anlamlı değildir.

$$\frac{N}{R_0^2 C} \ll \frac{1}{R_0}$$

bundan dolayı

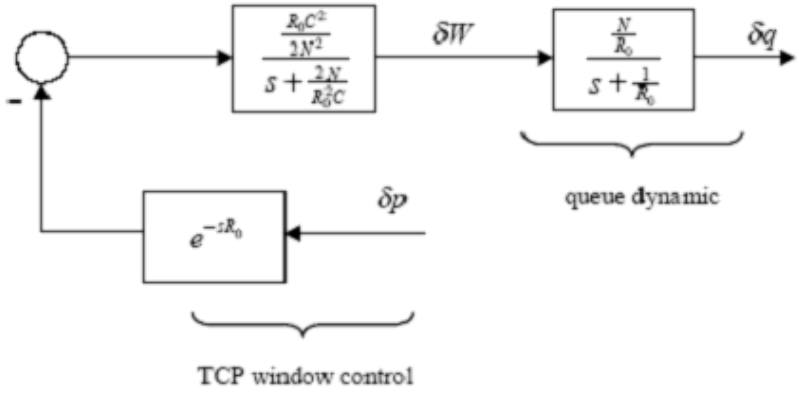
$$\frac{N}{R_0^2 C} = \frac{1}{W_0 R_0}$$

bu gecikme terimi eğer $W_0 >> 1$ ise yok sayılabilir. Tipik ağ şartları için, $W_0 >> 1$ kabul edilebilir bir varsayımdır ve bundan dolayı yazının geri kalanı için bu gecikme terimi yok sayılacak ve basitleştirilmiş dinamik düşünülecektir.

$$\delta\dot{W}(t) = -\frac{2N}{R_0^2 C}\delta W(t) - \frac{R_0 C^2}{2N^2}\delta p(t - R_0)$$

$$\dot{\delta q}(t) = \frac{N}{R_0}\delta W(t) - \frac{1}{R_0}\delta p(t) \qquad (6.4)$$

blok diyagramı şekil 6.3 de gösterilmiştir.



Şekil 6.3. *W_0 >> 1 olduğunda doğrusallaştırılmış TCP bağlantılarının block diyagramı* [25].

4 Doğrusallaştırılmış TCP'nin eigen değerleri ve sıra dinamikleri eşitlik (6.4) 'de sırasıyla,

$$-\frac{2N}{R_0^2 C} \quad (yada \quad \frac{-2}{W_0 R_0}) \quad ve - \frac{1}{R_0}$$

Tüm ağ parametrelerinin pozitif değerler olmasından dolayı, bu negatif Eigen değerleri doğrusal olmayan dinamiklerin denge durumunu belirtir, yerel olarak asimptotik olarak kararlıdır. TCP pencere-kontrol zaman sabiti $\frac{W_0 R_0}{2}$ 'nin yorumu, aşağıdaki $\delta \dot{W}$ eşitliğinin doğrusallaşmasının ifade edilmesinden gelir,

$$\delta \dot{W}(t) = -\lambda_0 \delta W(t) - \frac{R_0 C^2}{2N^2} \delta p(t - R_0)$$

burada λ_0, kaynak [54]'da tartışıldığı gibi dengeli paket işaretleme oranıdır. Bundan dolayı pencere kontrol zaman sabiti eşiti olarak *1/* λ_0 ifade edilebilir. Dengede, $\dot{W} = 0$ pencere ölçüsündeki çarpansal düşüşü (multiplicative decrease) $½ W_0 \lambda_0$, eklemeli artışını *$1/R_0$* dengeler. Sonuç olarak $\lambda_0 = \frac{-2}{W_0 R_0}$. TCP sıkışmadan kaçınma döngüsünün ortalama frekansı olarak gevşekce yorumlanır.

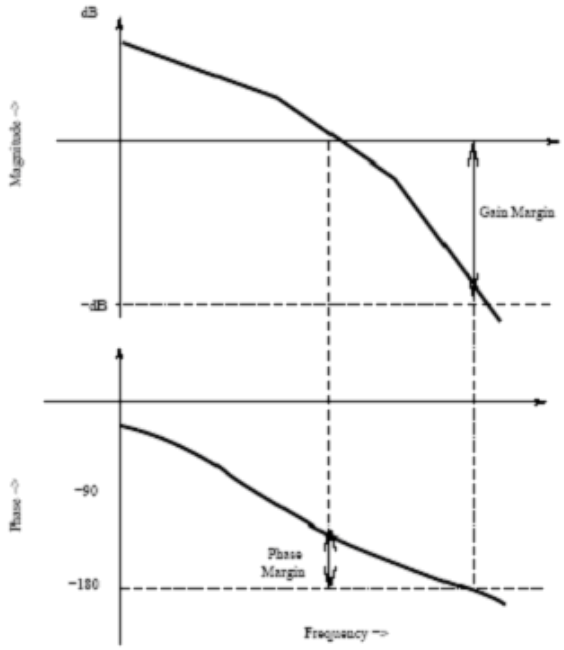
5 Son olarak, sıra dinamiklerinin doğrusallaştırılması sade bir bütüleştirici kazandırmaması beklenen ve literatürde görülene göre ilginçtir [14], ama R_0 zaman sabiti ile sızıntılı bir bütünleştirici üretilir. Bu parçalı olarak, sıra içine giden akışın sıra uzunluğunun bir fonksiyonu olmasıyla açıklanabilir. Bu akış NW/R_0 dır, burada RTT'nin bir parçası R_0 sıra gecikmesi q/C den dolayıdır.

6.2 AQM KONTROL PROBLEMİ

Bu bölümün konusu eşitlik (6.4)'de tanımlanan TCP dinamiklerini, TCP yükü N , RTT'si R_0 ve sıra kapasitesi C gibi ağ parametreleri cinsinden ve AQM'in geri besleme doğası cinsinden analiz eder. Ayrıca AQM'in performans amaçlarını tartışacağız.

Doğrusallaştırılmış TCP modeli eşitlik (6.4) kullanılarak şekil 6.5'deki blok diyagramda da görüldüğü gibi bir AQM kontrol sistemi modellenebilir. Bu diyagramda P_{tcp}, kaybolma olasılığından δ_p pencere büyüklüğüne δW ve $P_{sıra}$ bağlıdır δW sıra uzunluğuna q transfer fonksiyonlarını belirtir. $e^{-sR}{}_0$ terimi geciken düşme olasılığındaki $\delta p(t-R_0)$ zaman gecikmesinin Laplace dönüşümüdür. Kontrol sistem dilinde, "kontrolör" veya "denkleştirici" olarak, geri kalanını da "sistem" (plant) olarak AQM kontrol yasasını belirtiriz. Denkleştirici tasarımının amacı "kararlı" kapalı döngü sistemi sağlamaktır. Bunun yanında, kararlılığın ötesinde kontrol tasarımına etkilerine ilgileri vardır. İlk önce sistem, kabul edilebilir iletim cevabına sahip olmak zorundadır. İkinci olarak düzenleyici tasarımı, model hataları ve model parametrelerinin çeşitliliğinde kuvvetli olmalıdır. Bundan dolayı, kontrol mühendislerinin amacı emniyet payı ile sistem tasarlamaktır. Bu paylar *kararlılık payı* olarak adlandırılırlar. Bu göreceli kararlılığı ölçmek için iki klasik metrik vardır. Bunlardan birincisi *kazanç payıdır*. Bu, kararlı sistemin, kararsız olmasında kazanılan açık-döngü kazancının bir parçasıdır. Şekil 6.1'e bakarsak, kazanç payı kabaca yük seviyesi N de kesin değildir, tasarım bu durumu hoş görebilir. Bu ölçülerin ikincisi *faz payıdır*. Faz payının tanımı biraz daha karmaşıkdır, ama bu bağlamda, faz payını RTT gecikmesindeki belirsizliğin miktarı olarak yorumlayabiliriz, bir tasarım kararsız olmadan güçlü tutabilir. Sistemin kararlılık payı *Bode* çizimlerinden okunarak anlaşılabilir. Bir

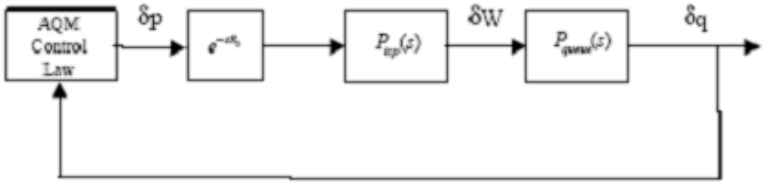
Bode çizimi, açık döngü sisteminin frekans cevabının çizimidir. Sistemin büyüklüğü ve faz cevabı çift kayıt skalası üzerinde çizilir. Sistemin kazanç payı faz cevabının -180^0 olduğu noktadaki, sistemin cevap büyüklüğüne eşittir. Faz payı ϕ_m , w_{pm}-*180* olarak tanımlanır, burada w_{pm} , cevap büyüklüğünün birleştiği (yada 0dB) yerdeki frekansdaki faz cevabıdır. Şekil 6.4'de iki miktar gösterilmektedir. Sezgisel olarak eğer pozitif payımız yoksa, geri besleme kontrol sistemi pozitif bir geri besleme sistemi gibi davranmaya başlar, yani birisi hata aldığında döngüde güçlendirilir, farklı ve kararsız davranmasını sağlar [25].



Şekil 6.4. *Kararlılık payı.*

6.2.1 Sistem dinamikleri

Şekil 6.5'de AQM sisteminin tarif edildiği bir geri besleme kontrol sistemi verdik. Bir AQM kontrol yasasının hareketi paketleri ölçülen sıra uzunluğu *q* nun bir fonksiyonu olarak işaretlemektir. (*p* olasılığı ile).



Şekil 6.5. *Geri besleme kontrolü olarak AQM.*

Şekil 6.5'den , sistem transfer fonksiyonu, $P(s) = P_{tcp}(s)P_{sira}(s)$, ağ parametrelerinin kazancı cinsinden ifade edilebilir.

$$P_{tcp}(s) = \frac{\frac{R_0 C^2}{2N^2}}{s + \frac{2N}{R_0^2 C}};$$

$$P_{Sira}(s) = \frac{\frac{N}{R_0}}{s + \frac{1}{R_0}} \qquad (6.5)$$

iki kutup için $-2N/(R_0^2 C)$ ve $-1/R_0$ olarak p_{tcp} ve p_{sira} sırasıyla gönderilir.

Sistem dinamikleri transfer fonksiyonu $P(s)$ tarafından belirtilir, sonra bu paket işaretleme olasılığının nasıl dinamik olarak sıra uzunluğunu etkilediğini gösterir. Eşitlik (6.4)'den ve şekil 6.3 den, şuna sahibiz ,

$$P(s) = \frac{(\frac{C^2}{2N})e^{-sR_0}}{(s + \frac{2N}{R_0^2 C})(s + \frac{1}{R_0})}. \qquad (6.6)$$

Hatırlatmalar 2:

1. Eşitlik (6.6)'da $P(s)$ nin yüksek frekans sistem kazancı $C^2/2N$ 'dir. Bu kazançdaki değişim TCP yükü N 'in bir fonksiyonu olarak AQM kontrol şemasının

tasarımıyla ilgisi olmalıdır, bundan dolayı kararlılık, geçici cevap ve sağlam durum performansı ile doğrudan ilgilidir. Gerçekten küçük bir TCP yükü *N* yüksek frekans kazancını arttırır, kararlılık payını düşürmeye ve salınım cevabını arttırmaya önder olur. Karşıt olarak, daha geniş TCP yükü, kapalı döngü geçici cevabının gücünü azaltmaya eğimlidir.

2. Zaman gecikmesi R_0 'ın yüzeyinde kararlı AQM, kapalı-döngü kontrol bant genişliğinde büyük bir limite yerleştirilir ve sonuç olarak geçici cevabın başarılabilir hızınındadır. Kararlı davranış için, kapalı-döngü zaman sabitleri aşağı yukarı $R_0/2$ *sn* tarafından sınırlandırılır.

6.2.2 AQM Performans Hedefleri

Herhangi bir kontrol sisteminin tasarımında, birinci adım performans hedeflerini ortaya çıkarmaktır. AQM için, performans hedefleri, etkin sıra kullanımı, düzenlenmiş sıra gecikmesi ve sağlamlıkdır.

1.Etkin sıra kullanımı *:* Etkin kullanım için, sıra aşırı yükden yada boşlukdan kaçınmalıdır. Önceki durum, boş bir tampon hattan faydalanırken kayıp paketler ve istenmeyen geri iletim sonuçlarını verir. Bu iki durumdan da, geçici ve sağlam durum işlemlerinde kaçınılmalıdır.

2.Sıra gecikmesi*:* Veri paketinin, yönlendirme sırası tarafından servis edilmesi için gereken zaman sıra gecikmesi olarak adlandırılır ve q/C ye eşittir. Bu zaman yayılma gecikmesi T_p ile birlikte, ağ gecikmesinin hesabını verir ve sıra gecikmesi ve dönüşümlerini küçük tutmak arzu edilebilir. Bu küçük sıra uzunluklarını düzeltmek için çağrılır ; bunun yanında, böyle yapmak hatta kullanım altında ve bu sınırlamayla AQM tasarımında temel bir değişimle sonuçlanabilir.

3.Sağlamlık *:* AQM şemaları, kapalı-döngü performansını devam ettirmek için ağ şartlarının görüntüsünde değişikliğe ihtiyaç duyabilir. Bu şartlar, TCP oturumlarının

sayısındaki değişimler *N,* yayılma gecikmesindeki değişimler T_p ve kısa yaşamlıların sıraya tanıtılmasıdır.

6.2.3 RED Tasarımı

Bir aktif sıra yönetim sistemi (AQM) şekil 6.6'da gösterildiği gibi bir geri besleme kontrol sistemi olarak modellenebilir. Burada $P(s)e^{-sR_0}$ daha önceden türetilen TCP sıra dinamiklerinin küçük-sinyal doğrusallaşmasını belirtir (sıra uzunluğu q_0 civarında doğrusallaştırılır). *P(s)* daha önceden türetilen $P_{tcp}(s)$ $P_{sira}(s)$ dır. *δp ve δq* kaybolma olasılığı ve sıra uzunluğundaki karışıklığı belirtir. Şekil 6.6'da transfer fonksiyonu *C(s)* tail-drop yada RED gibi bir AQM kontrol stratejisini belirtir.

Tail-drop bir açma-kapama kontrol stratejisidir. Şekil 6.6'da ayarlarımız cinsinden, tail-drop miktarları açık-kapalı hareketi için *δp Є {0,1} dir.* Kontrol teorisinden böyle bir açık-kapalı mekanizması salınımlara sebep olur, bu salınımlar karmaşık ve kaos davranışlarını göstermektedir [56]. Böyle salınımlar sıra yönetiminde istenmeyebilir ve RED bunları düzeltmek için tanıtılmıştır.

RED için bir transfer fonksiyon modeli ,

$$C(s) = C_{red}(s) = \frac{L_{red}}{s/K + 1}, \qquad (6.7)$$

Burada

$$L_{red} = \frac{p_{\max}}{\max_{th} - \min_{th}}; \qquad K = \frac{\log_e(1-a)}{\delta},$$

$\alpha > 0$ sıra ortalama parametresidir ve δ örnek zamandır [54]. $C_{red}(s)$ tasarımında AQM kontrol sistemini düzeltmek için, hem TCP oturumlarının sayısındaki değişim *N* ve hemde RTT R_0 hesap içine alınmalıdır. R_0 daki değişimler yayılma zamanı değişkeni T_p yüzündendir. Burada

$$R_0 = \frac{q_0}{C} + T_p$$

TCP oturumlarının sayısı için bir sınıf varsayalım ve $N \geq N^-$ ve RTT $R_0 \leq R^+$ olduğunu söyleyelim. RED parametreleri L_{red} *ve* K 'yı seçmek için eşitlik (6.7) hedef şekil 6.6 da tüm N ve R_0 için doğrusal kontrol sistemini düzeltmektir. Eğer sınırlanmış dış girişler sadece sınırlanmış çıktılar üretirse, şekil 6.6 daki doğrusal geri besleme kontrol sistemi kararlıdır. Bu gelmiş olan, ilk şartlara cevapları gerektirir, sınırlandırılabilir ve çarpansal olarak sıfırda birleştirilebilir. Kararlılığın bu tanımı altında, aşağıdaki iki önermeyi verebiliriz.

Önerme 6.1: L_{red} *veK aşağıdaki koşulu sağlar,*

$$\frac{L_{red}(R^+C)^3}{(2N^-)^2} \leq \sqrt{\frac{w_q^2}{K^2}+1}; \qquad (6.8)$$

Burada

$$w_q = 0.1\min\left\{\frac{2N^-}{(R^+)^2 C}, \frac{1}{R^+}\right\}. \qquad (6.9)$$

Daha sonra, şekil 6.6 daki geri besleme kontrol sistemi $C(s) = C_{red}(s)$ kullanarak eşitlik (6.7) her $N \geq N^-$ *ve her* $R_0 \leq R^+$ için kararlıdır.

İspat : Telafi edilen transfer fonksiyonu döngüsünün frekans cevabını düşünelim

$$L(jw) \doteq C_{red}(jw)P(jw)e^{-jwR_0}$$

$$= \frac{L_{red}\dfrac{(R_0C)^3}{(2N)^2}e^{-jwR_0}}{(\dfrac{jw}{K}+1)(\dfrac{jw}{\dfrac{2N}{R_0^2C}}+1)(\dfrac{jw}{\dfrac{1}{R_0}}+1)}$$

Bu ve eşitlik (6.9)'dan, şuna sahibiz,

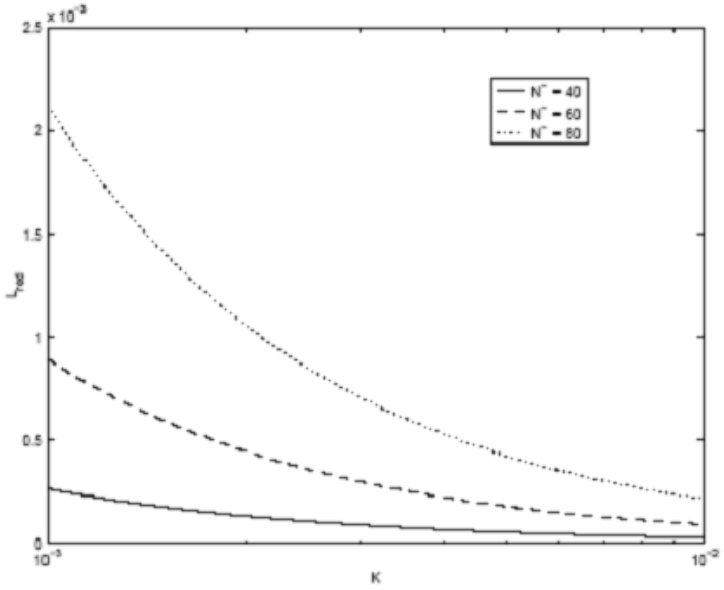
$$L(jw) \approx \frac{L_{red}\dfrac{(R_0C)^3}{(2N)^2}e^{-jwR_0}}{\dfrac{jw}{K}+1}, \qquad \forall w \in [0, w_q]$$

Şimdi verilen her $N \geq N^-$ *ve* her $R_0 \leq R^+$ için,

$$\left|L(jw_q)\right| \leq \frac{L_{red}\dfrac{(R^+C)^3}{(2N^-)^2}}{\sqrt{\dfrac{w^2}{K}+1}}.$$

Bundan ve eşitlik (6.8)'den, bu $|L(jw_q)| \leq 1$ her $N \geq N^-$ ve her $R_0 \leq R^+$ için, takip eder. Bundan dolayı, birleşen-kazanç frekans ile kesişir üstten w_q ile sınırlanır. Kapalı-döngü kararlılığını ayarlamak için, Nyquist kararlılık kriterini isteriz [12] ve $\angle L(jw_q) > -180^0$'i gösteririz. Bu sonda, tekrar eşitlik (6.9)'u aşağıdaki eşitliği elde etmek için kullanırız,

$$\angle L(jw_q) \geq \angle \frac{K_{red}\dfrac{(R^+C)^3}{(2N^-)^2}}{\dfrac{jw_q}{p_{red}}+1} - w_g R_0 \geq -90^0 - 0.1\frac{180^0}{\pi} > 180^0$$



Şekil 6.7. *Kararlı RED parametreleri L_{red} ve K.*

Hatırlatmalar 3:

1. Bu seçimin parametrelerinin arkasındaki mantık $C_{red}(s)$ i baskın kapalı-döngü davranışına zorlar. Bu, TCP zaman sabiti $\frac{(R^+)^2 C}{2N^-}$ yada sıra zaman sabiti R^+ dan birinden büyük kapalı-döngü zaman sabiti $(\approx 1/w_g)$ nin işaretlenmesi ile yapılır.

2. (L_{red},K) nın farklı seçimleri yukarıdaki şartı sağlar. Örneğin, $R^+=0.25$ sn, $N^-=40,60$ ve 80 akış ve $C=3750$ paket/sn olduğu zaman kabul edilebilir parametrelerin bir bölgesi şekil 6.7'de gösterilmiştir.

3. Bu RED tasarımı doğrusal olarak ağ parametre değişimlerine $N \geq N^-$ ve $R_0 \leq R^+$ sağlamdır. Genişletmek için aşağıdaki önerme 6.2 'de tanımlanan parametrelerin değişimine ek olarak bu geri besleme kontrol sistemi kararlıdır.

4. w_q nun seçiminde bu 0.1 artan parçası kararlılık payını sağlar. Eğer 0.1'den daha büyük bir değer seçersek, daha düşük bir kararlılık payıyla bir düzenleyici üretiriz.

Daha agresif olan tasarımın faydası daha hızlı cevap zamanları vermesidir (w_q daki artış yüzünden).

5. N^- den büyük tüm yük seviyeleri için sistemin kararlı olması sezgisel sayaç gibi görünmektedir. Aslında, eğer yük seviyeleri kaybolma profilinin süreksizlik bölgesinde bulunan işletim noktasının bulunduğu bölgeye sistemi sokarsa, sistem salınabilir. Bu kaynak [49] da ele alınmıştır. Bunun yanında gentle-mekanizma, kaynak [60]'da kararsızlıkla alakalı süreksizliği kaldırır.

6. Yüksek yük seviyelerinde, kaybolma olasılığı bazı akışların zaman aşımına gitmesi için yeterince yüksek olur. Modelimizde ve analizimizde zaman aşımlarını yok saydık. Zaman aşımları, analizimizde kararlılığı etkilememelidir; hakikaten sistemi daha az salınıma eğilimlendirirler.

7. Sunulan analizler RTT R^+ üzerinde bir üst sınır olarak düşünüldü. Bunun yanında, eğer bazı akışların RTT'leri bu sınırı aşarsa sistem kararsız olur. Aslında, karışık RTT'lerin önünde, akışların RTT'lerine ne eşdeğer çağırıyorsak bu sınır yorumlanmalıdır. Basit durumlar (tek daralan kısım) için, RTT akışların kişisel RTT'lerinin harmonik ortalamasıdır. N akışlı karışık RTT R_i 'ye sahip bir senaryo düşünelim. RTT'nin harmonik ortalaması (R_{eq}) şu şekilde verilir ,

$$\frac{1}{R_{eq}} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{R_i}$$

Şimdi, daraltılmış kısımdaki routerda, kapasite farklı akışlar tarafından paylaşılır. Bundan dolayı, dengede, zaman aşımlarını yoksayarak ve işlem hacmi için kaynak [61,62] basitleştirilmiş $\sqrt{2}$ formülü kullanılır ve şuna sahip oluruz,

$$C = \sum_{i=1}^{N}\frac{\sqrt{2}}{\sqrt{p_0}R_i}$$

$$= \sqrt{\frac{2}{p_0}} \sum_{i=1}^{N} \left(\frac{1}{R_i} \right)$$

$$= \sqrt{\frac{2}{p_0}} \left(\frac{N}{R_{eq}} \right)$$

Bundan dolayı ortalamadaki sistem davranışı N akışlı bir sistem olarak, her biri benzer RTT eşitliklere sahiptir. Buda R_{eq} dur.

Önerme 6.2 : Herhangi bir RED düzenleyicisi $C_{red}(s)$'in önerme 6.1 içerisindeki şartları eşitlik (6.8) ve eşitlik (6.9) sağladığını düşünelim. Daha sonra, şekil 6.6 daki doğrusal kontrol sisteminin kazanç payı (GM) ve faz payı (PM)'dır.

$$GM \geq 5\pi; \qquad PM \geq 85^0$$

Sonuç olarak, eğer $R_0 < 15R^+$ yada $N>(1/5\pi)N^-$ olduğunda, bu doğrusal kontrol sistemi kararlı olarak kalır.

İspat: Önerme 6.1'deki ispatında yapılanların bir faz hesaplamasını kuvvenlendirmesi şunu verir

$$\angle L(jw_q) \geq \angle \frac{K_{red} \dfrac{(R^+C)^3}{(2N^-)^2}}{\dfrac{jw_q}{p_{red}} + 1} - w_g R_0 \geq -90^0 - 0.1\frac{180^0}{\pi} \approx -95^0$$

Bundan dolayı, $PM = 180^0 + \angle L(jw_g) \geq 85^0$. Fazın yavaşlaması ilave edilen RTT gecikmesi ΔR den dolayıdır,

$$\angle e^{-jw_g \Delta R} = -w_g \Delta R.$$

Eşitlik (6.9)'dan, $w_q \leq 0.1/R_0$ dır. Bunu ve $w_g \Delta R = 85(\frac{\pi}{180})$ kullanılması $\Delta R \leq 14.8R$ yi verir. Kazanç payı hesaplaması için önerme 6.1'in ispatından şunu tekrar çağırırız,

$$P(jw) \approx \frac{L_{red} \frac{(R_0 C)^3}{(2N)^2}}{\frac{jw}{K} + 1}, \qquad \forall w \in [0, w_q]$$

Sonuç olarak,

$$\angle P(jw_g) \geq -90^0.$$

Bundan dolayı

$$\angle e^{-jw_g R_0} = -90^0$$

Sonra $\angle L(j\frac{\pi}{2R_0}) \geq -180^0$. Çünkü $|L(jw_g)| \leq 1$, $1 \Big/ \left|L(j\frac{\pi}{2R_0})\right| \approx \frac{\pi}{2R_0}$ kazanç payı için daha düşük bir sınır verir. $w_g \leq 0.1R_0$ olduğundan $GM \geq 5\pi$ dir.

Örnek 1: Ağ parametresi *C=3750 paket/sn,* $N^- = 60$ *ve* $R^+ = 0.2$ *sn* durumunu düşünelim. (6.9) 'dan,

$w_q = 0.1 \min\{0.5259, 4.0541\} = 0.053$ *rad/sn.*

K=0.005 için, eşitlik (6.8)'den şöyle hesaplarız,

$$L_{red} \leq \angle \frac{(2N^-)^2}{(R^+C)^3} \sqrt{\frac{w_g^2}{K^2} + 1} = 1.86(10)^{-4}$$

Bundan dolayı C_{red} için bir seçenek şudur,

$$C_{red}(s) = \frac{1.86(10)^{-4}}{\frac{s}{0.005}+1}$$

Gerçekleşme cinsinden, $C_{red}(s)$ ye aşağıdaki gibi kırabiliriz.

$$L_{red} = 1.86(10)^{-4}; \quad K = 0.005$$

Şimdi, 3750 paket/sn'lik bir hat kapasitesi için,$\delta = 2.66(10^{-4})$, α kazancı, ortalama ağırlık olarak $1.33(10^{-6})$ dır. $L_{red} = p_{max} / (max_{th}\text{-}min_{th})$ dır. Bundan dolayı, eğer p_{max}'ı 0.1 olarak seçersek, ortalama sıra uzunluğunun dinamik sınıfı yaklaşık 540 paket olur.

Hatırlatmalar 4:

1. Salınımsız durum düzeltmesi açısından, L_{red} i mümkün olduğunca büyük seçmek istenen bir durumdur. Salınımsız durum düzeltmesiyle, $\delta q,nın$ salınımsız durumda 0'a düşürülmesi gerektiğini söylemek istiyoruz. Bunun yanında RED mekanizması altında sıra uzunluğunu kararlı sistem için) salınımsız durum değeri ağ şartlarına göre değişir. Bundan dolayı doğrusal modelimizdeki δq istenmeyen bir özellikde olsa asla sıfıra gitmez. Bu salınımsız durum hatasını K yı düşürerek azaltabiliriz. Eşitlik (6.8)'den, $K \rightarrow 0$, $L_{red} \rightarrow \infty$ 'e izin verir. bu sınırlama durumunda şuna sahip oluruz.

$$C_{red} = \frac{K}{s}$$

Bu klasik bütünleştirme telafisine karşılık gelir.

2. Sıra uzunluğunu düzeltmek için RED kullanmanın güçlüğü, düşük bir kontrol bantgenişliğine sahip olmasıdır w_g , sıra yada TCP dinamiklerinden birisinin bant genişliğinden daha düşük olmak zorundadır. Sonuç olarak, kapalı-döngü cevaplar

yeterli miktarda yavaştır. Bu RED'de telafi kılavuzunun tanıtılmasıyla geliştirilebilir. Bu sonuç klasik orantılı-bütünleşme (proportional-integral, PI) telafisidir,

$$C_{PI} = K_{PI} \frac{(s/z + 1)}{s}$$

Böyle bir telafi edici kaynak [27]'de tartışılmıştır.

BÖLÜM 7

REM

Bu bölümde aktif sıra yönetim şeması için geliştirilen REM'i inceliyoruz . REM aşağıdaki anahtar özelliklere sahiptir.

1. Oran eşleştirme temiz tampon : Tampon temizken (küçük bir hedef etrafında kararlı sıralar), kullanıcı sayısını önemsemeden, kullanıcı oranlarını ağın bant genişliği kapasitesine eşleştirmeye çalışır.

2. Ücretlerin toplanması : Uçtan uca işaretleme (yada düşme) olasılığı, basit ve kesin bir tarzda, kullanıcının yolundaki tüm yönlendiricilerin üzerindeki toplanan hat ücretlerinin (sıkışma ölçüsü) toplamı tarafından dikkatle incelenir.

İlk özellik, alışıldık bilimin aksine, yüksek kullanımın ağ da geniş geciktirilmiş işlerin tutulmasıyla başarılmadığını, ama kullanıcı için oranlarını ayarlayarak doğru geri besleme ile başarılabilmesini sağlar. Kullanıcı sayısının artmış olsa bile REM ihmal edilebilir kayıp yada sıra gecikmesiyle yüksek kullanım sağladığını benzetme sonuçları ile gösterdik.

İkinci özellik, kullanıcıların çoklu sıkışmış hat boyunca ilerlediği ağ içerisinde önemlidir. Kullanıcı tarafından dikkatle izlenen uçtan uca işaretleme(yada düşme) olasılığı içerisinde gömülü sıkışma bilgisinin anlamını açıklaştırır ve bu yüzden uyum oranının tasarımında kullanılır.

Şimdi, REM'i ve bu iki özelliği nasıl başardığını açıklayacağız. RED'le keskin bir biçimde farklıdırlar [4]. Bu özellikler birbirinden bağımsız ve diğeri olmadan gerçekleştirilebildiğinde açık olacaktır. Daha sonra Drop Tail, RED ve REM'in kablolu ağlardaki performanslarını benzetmelerle karşılaştıracağız. TCP'nin kablosuz ağlarda performansının kötü olduğu bilinir çünkü tampon taşması yüzünden oluşan kayıplar ve zayıflama, parazit ve karışma gibi kablosuz etkiler yüzünden oluşanlar birbirinden ayrılamaz. REM'in bu probleme nasıl yardımcı olduğunu ve performansını benzetme sonuçlarıyla açıklayacağız.

7.1 RED'in Değerlendirilmesi

AQM'lerin ana amacı, kaynaklar için oranlarını ayarlayarak sıkışma bilgisini sağlamaktır. AQM algoritmasının tasarımı üç soruya cevap vermek zorundadır.

1. Sıkışma nasıl ölçülür ?
2. Olasılık fonksiyonunda ölçü nasıl gömülür ?
3. Kullanıcıya nasıl geri bildirim yapar ?

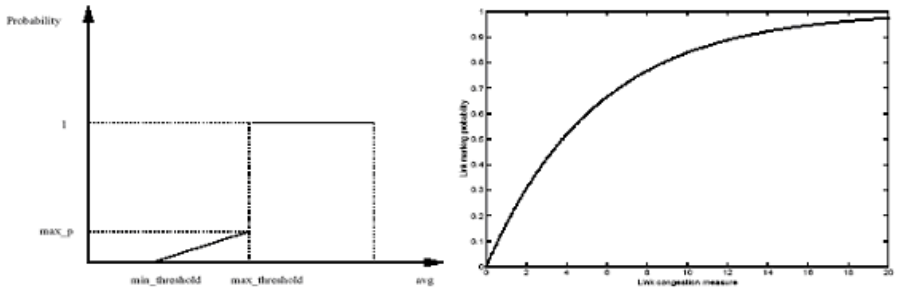
RED bu sorulara şöyle cevap verir.

İlk önce, RED sıkışmayı sıra uzunluğu (çarpansal ağırlıklı ortalama) ile ölçer. Önemle, sıkışma ölçüsünün seçimi sıkışma yansımasının nasıl güncellendiğine karar verir ve bundan dolayı TCP tarafından dahili olarak en iyi şekilde kullanılıyor olan kullanıcı araç (utility) fonksiyonunu etkiler [26]. İkincisi, olasılık fonksiyonu bir parça doğrusaldır ve şekil 7.1(a)'da gösterildiği gibi sıkışma ölçüsünün fonksiyonu artar. Son olarak, sıkışma bilgisi ya düşme yada paket işaretleme olasılığı tarafından kullanıcıya taşınır. Aslında RED sadece ilk iki soruya karar verir. Üçüncü soru büyük ölçüde bağımsızdır.

RED, TCP ile etkileşir, kaynak oranı arttıkca, sıra uzunluğu büyür, daha fazla paket işaretlenir, kaynakların oranlarını ve devir tekrarlarını düşürür. AQM, sıkışma ölçüsünün nasıl güncellendiğini tanımlarken, TCP kesin kaynak oranlarının nasıl sağlandığını tanımlar. RED için, sıkışma ölçüsü sıra uzunluğudur ve tampon işlemi tarafından otomatik olarak güncellenir. Sıra uzunluğu sonraki periyod da şimdiki sıra uzunluğu artı, toplam giriş eksi çıkışdır.

$$b_l(t+1) = [b_l(t) + x_l(t) - c_l(t)]^+ \qquad (7.1)$$

burada $[z]^+ = max\{z,0\}$ dır. $b_l(t)$, l sırası ve t periyodunda toplam sıra uzunluğudur, $x_l(t)$ sıraya l,t periyodunda toplam giriş oranıdır ve $c_l(t)$, t periyodunda çıkış oranıdır.



Şekil 7.1. *Sıkışma ölçüsünün bir fonksiyonu olarak işaretleme olasılığı.*

REM, sadece ilk iki tasarım sorusuyla RED'den ayrılır, farklı bir sıkışma ölçüsü tanımı ve farklı bir olasılık fonksiyonu kullanır. Bu farklılıklar son bölümde bahsettiğimiz, şimdi açıklayacağımız iki anahtar özelliği oluşturmaktadır.

Bu bölümün kalan kısmı için aksi belirtilmedikçe işaretleme demekle, ya bir paketin düşmesi yada ECN bitinin [20] ayarlanma olasılığını demek istiyoruz.

7.2 Random Exponential Marking (REM)

Şimdi REM'in ilk soruyu nasıl cevapladığını açıklayacağız. Ayrıntılı türeme ve gerekçeler, bir sözde kod gerçekleştirimi ve daha genişletilmiş benzetmeler [40,64] de bulunabilir.

7.2.1 Eşleşme Oranı Temiz Tampon

REM'in ilk düşüncesi, hem giriş oranını hat kapasitesi etrafında ve hemde sırayı küçük bir hedef etrafında, hattı paylaşan kaynakların sayısını önemsemeden kararlı hale getirmektir.

Herbir çıktı sırası için REM, sıkışma ölçüsü olarak, 'ücret' diye çağrılan bir değişken sağlar. Bu değişken sonraki alt bölümde açıklandığı gibi işaretleme olasılığını elde etmek için kullanılır. Ücret periyodik olarak yada eş zamanlı olmadan, oran eşleşmesine (yani giriş oranı ve hat kapasitesi arasındaki fark) ve sıra eşleşmesine (yani sıra uzunluğu ve hedef arasındaki fark) dayanarak güncellenir. Eğer bu eşleşmelerin ağırlıklı toplamı pozitifse ücret arttırılır, aksi takdirde düşürülür. Ağırlık toplam, giriş oranı hat kapasitesini aşarsa yada açıklaştırılmış giriş gecikmesi varsa pozitifdir aksi takdirde negatifdir. Kaynakların sayısı arttığı zaman, oranlardaki eşleşmeler ve sıradaki büyüme ücreti yükseltir ve bundan dolayı işaretleme olasılığı yükselir. Kaynaklara, daha sonra oranlarını düşüren, bir sıkışma sinyali gönderir. Kaynak oranları çok küçük olduğu zaman eşleşmeler negatif olur, sonunda eşleşmeler sıfıra yöneltilinceye, dengede ihmal edilebilir kayıp ve gecikme ile yüksek kullanım kazanıncaya kadar, ücret ve işaretleme olasılığı düşürülür, kaynak oranı yükselir. Eğer hedef sıra, sıfıra ayarlanmışsa denge durumunda tampon temizlenir.

Oysaki RED'de sıkışma ölçüsü (sıra uzunluğu), tampon işlemi ile eşitlik (7.1)'e göre otomatik olarak güncellenir, REM açıkca ilk özelliğini yerine getirmek için ücretinin güncellenmesini kontrol eder. Tam olarak, l sırası için, ücret $p_l(t)$, t periyodunda şuna göre güncellenir,

$$p_l(t+1) = \left[p_l(t) + \gamma(\alpha_l(b_l(t) - b_l^*) + x_l(t) - c_l(t))\right]^+ \qquad (7.2)$$

$\gamma > 0$ ve $\alpha_l > 0$ küçük sabitler ve $[z]^+ = max\{z,0\}$ dır. Burada $b_l(t)$, l sırasında t periyodunda bulunan toplam tampondur ve $b_l^* \geq 0$ hedeflenen sıra uzunluğudur, $x_l(t)$, t periyodunda l sırasına toplam giriş oranıdır ve , $c_l(t)$, t periyodunda l sırasına mevcut bant genişliğidir. $x_l(t) - c_l(t)$ farkı oran eşleşmesini ölçer ve $b_l(t) - b_l^*$ farkı sıra eşleşmesini ölçer. α_l sabiti kişisel olarak herbir sıra tarafından ayarlanabilir ve iletim sırasında kullanımı ve sıra gecikmesini değiştirir. γ sabiti ağ şartlarında değisen REM'in yanıt vermesini kontrol eder. Bundan dolayı, eşitlik (7.2) den, eğer oranın ağırlıklı toplamı ve sıra eşleştirilirse , α_l tarafından ağılıklandırılırsa, ücret arttırılır ve pozitifdir, aksi takdirde düşürülür. Denge durumunda, ücret kararlılaşır ve ağırlıklı toplam sıfır olmak zorundadır. Yani, $\alpha_l (b_l - b_l^*) + x_l + c_l = 0$ dır. Bu sadece giriş oranı kapasiteye eşitse ($x_l = c_l$) elde edilebilir ve gecikmiş iş hedefe eşittir ($b_l = b_l^*$) , bölümün başında bahsedilen birinci özelliğe kılavuzluk eder.

Gerçekleştirmede iki hatırlatma yapacağız. Birincisi, REM, özellikle herbir akış bilgisi gerekmiyorsa, sadece yerel ve toplu bilgileri kullanır ve servis disiplinini koruyan her iş ile çalışabilir. Diğer yönlendiriciler ve sıralardan bağımsız olarak ücretini günceller. Bundan dolayı karmaşıklığı kaynakların sayısından yada ağın büyüklüğünden ya da kapasiteden bağımsızdır.

İkincisi, genellikle pratikde, sıra uzunluğunu örneği, orandan daha kolaydır. Hedeflenen sıra uzunluğu b^* sıfır olmadığı zaman, ücret artışında oran eşleşmesinin ölçüsünü $x_l(t) - c_l(t)$, eşitlik (7.2)'deki gibi atlayabiliriz. $x_l(t) - c_l(t)$ sıra uzunluğu büyürken tampon boş olmadığı zamanki orandır. Bundan dolayı bu terimi gecikmedeki değişiklikler tarafından yaklaşık olarak tahmin edebiliriz, $b_l(t+1)-b_l(t)$. Daha sonra güncelleme kuralı eşitlik (7.2) gibi olur ,

$$p_l(t+1) = \left[p_l(t) + \gamma(b_l(t+1) - (1-\alpha_1)b_l(t) - \alpha_l b^*)\right]^+ \qquad (7.3)$$

Yani, ücret sadece şimdiki ve önceki sıra uzunluklarına bağlı olarak güncellenir.

Eşitlik (7.2) ve eşitlik (7.3)'de ifade edilen güncelleme kuralı RED'le tamamen çelişir. Kaynakların sayısı artmasıyla işaretleme olasılığı artmalıdır böylece sıkışma sinyalinin yoğunluğu artar. Bundan dolayı RED, işaretleme olasılığını elde etmek için sıra uzunluğunu kullanır, yani, kaynakların sayısı artarsa, ortalama sıra uzunluğu kararlı bir biçimde artmalıdır. Çelişki olarak, güncelleme kuralı eşitlik (7.3)'de, işaretleme olasılığını elde etmek için kullanılan bir ücreti güncellemek için sıra uzunluğunu kullanır. Bundan dolayı REM altında, ortalama sıra uzunluğu, hedef b_l^* etrafında kararlaştırılmışken ücret kararlı bir şekilde artar, kaynakların sayısı artar. Bu noktaya aşağıda geri döneceğiz.

7.2.2 Ücretlerin Toplanması

REM'in ikinci düşüncesi bir yol boyunca hat ücretlerinin toplamını, yoldaki sıkışmaların bir ölçüsü olarak kullanmak ve bunu kaynaklardan elde edilebilen, uçtan uca işaretleme olasılığının içine gömmektir.

Çıkış sırası, şimdiki ücret de çarpansal olarak artan bir olasılıkla, sıranın yukarı taraflarında henüz işaretlenmemiş her bir gelen paketi işaretler. Bu işaretleme olasılığı şekil 7.1(b)'de gösterilir. Eğer bir paket düşme bitinin yerine, ECN biti ayarlanmışsa, işareti gidilen yere taşınır ve daha sonra ACK yoluyla kaynağa geri taşır. İşaretleme olasılığının çarpansal formu , bir paketi, çoklu sıkışmış hatları bir uçtan diğerine taşımak için uçtan uca işaretleme olasılığı, yoldaki her hattaki işaretleme olasılığına bağlı olan geniş bir ağda kritikdir. Sadece ve sadece kişisel hat işaretleme olasılığı kendi hat ücreti içinde çarpansal olduğu zaman, bu uçtan uca işaretleme olasılığı, kendi yolunda tüm sıkışmış hatlardaki hat ücretlerinin toplamında çarpansal olarak artar. Bu toplam yoldaki sıkışmanın tam ölçüsüdür. Bundan dolayı uçtan uca işaretleme olasılığına gömülüdür. Kaynaklar tarafından işaretlenen paketlerinin bir parçasından kolaylıkla tahmin edebilir ve oran uyarlamasını tasarlamak için kullanılabilir.

Bir paketin $l=1,2,....,L$ hatları içinde iletildiğini ve t periyodunda $p_l(t)$ ücretlerine sahip olduğunu varsayalım. Daha sonra işaretleme olasılığı $m_l(t)$, l sırasında t periyodunda ,

$$m_l(t) = 1 - \phi^{-p_l(t)} \qquad (7.4)$$

Burada $\phi > 1$ bir sabittir. Daha sonra paket için uçtan uca işaretleme olasılığı şöyledir,

$$1 - \prod_{l=1}^{L} (1 - m_l(t)) = 1 - \phi^{-\sum_l p_l(t)} \qquad (7.5)$$

Yani, yolun sıkışma ölçüsü, $\sum_l p_l(t)$, geniş olduğu zaman, uçtan uca işaretleme olasılığı yüksektir.

Hattın işaretleme olasılığı $m_l(t)$ küçük olduğu zaman, buna bağlı olarak hat ücretleri $p_l(t)$ küçüktür, eşitlik (7.5) de verilen uçtan uca işaretleme olasılığı aşağı yukarı yoldaki hat ücretlerinin toplamına uygundur ,

$$uçdan\ \ uca\ \ isaretleme\ \ olasiligi\ \ \approx\ \ (\log_e \phi) \sum_l p_l(t)$$

7.2.3 Modülleştirilmiş Özellikler

Eşitlik (7.2) ve eşitlik (7.3) ‘de verilen ücret ayarlama kuralı, sıra uzunluğu hedeflenen değer, muhtemelen sıfır etrafında, kararlı hale getirilirken, hat kapasitesi ile kullanıcı oranlarını eşitleme çalışan REM özelliğine kılavuzluk eder. Eşitlik (7.4)’de verilen üstel işaretleme olasılık fonksiyonu, uçtan uca işaretleme olasılığını, yoldaki toplanmış tüm yönlendiricilerin, toplanmış ücretlerini kullanıcıya taşır. Bu iki özellik birbirinden bağımsız olarak gerçekleştirilebilir.

Örneğin, sıkışmayı ölçmek için ücretleri kullanmak seçebilir ama farklı bir işaretleme olasılık fonksiyonu kullanmaktadır. Yani, RED benzeri yada diğer bazı ücretin artan fonksiyonu birinciyi gerçekleştirmek için kastedilmiştir ama ikinci özellik değildir.

Alternatif olarak sıkışmayı ölçmek için farklı bir seçim yapılabilir, yani, kayıp, gecikme yada sıra uzunluğunun kullanılması, ama üstel işaretleme olasılık fonksiyonu ile işaretler. Bu ikinciyi gerçekleştirmek içindir, birinciyi değil.

7.2.4 Sıkışma ve Performans Ölçüleri

Reno, tampon taşmaları ile AQM olmadan sıkışmayı ölçer, Vegas sıra gecikmesi ile ölçer [30], RED ortalama sıra uzunluğu ile ve REM ise ücreti ile ölçer. Aralarındaki kritik fark, ilk üç şemada olduğu gibi kayıp, gecikme yada sıra uzunluğu gibi sıkışma ölçüsü ile performans ölçüsünün birleşmiş olmasıdır. Bu birleşme, kullanıcı sayısı artmasıyla, sıkışıklık büyümesini ve performansın daha kötüleşmesini gerektirir, yani, 'sıkışıklık' ın anlamı, geniş kayıp yada gecikme gibi 'kötü performans' dır. Eğer bunlar REM'deki gibi ayrılırsa, 'sıkışıklık' (yani yüksek hat ücretleri) ağ kaynaklarını desteklediği talepleri aşan sinyalleri basitleştirir. Bu engelleme talepdir ama iyi performans düşük gecikme ve kayıpla devam eder.

Ayırma demekle, sıkışma ölçüsünün denge değerlerini denge kaybı, sıra uzunluğu yada gecikmeden bağımsız hale getirmek kastedilmişdir. Eşitlik (7.3) de, sıra uzunluğu REM'de iletim sırasındaki sıkışma ölçüsününün güncellemesine karar verir ama denge değeri değildir. Kaynakların sayısı büyüdükce REM'deki ücretler büyür ama sıralar hedefler etrafında kararlıdır. Sıkışma ölçüsünün denge değeri, REM'deki ücret ve RED'deki ortalama sıra uzunluğu, yalnız ağ topolojisi ve kaynakların sayısından kaynak [26] elde edilir.

Bundan dolayı, RED altında, kaynakların sayısı ile ortalama sıranın yavaş ve hızlı büyümesi kaçınılmazdır. Esas RED'le , tüm paketlerin işaretlendiği en büyük sıra eşiği max_{th} a kadar büyüyebilir. Eğer max_{th} çok yüksekse, sıra gecikmesi çok aşırı olabilir; eğer çok düşükse, tampon salınımının şiddeti yüzünden hat kullanım altında olabilir. Dahası, eğer sıkışma sinyali işaretlemeden ziyade rastgele düşme boyunca geri beslenirse, paket kayıpları çok sık olabilir. Bundan dolayı, sıkışma zamanlarında, RED ya yüksek hat

kullanımına ulaşmak için yada düşük gecikme ve kayıp için ayarlanır, ama hepsi için ayarlanamaz. Karşıt olarak, sıkışma ve performans ölçülerinin ayrılmasıyla, sıra trafik yükünün bağımsız hedefi etrafında kararlı olabilir, bu dengede yüksek kullanım ve düşük gecikme ve kayıplara kılavuzluk eder. Bunlara ilişkin benzetme sonuçları sonraki bölümlerde vardır.

7.3 Performans

7.3.1 Kararlılık ve Araç Fonksiyonu

Son zamanlarda gösterilen esas TCP sıkışma kontrol şemaları, Reno, Reno/RED, Reno/REM, Vegas, Vegas/RED,Vegas/REM, toplu kaynak yararını en fazla yapabilmek için hepsi yorumlanarak bir eğim algoritmasına aktarılabilir [26, 30]; ayrıca kaynak [38,58] basit bir model için. Farklı TCP şemaları, işaretleme ile yada işaretleme olmadan, sadece kullanıcı araç fonksiyonunun seçiminde farklıdır. Duality modeli bu yüzden, kararlılığı çalışmak için uygun bir yol sağlar, bu şemaların en iyilik ve doğruluk özellikleri ve daha önemlisi, birbirlerine etkileşimleri keşfetmek içindir. Özellikle, eğim algoritmasının eş zamanlı olmayan çevrelerde bile kararlı olduğu matematiksel olarak kanıtlanmıştır [7,10]. Pencere ölçüleri nispeten küçük olduklarında geniş gerçek yaşam ve benzetme deneyimi bu TCP şemaları ile doğrulanır. Ayrıca iki anlamı vardır.

Birincisi, kullanıcılar ne tip bir araç fonksiyonu kullanacaklarını bilmeseler bile, oran ayarlamasının tasarlanması ile belli bir araç fonksiyonu seçmiş olur. Anlaşılır yaparak, iyileştirme modelleri kaynak [26,30,38,58], şimdiki protokollerin anlaşılmasını derinleştirir ve araç fonksiyonunun uygulamaya uydurulmasıyla yeni protokoller tasarlamak için yeni yöntemler önerir.

İkincisi, araç fonksiyonu sadece kullanıcının oran ayarlanmasıyla elde edilmeyebilir, ama ayrıca işaretleme algoritması ile elde edilebilir. Bu Reno için doğrudur, yani, Reno, Reno/RED, Reno/REM biraz farklı araç fonksiyonlarına sahiptir. Bu ihtiyacımızın bir sonucudur, AIMD algoritması tampon taşması yada RED yada REM

yüzünden olup olmadığını önemsemeden, bu şemalarda çok farklı sıkışma ölçülmüş ve gömülmüş olsa bile paket kayıplarına aynı yolla cevap verir.

Son zamanlarda, bir PI (proportional-plus-integral) kontrolörü kaynak [27]'de RED'e alternatif bir AQM olarak önerilmiş ve benzetme sonuçları üst dengesini ve iletim performansını göstermek için sunulmuştur. Eşitlik (7.3) de ifade edilen bu PI kontrolleri ve REM dengededir, kapatılır.

7.3.2 Kullanım,Kayıp ve Gecikme

REM ve RED'in performanslarını karşılaştırmak için Reno ve NewReno ile, tek ve çok hat ile kaynakların çeşitli sayısı, hat kapasitesi ve yayılma gecikmeleri ile geniş benzetmeleri yönettik [40,64]. REM ve RED'in ilgili performansları Reno ve NewReno ile benzer olması beklenir bundan dolayı bölüm 7.1 ve bölüm 7.2'de tartışılan özellikler kaynak algoritmalarından bağımsız AQM özellikleridir. Bu alt bölümde, NewReno/DropTail, NewReno/REM ve NewReno/RED'in performanslarının karşılaştırılmalarının sonuçlarını sunuyoruz [17].

Bant genişliği 64Mbps ve tampon kapasitesi 120 paket olan tek hat için benzetme ns-2 simülatöründe idare edilir. Paketlerin hepsi 1KB'dır. Bu hat 80ms'lik aynı gidiş-dönüş yayılma gecikmesine sahip 160 NewReno kullanıcısı tarafından paylaşılmaktadır. Başlangıçda 0 zamanında 20 kullanıcı aktifdir ve her 50 sn'den sonra 160 kullanıcıya ulaşıncaya kadar 20 kullanıcı daha aktive edilir. RED için parametrelerin iki kümesi kullanılır. İlk küme RED(20:80) 'i belirtir, minimum sıra eşiği min_{th} = 20 paket, maksimum sıra eşiği max_{th} =80 paket ve max_p = 0.1 dir. İkinci küme RED(10:30)'u belirtir, minimum sıra eşiği min_{th} =10 paket, maksimum sıra eşiği max_{th} = 30 paket ve max_p = 0.1 dir. REM'in parametre değerleri $\phi = 1.001, \alpha = 0.1, \gamma = 0.001, b^* = 20$ paketdir. Deneyleri hem işaretleme hemde paket düşmeleri ile sıkışma geri beslemesinin bir yolu olarak yönetiyoruz. Hat algoritması tarafından elde edilen olasılığa göre paketleri işaretliyoruz yada düşürüyoruz [17].

x ekseninde zaman arttıkca, kaynakların sayısı 20'den 160'a artar ve ortalama pencere ölçüsü 32 paketden 4 pakete düşer. y ekseni her bir periyoddaki performansı gösterir. Goodput, hat kapasitesine tüm varış yerlerinde alınan tekrarlanmamış paketlerin toplam sayısının oranıdır. Kaybolma oranı ise toplam düşen paketlerin toplam gönderilen paketlere oranıdır.

REM ile DropTail performansını karşılaştırıldığında, bu deney kümelerinde, hemen hemen her pencere ölçüsü ya düşme oranı yada ECN ile işaretlendiğinde REM, DropTail'dan biraz daha yüksek goodput'a ulaşır. Kaynakların sayısı artarsa, REM ortalama sıra hedeflenen $b^* = 20$ paket

Etrafında, ortalama sıra DropTail altında kararlılığı artarken, kararlıdır. Kayıp oranı, kaynakların sayısını önemsemeden neredeyse sıfır işaretleme ile REM altında hemen hemen aynıdır.

RED'in performansı DropTail ile karşılaştırıldığında, DropTail için goodput RED'in tüm değişikliklerini üst sınırlar, çünkü daha geniş bir ortalama sıra tutar. Ortalama sıra bu 5 şema altında, bölüm 7.2.4'de tartışıldığı gibi, kaynakların sayısı arttıkça kararlı bir şekilde artar. Beklendiği gibi tüm pencere ölçülerinde, RED(20:80), RED(10:30)'dan daha yüksek goodput ve ortalama sıraya sahiptir.

7.4 Kablosuz TCP

TCP (yada daha kesin, AIMD algoritması) esas olarak, tampon taşmaları yüzünden oluşan paket kayıpları tarafından sıkışma ölçülen ve kullanıcılara taşınan kablolu ağlar için tasarlanmıştır. Kablosuz ağlarda, bunun yanında, esas olarak bit hataları, sinyal zayıflaması ve karışma gibi sebeplerden paketler kaybedilir, ayrıca uzaklıkdan dolayı aralılıklı bağlantı da etkilidir. Paket kayıpları ve sıkışma ölçüsü arasındaki birleşme ve TCP'deki geri besleme kablosuz hatlar üzerindeki zayıf performansa kılavuzluk eder. Çünkü TCP kaynağı tampon taşması ve kablosuz etkilerden oluşan kayıpları ayırt edemez ve herbir kayıp olayında penceresini ikiye böler.

Bu problemi çözmek için üç yaklaşım önerlmiştir [12]. İlk yaklaşım, paket kayıplarını kablosuz ağ üzerinde saklar böylece kaynak sadece sıkışmanın teşvik ettiği kayıpları görür. Bu çeşitli karışım tutma teknikleri, hata kontrolü ve yerel geri iletim algoritmaları ile ilgilidir. İkinci yaklaşım, kablosuz etkilerden kaynaklanan kayıpları TCP seçenek alanlarını kullanarak kaynağa haber vermektir, böylece kaynak geri iletim sonrasında oranını yarıya düşürmeyecektir.

Üçüncü yaklaşımın amacı tampon taşmalarından kaynaklanan paket kayıplarını ortadan kaldırmakdır, böylece kaynak sadece kablosuz kayıpları görür. Bu TCP'nin varsayımını bozar. Kayıplar artık tampon taşmasını belirtmez. Sıkışma ölçülmeli ve geri besleme farklı bir mekanizma kullanmak zorundadır. REM'in ilk özelliğinden (oran eşleşmesi temiz tampon) faydalanarak, bu amaç için REM'i ECN işaretleme ile kullanmayı önerilmiştir [17]. Daha sonra bir TCP kaynağı sadece bir kayıp tespit edildiğinde geri iletim yapar ve işaretini gördüğü zaman penceresini ikiye böler.

Şimdi bu yaklaşımın sözünü göstermek için hazırlayıcı benzetme sonuçları sunacağız. Benzetme ns-2 simülatörü içerisinde bant genişliği 2Mbps ve tampon kapasitesi 100 paket olan tek bir kablosuz hat için idare edilmektedir. Paketleri Bernoulli kayıp modeline göre %1 olasılıkla rastgele düşürür (kaynak [40] daki bursty kayıp modeli ile benzetmeler). Rastgele düşürmenin etkisini azaltmak için küçük bir paket büyülüğü olarak 382 bit seçilmiştir. Bu kablosuz hat, 100 NewReno kullanıcısı tarafından aynı gidiş-dönüş yayılma gecikmesi (80ms) ile paylaşılır. 20 kullanıcı 0 zamanında ilk başta aktifleştirilir ve her 50sn sonra 100 aktif kullanıcıya ulaşıncaya kadar 20 kullanıcı daha aktifleştirilir.

AQM ile ECN biti ns-2'de 1'e ayarlanır böylece paketler olasılıksal olarak RED yada REM'e göre işaretlenmiştir. Paketler sadece dolu bir tampona ulaştıklarında düşürülür. NewReno'yu düzenledik böylece bir işaret aldığında yada zaman aşımından bir kayıp tespit ettiğinde penceresini ikiye böler, ama tekrarlanan doğrulamadan bir kayıp tespit ettiğinde pencereyi ikiye bölmeden geri iletir. NewReno'nun (DropTail ile), (düzenlenmiş) NewReno ile RED ve (düzenlenmiş) NewReno ile REM'in

performanslarını karşılaştırdık [17]. Önceki bölümde RED ve REM parametreleri aynı değere sahiptiler.

ECN işaretlemenin girişi NewReno'nun goodput'unu geliştirmede çok etkili olduğunu gösterir, %62 ve %91 arasından %82 ve %96 arasına kadar yükseltir, kullanıcının sayısına bağlıdır. REM ve RED'in karşılaştırması kablolu ağlardakine benzer bir sonuçtur. REM ve RED(20:80), RED(10:30)'dan(%82-%95 arası) daha yüksek bir goodput'u (%90-%96 arası) devam ettirir. Kaynakların sayısı arttıkça ortalama sıra REM altında, DropTail ve RED altında kararlı bir şekilde artarken kararlıdır.

Bu olağanüstü durum, sadece tampon taşması yüzünden olan kümülatif paket kayıplarında açıkca gösterir. Kayıp NewReno ile en hafiftir, RED(10:30) ve REM ile ihmal edilebilir ve RED(20:80) ile ılımlılaştırılmıştır. REM ve RED(10:30) altında tampon taşmaları sadece izleyen yeni kaynakların tanıtılmasının iletimi sırasında olur, ve bundan dolayı kümülatif kayıplar herbir periyodun başında zıplar ama zıplamalar arasında sabit kalır. RED(20:80) ve NewReno altında, diğer taraftan, tampon taşmaları da dengededir. Bundan dolayı kümülatif kayıplar zıplamalar arasında karalı şekilde artar.

Bu yaklaşımla karşı olarak , bazı ama hepsinde değil karışık (heterojen) ağlardaki uygulamalarda yönlendiriciler ECN'ye yatkındır. ECN'ye yatkın olmayan yönlendiriciler düşmede geri besleme sıkışmasına güvenmeye devam eder. Oranlarını sadece işaretlere dayanarak uyarlayan TCP kaynakları bu yönlendiricilerin aşırı yüklerinin riskini idare eder. Yönlendiriciler için olası çözüm ECN kapasitelerinin nasıl olduğunu belirtmek, muhtemelen kaynak [20] de önerilen iki ECN bitinin birisinin kullanımını sağlar. Bu tüm yönlendiricilerin en azından ECN'den haberdar olmasını gerektirebilir. Bir kaynak sadece yoldaki tüm kaynakları ECN'den haberdar ederek işaretlemek için tepki verir ama yoldaki bir yönlendirici ECN'den haberdar değilse, kayıba bilindik TCP kaynakları gibi tepki verir.

BÖLÜM 8

TCP Reno ile Paket Kayıplarının Kurtarılmasının Analitik Modelleri

İletim Kontrol Protokolü (TCP) internette geniş ölçüde taşıma katmanı(transport layer) olarak kullanılır. Çünkü TCP, kablolu bilgisayar ağlarındaki paket kayıp olasılıklarının ihmal edilebilecek kadar düşük olması kabullenmesi üzerine dizayn edilmiştir [65]. Ancak TCP, kablosuz ağ sistemleri için yüksek hata ihtimalleri ile birlikte anılır olmuştur [65,66].

TCP kullanılan kablosuz bağlantılar için performans düşüşleri tıkanıklık bulunmayan paket kayıpları ve tekrar hızlı iletim zaman aşımı (RTO) sıklığı sonucu oluşan gereksiz tıkanıklık kontrolleri ile açıklanabilir. RTO gerçekleştiğinde, özellikle gönderici sadece bilgi geçersiz olana kadar gönderememekte, ancak iletimi tekrar başlatmak için yavaş başlama yapmalıdır. Bu yüzden RTO gerçekleşmeden önce tıkanık pencerenin tekrar düzeltilmesi uzun zaman almaktadır. Sonuç olarak, yüksek RTO sıklığı, TCP performansını kötü yönde etkilemektedir [9,66,69].

Tıkanıklık bulunmayan paket kayıp durumlarında TCP performansını analizi için birçok çalışma yapılmıştır .[24,66]. Bu çalışmalar göstermiştir ki, sonucu TCP Reno performansı hızlı tekrar iletim olasılığına bağlıdır [67]. Ancak biz daha çok TCP Reno davranışlarındaki kayıpların kurtarılmasına ve bir pencerede gerçekleşen paket kaybı sayısı kriterine göre tekrar hızlı iletim olasılığı üzerinde duracağız. Özellikle ilişkili ve rasgele durumlardaki paket kayıplarının tipik niteliklerini inceleyeceğiz.

Öncelikle TCP tıkanıklık penceresi genel olarak periyodik bir yapı göstermektedir. Bu Markov zinciri ile analiz edilebilir [66,69]. Böylece durağan bir dağılım gösteren pencere işlemini sayısal olarak hesaplayabiliriz.

8.1 TCP Reno ile Kayıpların Kurtarılması

TCP Reno ile kayıp paketlerin kurtarılması için iki yol mevcuttur; birincisi RTO ile ve diğeri de tekrar hızlı iletim ve hızlı kurtarım iledir. Hızlı tekrar iletimi tetiklemek için gönderici bir kayıp paket için en azından K tane çift alındı (Acknowledgement, ACK) almalıdır (tipik olarak üç çift ACK).

t zamanında, göndericinin tıkanık penceresini ve yavaş başlama eşiğini belirtmeliyiz *ssthresh*, ile $W(t)$ ve $Wth(t)$. Varsayalım ki, paket kayboldu ve gönderici $t = t_0$ paketi için K.ncı çift ACK aldı. Kayıp paket tekrar hızlı iletim yöntemi ile vakit kaybetmeden tekrar iletildi. Tekrar hızlı iletimden sonra, gönderici şunu hazırlar

$$W(t_0^+) = \left[\frac{W(t_0)}{2}\right] + K, \quad W_{th}(t_0^+) = \left[\frac{W(t_0)}{2}\right] \qquad (8.1)$$

Tekrar hızlı iletim süresince gönderici her çift ACK alınca pencere boyutlarını birer birer arttırır. Eğer büyütülmüş pencere yeni bir paket içeriyorsa, göndericinin bu paketi göndermesine izin verilir. Tekrar gönderilmiş paket başarılı bir şekilde iletildiyse, normal(çift olmayan) ACK üretilir. Böylece tıkanmış pencere hemen iletilecek pakete kaydırılmış olur ve hızlı kurtarma işlemi sonlanmış olur. Gönderici tıkanıklığın önlendiği paketleri ile tıkanık pencereyi (ki bu da eşitlik (8.1) de belirlenmiş W_{th} 'ye eşittir) göndermeye devam eder.

Eğer bir penceredeki birden fazla paket kaybolduysa, birkaç tekrar hızlı iletim ve hızlı kurtarma işlemi tekrarlanabilir. Bir pencere için n tane kayıp paket n tane hızlı tekrar iletim ile kurtarılabilir. Eğer ilk tekrar hızlı iletim hemen önceki zaman t_1 ise ve n.nci kayıp paket t_2'de tekrar hızlı transfer yapıldıysa, $W(t_1)$ ve $W(t_2)$ arasındaki ilişki aşağıdaki gibi olur [24],

$$W(t_2) = \left[\frac{W(t_1)}{2^n}\right] \qquad (8.2)$$

8.2. Modelleme

TCP hareket biçiminde kayıpların kurtarılmasını kaynak [69]'da tanımlanan turlar (rounds) cinsinden modelleyelim. Pencerenin durağan dağılımını bulmak ve kayıp kurtarılması olasılıklarını çıkartabilmek için, Markov Zinciri analiz yöntemini kullanacağız [67].

8.2.1 Kabullenmeler ve Tanımlamalar

Göndericinin göndereceği sonsuz sayıda paket olduğundan tıkanık pencereler kesinlikle artarak devam edecektir. Bütün paketlerin aynı boyutlara sahip olduğunu kabul edelim. Bu durumda başarılı paket iletimlerinin sonucunda gönderici her zaman bir ACK alacağından, gecikmiş ACK durumunu dikkate almıyoruz [63]. Bilgi paketi boyutlarına göre ACK paketinin boyutu göz ardı edilebilecek kadar küçük olduğundan ACK paketlerinden kayıp vermeyeceğimizi kabul edelim. Hızlı tekrar iletim eşiğini K ve bağlantı esnasında ilan edilmiş maksimum pencere boyutunu W_{max} olarak tanımlayalım. l_i de penceredeki i.nci kayıp paketi göstersin. Eğer (m-1) paketin bütün normal ACK'leri alındıysa ve l_1 , k.ncı turda iletilecek m.nci paket ise pencere kaybı Ω. Kayıp paket içeren bir pencerenin ilk paketi daima ilk kayıp paketidir. Kayıplı bir pencerede n kayıp paket için kaybedilmeyen veya k.ncı kurtarma periyoundaki turda yeni iletilmiş kayıp paket sayısını Φ_k olarak ifade edelim. Eğer Ω, u paket sayısına eşitse, Φ_1 her zaman ($u - n$)'e eşittir. $h \geq 2$ ise Φ_h kayma ve ($h - 1$).nci paket kaybının tekrar iletimden sonra kullanılabilir penceredeki kayma ve şişme ile iletilmiş paket sayısını ifade eder [24].

8.2.2 Φ_n'nin Türetilmesi

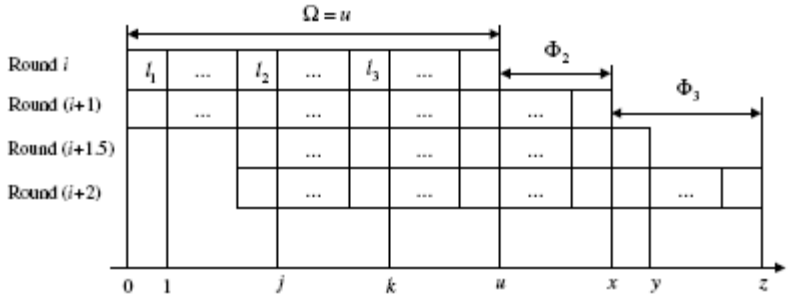
$\Omega = u$ için Φ_1 u paketten düzgün iletilmiş paket sayısı olsun. Bu durumda,

$$\phi_1 = u - 1 \qquad (8.3)$$

l_1 için tekrar hızlı iletim işleminin yeniden yapılmasından sonra Φ_2 artması ile kullanılabilir pencerenin yeni paketleridir. Son ulaşan çift ACK'yı ele alacak olursak l_1 pencereyi *[u/2] + (u-2)* ve *u* paket hala yarım kalmışdır; Φ_2 şöyle olur [24].

$$\phi_2 = [u/2] + (u-2) - u = [u/2] - 2 \qquad (7.4)$$

Eğer $\Phi 2 \geq K$ ise, l_2 tekrar hızlı iletim ile kurtarılabilir. Şekil 8.1 TCP Reno'nun kayıpların kurtarılması ile ilgili özelliklerini bir penceredeki üç paket kaybı için göstermektedir.Herbir tur aşağıdaki gibi açıklanabilir.



Şekil 8.1. *TCP Reno'nun üç kayıp paket için kayıpların kurtarılması davranışı [24].*

- Kayıpların kurtarılması *i* turunda başlar.
- l_1 için son çift ACK (*i*+1).nci turda alınır.
- l_2 için ilk ACK, (*i*+15).nci turda l_1'in tekrar iletimi ile alınır.
- l_2 için son çift ACK (*i*+2).nci turda alınır.

Eğer ilk kayıp pencereyi $\Omega(i)$ ve *a*'nın sağ sınır değerini de $R(a)$ şeklinde ifade edersek, herbir sınır değeri $L(\Omega(i)) = 0$, $R(\Omega(i)) = u$, $R(l2) = j$, $R(l3) = k$, $R(\Omega(i + 1)) = x$, $R(\Omega(i + 1.5)) = y$ ve $R(\Omega(i + 2)) = z$ olur [24].

Bütün j,k değerleri için $2 \le j \le u-1,\ j+1 \le k \le u,\ x, y$ ve z aşağıdaki gibi ifade edilebilir.

$$\begin{aligned} x &= [u/2] + (u-3) \\ y &= [j-1] + (u/2) \\ z &= [j-1] + (u/4) + \phi_2 \end{aligned} \qquad (8.5)$$

Eğer (i+1.5).nci turda $y \ge x$ olursa $y - x$ adet paket iletilmiştir. Çünkü l_2 daha tekrar iletilmemiştir ve l_2 için çift ACK üretilebilir. Bu iki durum için, Φ_3 şeklinde verilebilir [24].

$$\begin{aligned} \phi_3 &= z - \max(x, y) \\ &= \begin{cases} (j-1) + [u/4] - u & 2 \le j \le u-2 \\ [u/4] - 3 & j = u-1. \end{cases} \end{aligned} \qquad (8.6)$$

Φ_3, Φ_1 ve Φ_2'den farklı olarak l_2'nin pozisyonu ile olduğu kadar Ω'nin boyutlarıyla da bağlantılıdır. j maksimum olduğunda, eşitlik (8.5)'e göre z de maksimum değeri alır. Bu şu nedenledir, pencere en çok $j = u - 1$ olduğunda kayar. Bu yüzden, Ω'nun minimum değeri, RTO'suz üçlü paket kayıplarının kurtarılmasında $[u/4] - 3 = K$ ile elde edilebilir. $2 \le j \le u - 2$ ise l_2 için kurtarılma koşulu şöyle olur [24]

$$(j-1) + [u/4] - u \ge K \qquad (8.7)$$

Eğer l_1 ve l_2 arasındaki paket sayısı $u-[u/4]+(K-1)$ 'e eşit veya daha büyükse l_3 tekrar hızlı transfer işlemi ile kurtarılabilir.

Bir pencere için dörtlü paket kayıpları, tekrar hızlı transfer yöntemi ile hiçbir şekilde kurtarılamaz [24].

8.3 Olasılık Analizleri

8.3.1 Kayıp Paket Modelleri

Rastgele paket kayıpları için, herbir paket p olasılığınca kaybedilir ve bu kayıplar bağımsızdır [66,68]. İlişkili paket kayıpları ilk durum Markov Zinciri ile modellenir [9,15]. Markov zincirinin bağlantı olasılık matrisi Q_c;

$$Q_C = \begin{pmatrix} p_{BB} & p_{BG} \\ p_{GB} & p_{GG} \end{pmatrix} \qquad (8.8)$$

şeklindedir. Kanal iyi ve kötü durum olmak üzere iki durumda olabilir. Durum kötü iken bir paketin kaybolma olasılığı 1'dir. Eğer α'ya p_{GB} ve $\beta' ye$ p_{BG} dersek, iyi veya kötü durumda olma olasılığı (π_G, π_B) aşağıdaki gibidir [24];

$$\Pi_G = \frac{\beta}{\alpha+\beta}, \quad \Pi_B = \frac{\alpha}{\alpha+\beta}. \qquad (8.9)$$

Hatta ortalama iyi durum süreci $1/\alpha$ ise ortalama kötü durum süreci $1/\beta$ dir. Sürecin miktarı paketlerin sayısı kadardır. Çünkü her iletim zamanı için durum değişmekte olduğu kabul ediyoruz [24].

8.3.2 Markov İşlemi

Bir penceredeki TCP değişimi Markov Zincirine adapte edilerek analiz edilebilir. Markov zincirinin durağan dağılımı, değişim olasılıkları belirlendiği zaman sayısal olarak elde edilebilir. Tıkanık pencere boyutu kayıp kurtarılmasından sonra her zaman *[u / 2]*'e kadar düşmesi hariç, değişim olasılıklarını hesaplamak için [66,68] 'teki işlemleri takip

edeceğiz. Gelecek göngüdeki kaybolan paket sayısı ve pencere boyutları arasındaki ilişki eşitlik (8.2) ile tanımlanabilir [24].

Her RTO oluşumunda gelecek döngünün W_{th}'ı aşağıdaki gibi ifade edilebilir. $\Omega = u$ için, paketlerin arasından iki paketin kaybolacağını ve RTO oluşacağını varsayalım. Eğer $u \geq K + 2$ ise, en azından ilk kayıp paket tekrar hızlı transfer ile belki kurtarılabilir. Bu durumda ikinci kayıp paketin RTO'ya neden olacağından emin olabiliriz. Bu yüzden gelecek döngüde W_{th}'nin değeri $[u / 2]$ olur. Ω'nin değerine ve kayıp paket sayısına bağlı olarak, gelecek döngüde W_{th}'nin değeri $[u / 2]$, $[u / 4]$ ve $[u / 8]$ değerlerinden biri olur [24].

İlişkili kayıp paket modeline göre, hiç paket gönderilememiş bir kanal için RTO süresince kanalın durumunu bilmek imkansızdır. Eğer gelecek döngüdeki ilk paket kaybolmaz, ancak kanalın durumunun iyi olduğu da bir açıktır. Kanal gelecek döngüye her zaman iyi durumda başlamış olmasına rağmen arka arkaya ne kadar RTO gerçekleşeceğini bilemeyiz. Bu yüzden $\{2,3,4.....,W_{max}\}$ alanı üzerinde $\{\Omega_i\}$ işlemini hesaba katmalıyız. Birbiri ardına gerçekleşen RTO'ların tıkanık pencere işlemi değişimini etkilemediğini de vurgulamalıyız [24].

8.3.3 Tekrar Hızlı İletme Olasılığı

Bir penceredeki her paketin tekrar hızlı iletimle kurtarılabilirliği olan R_R TCP Reno'nun tekrar hızlı iletim olasılığı olsun. Öyleyse, elimizde

$$R_R = \sum_{n} \sum_{u=1}^{W_{\max}} R_R^{(n)}(u) \pi_n(u) \qquad (8.10)$$

var. Buradaki $\pi_n(w)$, n ve $R_R^{(n)}(u)$ için kayıp pencerenin durağan durum olasılığı olsun. $R_R^{(n)}(u)$ aşağıdaki gibi [24]

$R_R^{(n)}(u)$ = *P{(u-1) paket dışında (n-1) paket kaybedilir}*

**P {kayıp düzeltme sırasında paket kaybı yoktur}.* *(8.11)*

olur. Rasgele paket kayıpları için $R_R^{(n)}(u)$ n cinsinden şöyle yazılabilir [24],

$$R_R^{(1)}(u) = (1-p)^u$$
$$R_R^{(2)}(u) = (u-1)p(1-p)^{u+\phi_2}$$
$$R_R^{(3)}(u) = \binom{\lfloor u/4 \rfloor - K}{2} p^2 (1-p)^{u+\phi_2+\phi_3}$$

Sonuçta rasgele paket kayıpları için toplam tekrar hızlı iletim olasılığı [24],

$$R_R(u) = (1-p)^u \{1 + (u-1)p(1-p)^{\phi_2} + \binom{\lfloor u/4 \rfloor}{2} p^2 (1-p)^{\phi_2+\phi_3}\}. \quad (8.13)$$

gibidir.

İlişkili paket kayıpları modelinde, $R_R^{(n)}(u)$'yi hesaplayabilmek için kayıp paketlerin düzenleri incelenmelidir. *n=1* için, bir paketin kaybedilmesinden sonra kanal iyi duruma geçmelidir ve tekrar iletimler dahil *u* kadar paketin iletimi tamamlanana kadar da iyi olarak kalmalıdır. *n=2* için $R_R^{(2)}(u)$ iki kayıp paketin ardarda olma olasılığı olsun ve $R_R^{(2)}(u)_{ns}$ iki kayıp paketin ardarda olmama olasılığı olsun. O zaman $R_R^{(2)}(u)$ değeri $R_R^{(2)}(u)_{su}$ ve $R_R^{(2)}(u)_{ns}$'nin toplamına eşittir. *n=3* için l_2 ve l_3 ardarda olmayabilir veya verilen şartlar ve *j=u-1* için ger zaman ardarda iken $j_{\min} \le j \le u-2$ olmayabilir. Sonuç olarak ilişkili paket kayıpları için $R_R^{(n)}(u)$

$R_R^{(1)}(u) = \beta(1-\alpha)^{\phi_1}$

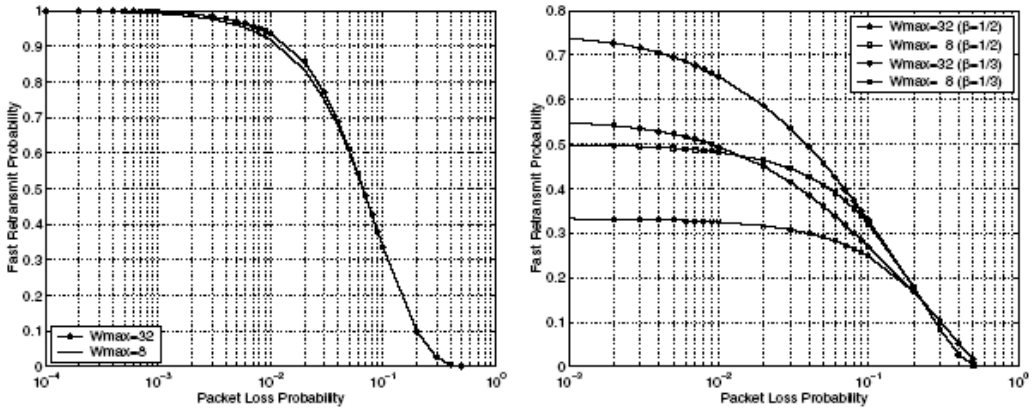
$$R_R^{(2)}(u) = \beta(u-1)^{u+\phi_2-3}\{(1-\alpha)(1-\beta)+(u-2)\alpha\beta\}$$

$$R_R^{(3)}(u) = \begin{cases} \alpha\beta^2(1-\beta)(1-\alpha)^{\phi_2+\phi_3-2} & j = u-1 \\ \alpha\beta^2(1-\alpha)^{\phi_2+\phi_3-3}\{N_{su}(1-\alpha)+N_{ns}(1-\beta)\} & j_{\min} \le j \le u-2 \end{cases} \qquad (8.14)$$

Burada l_2 ve l_3 ardarda ise durum sayisi N_{su} ve l_2 ve l_3 ardarda değilse durum sayisi N_{ns}. N_{su} ve N_{ns}'in değerleri aşağıdaki gibidir [24]

$$N_{su} = u - (j_{min} - 1) - 2$$

$$N_{ns} = \begin{pmatrix} u - (j_{min} - 1) \\ 2 \end{pmatrix} - (N_{su} - 1). \qquad (8.15)$$



Şekil 8.2. *(a) rasgele paketlerin için* *(b) ilişkili paketler için*

SONUÇLAR

Paket anahtarlamalı ağlar, farklı kaynaklardan verinin aynı yol boyunca iletilmesine izin verir. Bu yol üzerinde paketlerin göndericiden alıcıya iletilebilmesi için kullanılan yönlendiriciler, paketlerin alınma oranı işlem yapılarak gönderilme oranından büyük olabileceğinden gelen paketleri tampon adı verilen bir sırada saklar. Bu sıralar ilk giren ilk çıkar prensibine göre çalışır ve sınırlı bir kapasiteleri vardır. Sınırlı kapasitesi olan sıralar dolduğu zaman sıkışıklık oluşmakta ve gelen paketler düşürülmeye başlanmaktadır. Bu araştırma içerisinde sıkışıklığın, oluşmadan önce tespit edilerek önlenmesi için geliştirilen RED, REM ve RENO modelleri incelenmiştir. Sıkışıklık seviyesi, RENO'da, tampon taşmaları ile AQM olmadan ölçülürken, RED ortalama sıra uzunluğu parametresi ile REM'de ise ücret parametresi ile ölçülmektedir

Aktif sıra yönetiminin esas amacı genel olarak, düşük ortalama sıra gecikmesi ve yüksek işlem hacminin sağlanmasıdır. Burada bunların ayrıntıları ve yapılan benzetmelere bakılarak, çeşitli yorumlar yapılmıştır. RED'in ana amaçlarından bir tanesi, sıra uzunluğu algoritması ve erken sıkışma bildirimi kombinasyonunu kullanarak, düşük ortalama sıra gecikmesi ve yüksek işlem hacmini birarada başarmakdır. RED'in benzetme denemeleri ve işlemsel deneyler bu konuda oldukça başarılı olduğunu ortaya koymuştur. Bunun yanında RED'in en zayıf noktası ise, sıkışma seviyesinde ve parametre ayarlarında ortalama sıra uzunluğunun çeşitli olmasıdır.

REM'in esas amaçlarından birisi sıkışıklık ölçüsünü (ücret) performans ölçüsünden (kayıp ve sıra) ayırmaktır, böylece sıkışma ölçüsü, kaynakların sayısı ile çeşitlenmek zorundadır, performans ölçüsü hedef etrafında bağımsız olarak kararlı hale getirilebilir. Benzetme sonuçlarından, temel RED'in basitlik ve kararlılığından fedakarlık etmeden bu amacı başarabildiğimiz görünmektedir. Bu özellik kablolu ağlar üzerinde TCP'nin performansını geliştirmek için kullanılabilir. Bunun yanında bunun bir denge özelliği olduğunu ve REM'in iletim davranışının daha dikkatli çalışılması gerektiği

vurgulanmaktadır. Diğer bir amacı ise hem giriş oranını hat kapasitesi etrafında ve hemde sırayı küçük bir hedef etrafında, hattı paylaşan kaynakların sayısını önemsemeden kararlı hale getirmektir.

AQM'in kararlılığını anlatmak için çok hatlı çok kaynaklı model geliştirilmiştir. Karışık kaynaklar ve RED'in kararlı bölgesinin gösterilen formu ile tek hatlı durumu için uygun kararlılık şartı sunulmuştur. Ağın gecikme yada kapasitesindeki büyüme sonucunda RED kararsız olmaktadır. Analizler de, TCP kararlılığında RED'in zorluğunun rolünü belirtilmiştir. Ayrıca, kontrol teorisine göre TCP ve AQM modelinin bir birleşimini analiz edilmiştir. Daha önceden geliştirilmiş sistemin doğrusal olmayan modeli doğrusallaştırılarak kullanılmış ve AQM sisteminde RED gerçekleştirilerek bu analiz gösterilmiştir. RED'in doğrusal geri besleme kontrol sisteminin, kararlılık operasyonuna yol göstermesi açısından, parametrelerin seçimi ile ilgili tasarım rehberi sunulmuştur. Ayrıca sistemin kararlılığına ilişkin ifadeler türetilerek tasarlanmıştır.

KAYNAKLAR

[1]. *Jacobson V., Karels M.J.,"Congestion Avoidance and Control", Kasım 1988.*

[2]. *Fabiani M., "TCP Congestion Avoidance", Internetworking 2G1305, 2003.*

[3]. *Weigle M.A.C.,"Investigating The Use Of Syncronized Clocks in TCP Congestion Control ", Doktora Tezi, 2003.*

[4]. *Floyd S., Jacobson V., "Random Early Detection Gateways for Congestion Avoidance" , IEEE/ACM, Ağustos 2003.*

[5]. *Floyd S.,Gummadi R.,Shenker S, "Adaptive RED : An Algorithm for Increasing the Robustness of RED's Active Queue Management", Ağustos 2001.*

[6]. *Wydrowski B., Zukerman M., "GREEN: An Active Queue Management Algorithm for a Self Managed Internet", ICC 2002, New York, Vol. 4, 2368-2372*

[7]. *Braden B., Clark D., Crowcroft J., Davie B., Deering S., Estrin D., Floyd S., Jacobson, V., Minshall G., Partridge C., Peterson L., Ramakrishnan K. K., Shenker S., and Wroclawski J., "Recommendations on queue management and congestion avoidance in the internet", Internet Draft, 1998*

[8]. *Özekes S., "Evaluation Of Active Queue Management Algorithms", İstanbul Ticaret Üniversitesi Fen Bilimleri Dergisi Yıl : 4 Sayı :7 Bahar 2005/1 123-140*

[9]. *Michele Zorzi and A. Chockalingam: Throughput Analysis of TCP on Channels with Memory. IEEE Journals on Selected Areas in Communications, Vol: 18, 2000, 1289–1300*

[10]. *M. Christiansen, K. Jeffay, D. Ott, and F.D. Smith, "Tuning RED for web traffic", ACM/SIGCOMM, 2000.*

[11]. *Alhussein A. Abouzeid, S. Roy, and M. Azizoglu: "Stochastic Modeling of TCP over Lossy Links", IEEE INFOCOM, 2000, 1724–1733*

[12]. *Teunis J. Ott, T. V. Lakshman, and L. H.Wong, "SRED: Stabilized RED" IEEE/INFOCOM, 1999.*

[13]. *W. Feng, D. Kandlur, D. Saha, and K. Shin, "Blue: A New Class of Active Queue Management Algorithms," Tech. Rep., UM CSE-TR-387-99, 1999.*

[14]. *Dong Lin and Robert Morris, "Dynamics of random early detection," ACM/SIGCOMM, 1997.*

[15]. *Farooq Anjum and Leandros Tassiulas: "On the Behavior of Different TCP Algorithms over a Wireless Channel with Correlated Packet Losses", ACM SIGMETRICS, 1999, 155–165.*

[16]. *Feng W., Shin K. G., Kandlur D. D., Saha D., "The BLUE active queue management algorithms", IEEE/ACM , Vol.10, No:4, 2002, 513-528.*

[17]. *S.Athuraliya, V.H.Li, S.H.Low, Q.Yin, "REM, Active Queue Management", January 2001.*

[18]. *Feng W., Kapadia A., Thulasidasan S., (2002/a), "GREEN: Proactive Queue Management over a Best-Effort Network", IEEE Global Telekomünikasyon Konferansı (GLOBECOM 2002), Taipei, Taiwan, Vol.21, No:1, 1784-1788*

[19]. *Pletka R., Waldvogel M., Mannal S., "PURPLE: Predictive Active Queue Management Utilizing Congestion Information", Proceedings of the 28. Yıllık IEEE Konferansı, Yerel Bilgisayar Ağları, LCN 2003, 21-30*

[20]. *K. Ramakrishnan, S. Floyd, "A Proposal to add Explicit Congestion Notification (ECN) to IP", RFC 2481, Ocak 1999.*

[21]. *S.H.Low, F.Paganini, J. Wang, S.Adlakha, J.C.Doyle, "Dynamics of TCP/RED and Scalable Control", IEEE Infocom, Haziran 2002.*

[22]. *Martin May, Thomas Bonald, and Jean-Chrysostome Bolot, "Analytic evaluation of RED performance," IEEE Infocom, Mart 2000.*

[23]. *P.P. Mishira, D. Sanghi, and S. K. Tripathi: "TCP Flow Control in Lossy Networks: Analysis and Enhancements", IFIP Transactions C-13, S.V. Raghavan, G.V. Bochman, and G. Pujolle, Eds. Amsterdam, The Netherlands: Elsevier North-Holland, 1993, 181–193.*

[24]. *B. J. Kim and J.Y. Lee: "Analytic Models of Loss Recovery of TCP Reno with Packet Losses", ICOIN, 2003.*

[25]. *Chris Hollot, Vishal Misra, Don Towsley, and Wei-Bo Gong, "A control theoretic analysis of RED," IEEE Infocom, Nisan 2001.*

[26]. *Steven H. Low, "A duality model of TCP flow controls", IP Trafiğini Ölçme, Modelleme ve Yönetme için ITC Semineri, Eylül 2000.*

[27]. *Chris Hollot, Vishal Misra, Don Towsley, and Wei-Bo Gong, "On designing improved controllers for AQM routers supporting TCP flows," IEEE Infocom, Nisan 2001.*

[28]. *Fernando Paganini, John C. Doyle, and Steven H. Low, "Scalable laws for stable network congestion control", Karar ve Kontrol Semineri,*

Aralık 2001.

[29]. *Mark Allman, "A web server's view of the transport layer," ACM Computer Communication Review, vol. 30, no. 5, Ekim 2000.*

[30]. *Steven H. Low, Larry Peterson, and Limin Wang, "Understanding Vegas: a duality model," ACM, 2002.*

[31]. *Lawrence S. Brakmo and Larry L. Peterson, "TCP Vegas: end to end congestion avoidance on a global Internet," IEEE, vol. 13, no. 8, pp. 1465–80, Ekim 1995.*

[32]. *Barrett O'Neill, Elementary Differential Geometry, Academic Press,1966.*

[33]. *Glenn Vinnicombe, "On the stability of end-to-end congestion control for the Internet," Teknik Rapor., Cambridge Univiversity, Aralık 2000.*

[34]. *R. Johari and D Tan, "End-to-end congestion control for the internet: Delays and stability", Teknik Rapor, 2000, Cambridge Univ. İstatistiksel Labaratuar Araştırma raporu 2000-2.*

[35]. *L. Massoulie, "Stability of distributed congestion control with heterogeneous feedback delays", Teknik Rapor, Microsoft Araştırması, Cambridge UK, TR 2000-111, 2000.*

[36]. *W. Feng, D. Kandlur, D. Saha, and K. Shin, "A self–configuring RED gateway," IEEE INFOCOM, Mart 1999.*

[37]. *R. J. Gibbens and F. P. Kelly, "Resource pricing and the evolution of congestion control," Automatica, vol. 35, 1999.*

[38]. *Srisankar Kunniyur and R. Srikant, "End–to–end congestion control schemes: utility functions, random losses and ECN marks," IEEE Infocom, Mart 2000.*

[39]. *Srisankar Kunniyur and R. Srikant, "A time–scale decomposition approach to adaptive ECN marking," IEEE Infocom, Nisan 2001.*

[40]. *Sanjeewa Athuraliya, Victor H. Li, Steven H. Low, and Qinghe Yin, "REM: active queue management," IEEE Network, Mayıs/Haziran 2001, ITC17'nin genişletilmiş versiyonu, Salvador, Brezilya, Eylül 2001.*

[41]. *"Parallel simulation environment for complex systems," http://pcl.cs.ucla.edu/projects/parsec/.*

[42]. *Matthew Mathis, Jamshid Mahdivi, Sally Floyd, and Allyn Romanow," TCP selective acknowledgement options", RFC 2018, Ekim 1996.*

[43]. *Kevin Fall and Sally Floyd. Simulation-based comparisons of Tahoe, Reno, and SACK TCP. ACM Computer Communication Review, Temmuz 1996.*

[44]. *Mark Allman, Vern Paxson, and W. Richard Stevens. "TCP congestion control", RFC 2581, Nisan 1999.*

[45]. *Janey Hoe. "Improving the start-up behavior of a congestion control scheme for TCP", ACM SIGCOMM, 1996.*

[46]. *Janey Hoe." Startup dynamics of TCP's congestion control and avoidance Schemes", Master's thesis, MIT, http://ana-www.lcs.mit.edu/anaweb/ps-papers/hoe-thesis.ps, 1995.*

[47]. *Sally Floyd and Tom Henderson. "The NewReno modification to TCP's fast recovery algorithm", RFC 2582, Experimental, Nisan 1999.*

[48]. *Jitendra Padhye and Sally Floyd, "On inferring TCP behavior", ACM SIGCOMM, 287–298, Eylül 2001.*

[49]. *Victor Firoiu and Marty Borden," A Study of Active Queue Management for Congestion Control", IEEE Infocom, 2000, 1435–1444.*

[50]. *V. Jacobson, K. Nichols, and K. Poduri, "RED in a Different Light", Eylül 1999.*

[51]. *T. Ziegler, S. Fdida, and C. Brandauer "Stability Criteria for RED with Bulk-data TCP Traffic", 2001. Teknik Rapor, Ağustos 1999.*

[52]. *S. Floyd, M. Handley, J. Padhye, and J. Widmer. TFRC Web Page, 2000.*

[53]. *W. Stevens,"TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms", RFC2001, 1997.*

[54]. *Vishal Misra, Wei-Bo Gong, and Don Towsley, "Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED," ACM/SIGCOMM, 2000.*

[55]. *"ns-2 Network Simulator", http://www.isi.edu/nsnam/ns/.*

[56]. *S. Mascolo, "Congestion control in high-speed communication networks," Automatica, vol. 35, Mart 1999, 1921–1935.*

[57]. *Gene F. Franklin, J. David Powell, and Abbas Emami-Naeini, "Feedback Control of Dynamic Systems", Addison-Wesley, 1995.*

[58]. *F. Kelly, "Mathematical modelling of the Internet," , 2001.*

[59]. *Karl J. A., "Oscillations in Systems with Relay Feedback", IMA Volume Matematik and Uygulamaları, 1995, vol. 74, 1–25.*

[60]. *Sally Floyd, "Recommendation on using the "gentle " variant of RED," http://www.aciri.org/floyd/red/gentle.html, Mart 2000.*

[61]. *J. Mahdavi and Sally Floyd, "TCP-friendly unicast rate-based flow control", Ocak 1997.*

[62]. *M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The macroscopic behavior of the TCP congestion avoidance algorithm," Computer Communication Review, vol. 27, no. 3, Temmuz 1997.*

[63]. *W. Stevens: TCP/IP Illustrated, Vol. 1 The Protocols. Addison-Wesley, 1997*

[64]. *Sanjeewa Athuraliya and Steven H. Low, "Optimization flow control, II: Implementation", http://netlab.caltech.edu, Mayıs 2000.*

[65]. *H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links", IEEE/ACM, Vol. 5, 1997, 756–769*

[66]. *T.V. Lakshman and Upamanyu Madhow, "The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss", IEEE/ACM, Vol. 5, 1997, 336–350*

[67]. *Anurag Kumar, "Comparative Performance Analysis of Versions of TCP in a Local Network with a Lossy Link" IEEE/ACM, Vol. 6, 1998, 485–498*

[68]. *Anurag Kumar and Jack Holtzman, "Comparative Performance Analysis of Versions of TCP in a Local Network with a Mobile Radio Link", http://ece.iisc.ernet.in/ anurag/*

[69]. *J. Padhye,V. Firoiu, D.F. Towsley, and J.F.Kurose," Modeling TCP Reno Performance: A Simple Model and Its Empirical Validation. IEEE/ACM , Vol. 8, 2000, 133–145*