BİLGİSAYAR AĞLARINDA SIKIŞIKLIK VEGAS VE DUALİTY PROTOKOLLERİ MODELLERİ

(ÖDEV)

Prof. Dr. Remzi YILDIRIM

2016- AYBÜ-ANKARA

BİLGİSAYAR AĞLARINDA SIKIŞIKLIK VEGAS VE DUALİTY PROTOKOLLERİ MODELLERİ

Danışman:

Yrd. Doç. Dr. Remzi YILDIRIM

HAZİRAN 2006

ANKARA

BİLGİSAYAR AĞLARINDA SIKIŞIKLIK KONTROLÜ VE VEGAS VE DUALİTY MODELLERİ

(Yüksek Lisans Dönem Projesi)

Mehmet MİNTAŞ

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Haziran 2006

ÖZET

Gelişen internet ağı beraberinde alıcı-verici arasındaki arz ve talebi karşılama sıkıntısı doğurmuştur. Bu iletişim kapasitesinin artışı iletilen verilerin kayıpsız ve sorunsuz bir şekilde taşınması için daha kararlı ağ yönetimleri arayışı getirmiştir. Sıkışıklık bir ağın mevcut en önemli problemidir. Bu sorunu aşmak için birçok model üzerine çalışmalar yapılmış ve halen daha kararlı bir ağ elde etmek için devam etmektedir. Geliştirilen sıkışıklık yönetim algoritmaları sıkışıklığa daha farklı bakış açıları ile yaklaşmış ve sıkışıklık ölçütü olarak farklı parametreler kullanmışlardır. Bu çalışmada mevcut VEGAS ve Duality modelleri üzerine bir kaynak araştırması yapılarak sağladığı avantaj, yöntem ve performanslarını incelenmektedir.

Bilim Kodu :
Anahtar Kelimeler : **Sıkışıklık Kontrolü, VEGAS, DUALİTY**
Sayfa Adedi :
Proje Yöneticisi : **Yrd. Doç. Remzi YILDIRIM**

BÖLÜM 1

GİRİŞ

TCP VEGAS 1994 yılında TCP RENO' ya bir alternatif olarak ortaya çıktı. Kapasite kullanım ölçüsü olarak paket kaybını kullanan RENO dan farklı olarak VEGAS sıkışıklık gecikmesini kullanmaktadır. TCP VEGAS TCP için ilk kez Brakmo tarafından sunulan bir tasarımıdır. TCP VEGAS geliştirilmiş bir yeniden transmisyon stratejisini içerir. Bu strateji iyi parçalanmış tur ölçümlerine ve yavaş-başlangıç ve tıkanıklıktan kaçınma esnasında tıkanıklık tespiti için oluşturulan yeni mekanizmalara dayanır. Yaratıcı teknikler ve etkileyici performans kazanımları son yıllarda birçok tartışmanın konusu olmuştur. Bu çalışma TCP VEGAS' ın tasarımına taze bir bakış sunmakta ve TCP VEGAS' ın yeniliklerinin avantajlarına ışık tutmaya çalışmaktadır.

TCP RENO nun sıkışıklık tespiti ve kontrol mekanizmaları parçaların kaybını bir sinyal olarak kullanmaktadır. Bu parça kayıpları şebekede tıkanıklık olduğunu göstermektedir. Bu yüzden TCP RENO nun kayıplar olmadan önce tıkanıklığın başlangıç aşamalarını tespit edecek bir mekanizması yoktur. Dolayısıyla TCP RENO kayıpları engelleyemez. Dahası, TCP RENO reaktiftir, yani bağlantının mevcut band genişliğini bulmak için kayıplar üretmeye ihtiyacı vardır. Öbür taraftan, TCP VEGAS ın tıkanıklık tespit mekanizması aktiftir, yani çıktı oranındaki değişiklikleri gözlemleyerek tıkanıklıktaki başlangıcı tespit etmeye çalışır. TCP VEGAS bu çıktı ölçümlerinden tıkanıklık penceresi ayarlama politikasını çıkarır, bu da bağlantı kayıplar vermeden önce gönderme oranını azaltabilmeyi sağlar.

TCP VEGAS çeşitli değişik tekniklerin bir birleşimidir. Her bir teknik kendi başına bir tartışma konusudur. Daha önce yapılan tartışma ve çalışmalar ya yalnız belli bir mekanizma üzerinde yoğunlaşmış ya da TCP VEGAS ın bütün olarak davranışını değerlendirmeye çalışmıştır. Ancak asıl soru TCP VEGAS içerisindeki hangi tekniğin performans kazanımlarından sorumlu olduğudur. Bu soru şu ana kadar cevapsız kalmıştır.

TCP paketleri adım adım iletmek için pencere tabanlı akış kontrol sistemi kullanır. Her kaynak gönderilebilecek maksimumum paket sayısını içeren: iletilmiş ama bilgilendirme yapılmamış pencere boyutu değişkenini içerir. Bir veri gönderilirken kaynak yeni bir paketi göndermek için bilgilendirmeyi beklemek zorundadır. Genel stratejinin 2 özelliği önemlidir. Birincisi bilgilendirmeler geç kaldığı ve ağ karıştığı zaman algoritma kendi kendine zamanlamalıdır ki TCP kaynağı otomatikman yavaşlasın. İkinci olarak pencere boyutu değeri kaynağın oranını belirler: her bir döngü zamanında pencere paketi gönderilir. Bu ikinci özellik Jacobenin 1988 de yazdığı kitapta açıklanmıştır. Eklenebilir artırılabilir çoklu aktif azaltılabilir algoritmayı iddia etmiş. Ağ sıkışıklığına göre pencere genişliğini ayarlayabileceğini iddia etmiş. Bu algoritma TCP RENO' da uygulanmış ve bu algoritmayı içeren TCP'nin bir değişkenidir. TCP RENO 3 ana mekanizmadan oluşur: Yavaş başlama, sıkışıklığı giderme ve hızlı düzeltme. Kaynak küçük pencere boyutları ile ihtiyatlı olarak başlar ve her bilgilendirmeyi aldığı zaman paket boyutu büyültür. Her döngü sonunda pencere boyutunu 2 ile çarpar. Pencere boyutu eşik değerine ulaştığı zaman kaynak sıkışıklık kaçınma

fazı girer her bilgilendirme aldığı zaman karşılıklı anlık pencere büyüklüğünü daha yavaş olarak artırır. Her bir dönüş zamanında bu pencereyi bir paket büyütür. Her çift bilgilendirmedeki kayıpları ölçerken kayıp olan paketleri tekrar gönderir. Bunların pencerelerini yarıya böler ve sıkışıklık kaçınmalarını da tekrar girer. Bu hızlı tekrar gönderme ve hızlı düzeltme olarak adlandırılır. Zaman aşımı sürecinde kayıpları kaynaktan belirleyebilmek için sıkışıklık giderme yerine yavaş başlamaya geçilmelidir. 1994 de TCP RENO'nun alternatif olarak TCP VEGAS tanıtılmıştır (Brakmo ve Peterson 1995). TCP RENO'nun 3 mekanizmasını geliştirir. Bunlardan ilki yavaş başlama ve daha az kayıpları meydana getirme sırasında pencere büyüklüğünü daha tutarlı büyütmesidir. İkincisi tekrar gönderme mekanizmasını çiftli bilgilendirmenin alınmasının kontrol edildiğinde RENO da olduğu gibi üçüncü çiftli bilgilendirmeyi beklemeden yeniden gönderme mekanizmasıdır. Bu yöntem ve kayıpları zamanında yönlendirmektedir. Üçüncüsü RENO' nun davranışlarını düzelten yeni bir sıkışıklık giderme mekanizması. VEGAS'ın RENO algoritmasından farkı ağ kapasitesinin ne kadar olduğunu öğrenmek için sıkışıklığa teşvik eder, VEGAS kaynağı gerçekleşen ve beklediği sıkışıklık saldırısı arasındaki farkı göstererek bekler. VEGAS yol boyunca yönlendiricide daha az sayıda paket tamponlanmasını sağlamak için kaynağın gönderim boyutunu artırma stratejisidir. Bu yazıda VEGAS' ın sıkışıklık giderme mekanizmasını anlatacağız. İyi bilinir ki dosya büyüklükleri internet üzerinden büyük bir yük getirir. Basit olarak TCP bağlantıları kısa olduğu zaman birçok paket büyük TCP bağlantılarından oluşur. Bu büyük paketler küçük değildir. TCP tarafından etkin olarak kontrol edilmelidir. İleride daha açık anlatılacağı gibi sıkışıklık giderme bant genişliği ayırmak olarak belirlenir ve bu paketler tarafından servisin kalitesi tecrübelendirilir

Bu çalışma şu bölümlerden oluşmaktadır. Bölüm 2 sıkışıklık giderme ve kontrol hakkında temel bilgilendirme sunmaktadır. Bölüm 3 de VEGASIN RENO ya alternatif olan farklılıkları ve performansı değerlendirilmiştir. Bölüm 4 de TCP sorgu yönetim algoritmasının DUALİTY modeli ve 5. bölümde VEGAS' ın DUALİTY modeli ele alınmış, bölüm 6 da TCP VEGAS kararlılık modelleri ayrıntısıyla verilmiştir 7. bölümde TCP VEGAS ile modellemeye deyinilmiştir.

BÖLÜM 2

SIKIŞIKLIK GİDERME VE KONTROL

Bilgisayar ağları son birkaç yıldır çok ilerlemiş ve tecrübe kazanmıştır. Bu gelişme ile sıkışıklık problemi meydana gelmiştir. Lokal tamponlama aşımlarından dolayı paketlerin %10 internet ağ geçitlerinde kayıp olduğu görülmüştür. Bu problemleri incelediğimiz sorunun protokolün kendisinden değil zaman transfer protokol uygulamasından kaynaklandığı görülüyor: Pencere tabanlı iletim protokolünü gerçekleştirmek için açık olan yollar ağ sıkışıklığının cevabını yanlış gösterir. Yanlış eylemlere örnekler vereceğiz ve doğru şeylerin oluşması sağlayan basit algoritmaların bazılarını açıklayacağız. Bu algoritmalar paket haberleşme prensiplerine dayanarak transfer bağlantılarını güçlendirmekle ağ kararlılığını gerçekleştirmeye dayanır. Bu prensipten Algoritmaların nasıl çıkarılacağını göstereceğiz ve sıkıştırılmış bir ağda trafiği nasıl etkilediğini göstereceğiz.

1986'ın Ekiminde internet birkaç seri sıkışıklık çökmesi getirmiştir. Bu bant genişliğindeki ani düşmeden çok şaşırdık ve bu problemin neden kötüye gittiğini düşünmeye başladık. Özellikle, 4,3 BSD (BERKELEY UNİX) TCP yanlış hareket etimi ya da bitmeyen ağ koşullarında daha iyi çalışmaya başlayacağını merak ettik.

2.1. Yavaş Başlama Dengesini Sağlamak

Bu zamandan sonra 4 BSD TCP ye 7 yeni algoritma daha eklendi

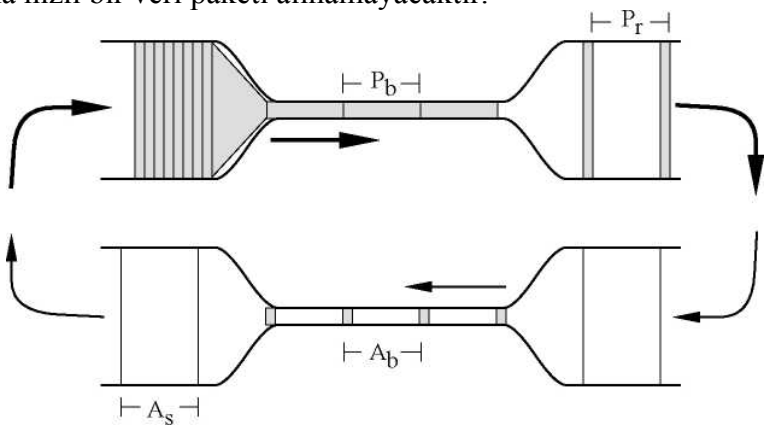
1- Çevrim zamanı değişim tahmini
2- Logaritmik yeniden gönderim zamanlayıcı bitimi
3- Yavaş başlama
4- Daha girişken alıcı bilgilendirme politikası
5- Sıkışıklıkta dinamik pencere boyutlandırma
6- Karn's tekrar gönderimlerin bitimini sıkıştırmış
7- Hızlı yeniden gönderim

Ölçümlerin ve β test edicilerin raporları tavsiye ediyor ki internetteki sıkışıklık durumları ile ilgilenmenin iyi olması için bu doküman arkalarındaki bağlantıları ve 1 ve 5. maddeler için açık bir tariftir. 6 numaralı algoritma BEL haberleşme enstitüsü tarafından değiştirilmiştir kaynak [1] da tanımlanmıştır. 7 numaralı algoritma AFS (Arpanet) tarafından yayınlanmıştır. Algoritma 1 ve 5 bir gözlemden ileri çıkar. TCP bağlantısının akışı veya İSOTP–4 veya Zerox NSSPP bağlantısı paketlerin korunması prensibine tabi olmalıdır. Bu prensibe tabi olursa sıkışıklık çökmesi kurallardan istisna olur. Bu sıkışıklık kontrolü muhafazayı ihlal etmek ve onları onarmakta yer bulmayı sağlar. Paketlerin korunmasıyla bağlantı için dengeyi kastediyoruz. Örneğin veri iletiminde tam pencere ile kararlı çalışmak, paket akışı fizikçiler tarafından ılımlı olarak adlandırılır. Yeni paket, eski paket ağdan ayrılmadan ağa verilmez. Akışın fiziği sıkışıklığın yüzünde bu özellikledeki sistemin düzgün olmasını haber verir. İnternetin incelenmesi kısmi olarak düzgün olmadığını gösterir. Bu zıtlık nedendir? Paket korunmasının başarısız olmasına üç neden vardır. Bağlantı dengeye

gelmemiştir veya gönderici eski paket çıkmadan yeni paket göndermiştir veya dengeye yol boyunca kaynağın sınırlarından dolayı ulaşılamamıştır. Aşağıdaki bölümlerinde bunları inceleyeceğiz kaynak [2], [3].

2.2. Yavaş Başlamayla Dengeye Ulaşma

Başarısızlık Paket kaybından sonra başlama veya yeniden başlama bağlantıdandır. Korunma özelliğine başka bir bakış söylenebilir ki gönderici 2. paketi gönderebilmesi için bilgilendirme şekil 2.1 de olduğu gibi kullanır. Alıcının bilgilendirme üretmeye başladığı zaman ağdan daha hızlı bir veri paketi alınamayacaktır.

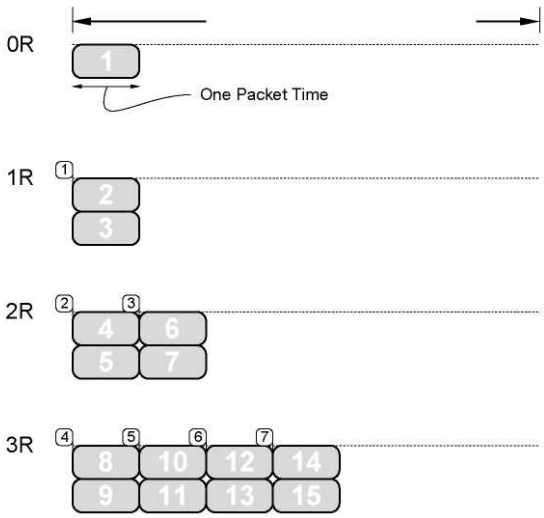


Şekil 2.1. Pencere akış kontrolü.

Yüksek bant genişliğindeki ağın gönderici ve alıcılarının şematik gösterimi daha yavaş, uzun mesafeli bir ağ bağlanmıştır. Gönderici şu anda başlamış ve arkadan arkaya bir pencere varmıştır. Bu ilk paketlerin bilgilendirilmesi göndericiye gelmek üzere sol hunide bulunan ağızdaki dikey çizgi ve dikey boyutlar bant genişliğidir, yatay boyutlar ise zamandır. Her parçalanmış kutu bir pakettir. Bant genişliği × zaman = bit'tir böylelikle kutunun alanı paket boyutudur. Bitlerin sayısı paketler ağ boyunca gittiği sürece değişmez. Böylelikle paket zamanda küçültülmüş daha küçük uzun bant genişliğine sıkıştırılır. Yoldaki en küçük bağlantıda p_b zamanı minimum paket boşluğunun genişliğini ifade eder. Paket hedefe ulaşmak için şişe boynunu geçerken paketin iç aralığında değişim olmaz. Alıcının paket aralığı $p_r = p_b$dir. Alıcı işlev zamanı bütün paketler için aynıysa alıcıdaki bilgilendirmeler arasındaki boşluk $A_r = p_r = p_b$dir. Zaman çizelgesi p_b paketler için yeterince büyük ise bilgilendirmeler içinde yeterince büyüktür ki dönüş yolu boyunca bilgilendirme aralığı korunur. Bu göndericideki bilgilendirme aralığı $A_s = p_b$ dir. Şöyle ki: ilk gönderimden sonra eğer paketler bilgilendirmenin cevabıyla birlikte gönderildiyse gönderici paketleri boşluğu yoldaki en yavaş hatta paket zamanı ile örtüşecektir.

Protokol şekil 2.1 deki gibi kendi kendini zamanlamalıdır. Kendi kendini zamanlama sistemi bant genişliğini otomatik artırır ve varyasyonları geciktirir ve geniş bir dinamik ararlık meydana getirir. TCP 800 Mbit/sn taşıma kanalından 1200 bit/sn radyo bağlantısına kadar bir aralıkta değişebileceği düşünülmesi önemlidir). Fakat aynı şey kendi kendine zamanlamalı sistemi çalışması başlamasını zorlaştırdığı zaman kararlıdır. Data akışı meydana getirmek için bilgilendirme olmalıdır fakat bilgilendirmeyi elde edebilmek içinde bilgi akışı olmalıdır kaynak [4].

Zamanlamayı başlatmak, transferdeki verinin büyüklüğünü ölçülebilir artırmak için yavaş başlama algoritmasını geliştirdik. Bunla birlikte kendimiz övüyoruz ki bu algoritmanın tasarımı ustaca bir iş Uygulaması ehemmiyetsiz ve yeni durum değişkeni ve göndericideki kodun 3 çizgisi şekil 2.2 deki gibidir.

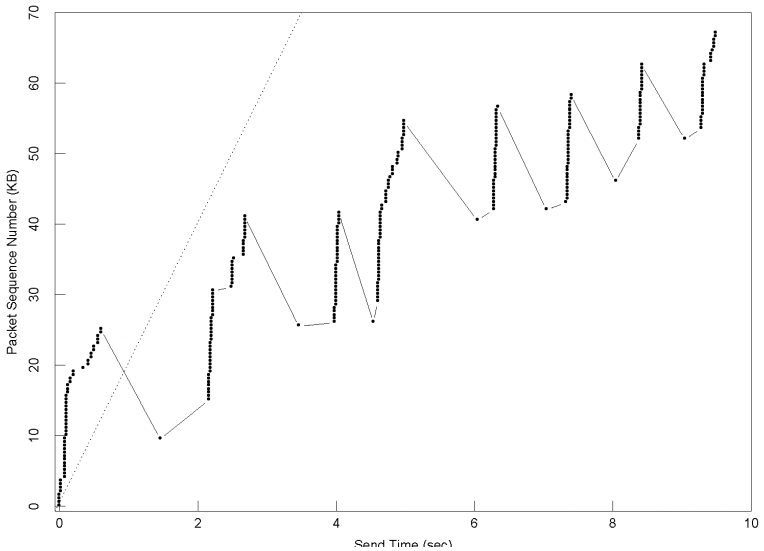


Şekil 2.2. Yavaş başlamanın kronolojisi.

Yatay yön zamandır. Devamlı zaman çizgisi bir dönüş zamanı parçasına bölünmüştür. Sayfanın aşağısına doğru artan şekilde Dikey olarak istiflenmiştir. Gri numaralı kutular paketlerdir. Beyaz numaralı kutular bilgilendirme mesajlarıdır. Her bir bilgilendirme geldiği zaman 2 paket üretilir. Birisi bilgilendirme için yeni paketin gönderildiği paket, ikincisi paketin sistemi terk ettiğini gösteren bilgilendirme ki sıkışıklık penceresini bir paketle bu bilgilendirme açar. Şekil 2.2 açıkça gösterir ki zamanda logaritmik olarak pencereye bir paket açma politikasına neden olduğunu gösterir. Eğer yerel ağ uzun mesafeli ağdan daha hızlıysa bilgilendirmelerin iki paket aynı zamanda şişe boynuna varır. Bu iki paket yığın olarak birinin en tepesinde görülür. Bunlardan birisi ağ geçidin çıkış sorgusunda bir boşluk kaplayacağını gösterir. Bu kısa sorgu logaritmik olarak ağ geçidinde artar ve şişe boynunda *w* boyutundaki paketlerin *W/2* tampon kapasitedeki paketlere dönüşmesini gerektirir.

—Sıkışıklık penceresi ekle CWND hr bir bağlantı durumuna
—Kayıttan sonra başlarken ve yeniden başlarken, CNWD yi bir palet için belirle
—Her yeni veri için her bilgilendirme CNWD yi bir paket artırır.
—Gönderirken alıcının reklâm penceresini ve CNWD sini minimum değerde gönder

Aslında yavaş başlama penceresi artışı yavaş değildir. Zamanı $R\log_2 W$ olarak alır r dönüş zamanı olduğu zaman ve w paketlerin pencere boyutu olduğu zaman şekil 2.2 deki gibi Bu performansın ihmal edilebilir etkisine sahip olmak için yeteri kadar çabuk pencere açar manasına gelir. Hatta geniş bant genişliğindeki hatlar için ürünleri geciktirir ve algoritma yoldaki maksimum mümkün olan en yüksek iki veri bağlantısını garantiler.



Şekil 2.3. TCP nin yavaş başlamasız davranışı.

2 tane san 3/50 ve san 3,5 makine arasındaki TCP bağlantısının başlangıç verileri. Bu 2 san makine ip ağ geçidi tarafından farklı bağdaştırıcılara bağlı 230 KB/sn noktadan noktaya bağlantıyı içermektedir. Bağlantının pencere boyutu 16 KB dır ve hat bant genişliği ağ geçidinde 30 tane tampon paket mevcuttur. Gerçek yol 6 tane düğüm ihtiva eder öğle geçidi bant genişliği sorgusu tam pencere için yeterli kapasitededir ama ağ geçidi sorgusu yalnız başına yeterli değildir. Her bir nokta 512 byte paket içerir. X aksı gönderilen paketlerin zamanıdır. Y aksı paket başlığındaki ardışık numaralardır. Noktaların yatay dizisi arka arkaya paketleri işaret eder ve aynı y deki ve farklı x deki 2 nokta yeniden gönderim işaret eder. Grafikteki arzı edilen davranış şeklin sol alt tarafından üst sağ tarafına doğru kısmen düzgün bir hattır. Çizginin eğimi gerekli bant genişliğine eşit olabilir. Hiçbir şey bu durumda arzu edilen davranışa benzemez. Çizgili kısım gösterir ki 20 KB/sn bant genişliği bu bağlantı için mümkündür. Bu bant genişliğinin %35 i kullanılmıştır. Geri kalan kısmı tekrar gönderimler için kullanılır. Hemen hemen her şey 54 KB den 58 KB ye kadar gönderilen 5 kez iletilir.

Yavaş başlamasız da tezat olarak 10 b/sn Eternet 56 KB/sn Arpanet ile görüştüğü zaman ağ geçidi ile ilk ağ geçidi sıçraması yol bant genişliğinde 200 kez gelen paketlerin 8 nin patladığını görürüz. Bu paketlerin patlaması devamlı yeniden göndermenin sıklıkla kalıcı bir başarısızlığına götürür.(Şekil 2.3 ve 2.4)

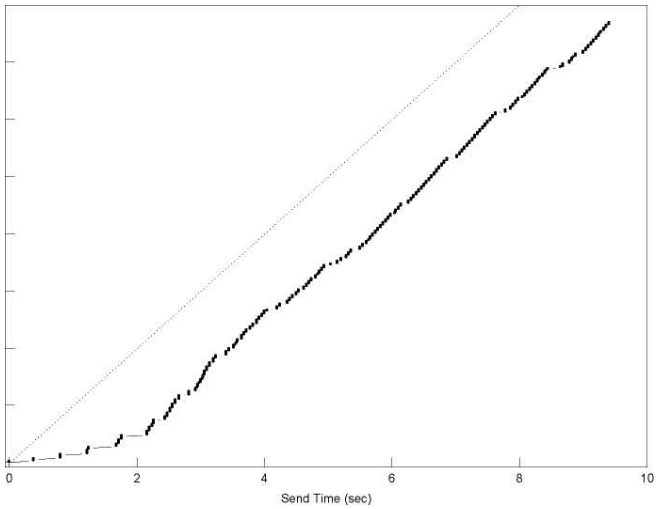
2.3. Eşitliği Korumak

Tur Zamanı: İyi bir tur zamanı tahmin edicisi yeniden gönderim zamanlayıcısının çekirdeği ağır yüklü ortamlar hariç herhangi bir protokol uygulamasının önemli bir özelliğidir. Kaynak [5] ve [6] de tipik problemde izah edildiği gibi sıklıkla yamanmıştır. Bir hata dönüşümü tahmini σ_R dönüş zamanı *r* olsun. Sorgulama teorisinde biliyoruz ki r ve r nin varyasyonları yüklenme ile çabuk bir şekilde yükselir. Eğer yüklenme p ise maksimum varan sayısı değeri *R* ve σ_R formüldeki gibi ölçülür $(1-p)^{-1}$. Bunu somutlaştırmak için ağ %75 kapasite ile çalıştığında Arpanet son nisanda çökmüş 16 faktör tarafından dönüş zamanı değiştiği tahmin edilir. TCP protokol özellikleri alçak geçirgen filtreye dayalı olarak dönüş zamanını tahmin etmeyi önerir.

$$R \leftarrow \alpha R + (1-\alpha)M$$

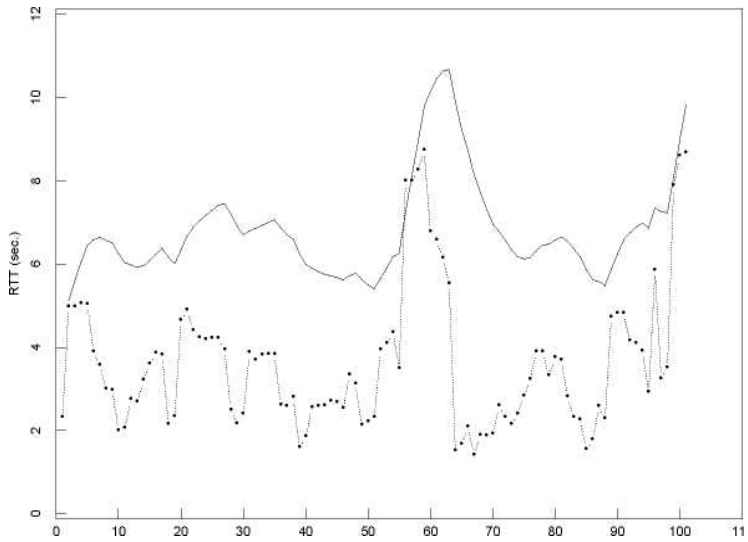
Bu formülde R en yüksek RTT tahmini değeridir. Bu formülde *M* en yakın bilgilendirme veri paketi tur zamanı ölçüm değeri ve α filtre kazanç sabiti tavsiye edilen 0,9

değeri ile *R* tahmini değeri güncellendiği zaman yeniden gönderim zaman aşımı aralığı RTO bir sonraki gönderilen veri paketinde β*R* olarak atanır.



Şekil 2.4. TCP yavaş başlama davranış başlangıcı.

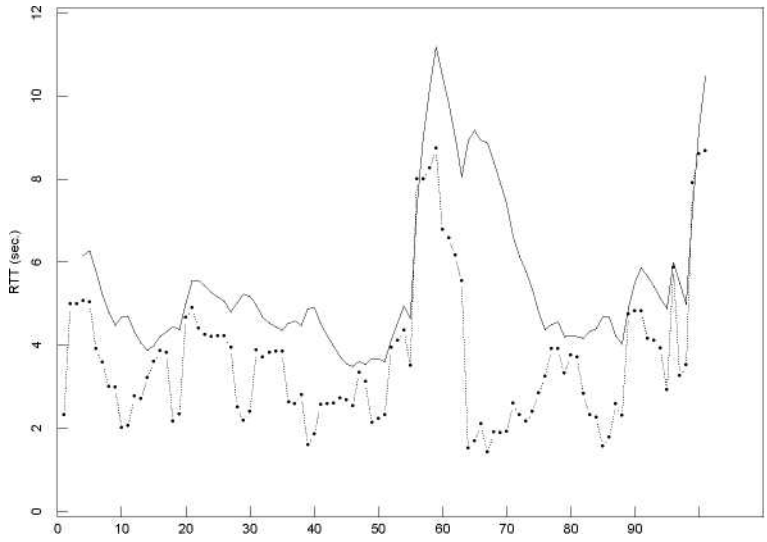
Aynı koşullarda önceki Şekil 2.3 gibi aynı günde aynı ağ yolunda aynı tamponlama ve pencere boyutu ile bunların dışındaki makinelerle 4.3 + TCP yavaş başlamayı kullanıyorlar. Yeniden gönderimde hiçbir bant genişliği harcanmıyor. Ama 2 saniye yavaş başlamada harcanıyor. Böylelikle bu bölümdeki efektif bant genişliği Şekil 2.3 dekinden 2 kat daha iyi olarak 12 KB/sn oluyor. Daha önceki şekle bakmaksızın, işlemin eğimini 20 KB/sn ve işlemin boyunu 2 saniye daha düşmesine etki ediyor. Örneğin bu işlem 1 dakika sürerse efektif bant genişliği 19 KB/sn olur. Yavaş başlamasız efektif bant genişliği ile 7 KB/sn da kalır.



Şekil 2.5. RFS 793 yeniden gönderim zamanlayıcı performansı.

İşlem verisi iyi oluşturulmuş arpa net bağlantısındaki her paketinin dönüş zamanını gösterir. X aksı paket numaralarıdır ki paketler ardışık numaralandırılmıştır 1 den başlar ve y aksı paketin gönderiminden gönderenin aldığı bilgilendirmeye kadar geçen zamanı gösterir. İşlemin bu parçasına kadar hiçbir paket kaybolmamış ve tekrar iletilmemiştir. Paketler nokta şeklinde ifade edilir. Noktalı çizgiler sırayı daha kolay takip etmek için onları bağlar. Düz çizgiler RFS 793 kuralına göre yeniden gönderim zamanlayıcının davranışını gösterir.

RTT değişimi kaynak [7] de olduğu gibi β parametresi ile hesaplanır. $\beta = 2$ değerinde % 30 yüklemeye adapte edebilir. Bu noktadan sonra paketlerin geri gönderimi artarak yüklemeye cevap verir. Dönüşümde sadece ertelenir. Bu ağı gereksiz iş yapmaya zorlar, bant genişliğini boşa harcamaya paketleri çiftlemeye zorlar. Örneğin bu ateşe benzinle gitmeğe eş değerdir. Biz ucuz bir dönüşüm metodu geliştirdik ve yeniden gönderim zamanlayıcı sonuçları sahte yeniden gönderimleri özellikle elemine eder. Sabit değerler kullanmak yerine β yaklaşımının güzel tarafı düşük yük de ve yüksek yükte performansı geliştirmesidir. Özellikle yüksek gecikmedeki yollarda uydu bağlantıları gibi (şekil 2.5 ve 2.6) başka zamanlayıcı hataları geri gönderimden sonra kaybolur. Eğer paket birden çok kere yeniden gönderilirse yeniden gönderimler nasıl yer almalıdır? Transferin son noktasında ağın bilinmeyen bir topolojisi gömülmüş ve bilinemez. Konuşmanın sayısının değişimi sabit oluyor ve sadece bir şekil çalışan logaritmik düşüş düğümüne sahip ama bunun ispatı bu yazının ispatıdır. Kaynak [1],[8],[9].



Şekil 2.6. Asıl ve değişen yeniden gönderim zamanlayıcısı performansı.

Yukarıda ki aynı veri ama sabit çizgi ek A da ki algoritmaya göre hesaplanmış yeniden gönderim zamanlayıcısını gösterir.

İspatı idare etmek için ağın iyi bir yaklaşımda, doğrusal sistem olduğunu not edin. Bu elementlerin tamamı doğrusal operatörler, gecikmeler, kazanç durumları ve benzerleri gibi davranırlar. Doğrusal sistem teorisi söyler ki sistem kararlı ise kararlığı logaritmiktir. Bu kararlı olmayan sistemleri logaritmik zamanlayıcı düşümleri gibi bazı logaritmik düşümler kararlı hale getirilebilir.

2.4. Yolu Adapte Etmek ve Sıkışıklık Gidermek

Eğer zamanlayıcı iyi bir şekle sahip ise bazı şeyler güvenebiliriz ki gecikme paketlerin kaybı veya zamanlayıcının kırılmamasındandır. Bu noktada bir şeyler bunun hakkında yapılabilir. Paketler 2 sebepten dolayı kaybolabilir. İletim sırasında zarar görmüş olabilir veya ağ sıkışmış olabilir ve yolun herhangi bir yerinde uygun olmayan tampon kapasitesine sahiptir. Birçok ağ yolunda hasar oranı % 1 den küçüktür ki ağdaki sıkışıklıktan dolayı bu medyana gelir. Sıkışıklık giderme stratejisi kaynak [10] de arz edildiği gibi 2 elamana sahiptir. Son noktaya işareti iletebilir ki sıkışıklık meydana gelir veya meydana gelmek üzeredir. Ve son nokta eğer sinyal alınmışsa sıkışıklığı düşürmek için bir politikaya sahip

olmak zorundadır. İşaret alınmamışsa araçlaşmayı yükseltmek gerekir. Eğer paket kayıpları hemen hemen her zaman sıkışıklıktan dolayı ve nerdeyse her zaman, zaman aşımı paket kayıplarından kaynaklanırsa ağın sıkışıklığının işareti için iyi bir adaya sahibiz. Sıkışıklık giderme stratejisinin başka bölümü, son düğüm hareketi DEK/İSO ve kendi TCP mize göre neredeyse aynıdır ve ağın ilk sıralama zaman seri modeline göre direk olarak takip eder. Maksimum sorgu uzunluğu tarafından ağ yükünün bazı uygun uzunluktaki sabit girişler üzerinden ölçüldüğünü kabul edelim.

Sıkışmamış ağlarda L_i girişin yükü ise örnekleme zamanı ile karşılaştırıldığı zaman L_i daha yavaş değişeceğini söyleyebiliriz. Örneğin

$$L_i = N$$

Eğer *n* sabit ise ve ağ sıkışıklığı söz konusu ise bu sıfırıncı sıra modeli bozulur. En yüksek sorgu uzunluğu 2 terimin toplamı olur. Bu hesapların yukarıdaki n değeri yeni trafiğin maksimum varış değeri için ve trafik parçası için yeni hesap terimi en son geliş zamanından ve varış zamanı trafik etkisine kadardır.

$$L_i = N + \gamma L_{i-1}$$

Taylor seri açılımın ilk 2 terimi yukarıdaki formüldür. Sonuç olarak 3 terime ihtiyaç duyulacağının bir sebebi vardır. Ağ sıkışık olduğu zaman γ büyük olmalıdır ve sorgu boyu logaritmik olarak artan şeklide başlamalıdır. Sorgu büyüklükleri çok hızlı sıkıştırıldığında trafik kaynağında sadece sistem kararlı olur. Pencere büyüklüğünü ayarlamakla pencere bağımlı protokol kaynak kontrolüne yüklenebilir. W gönderici politikası ile sonlanır. Sıkışıklıkta:

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

Örneğin pencere boyutunun çoklu aktif düşümü sıkışıklık daimi kaldığında zaman çizelgesinde logaritmik düşme meydana gelir. Eğer sıkışıklık yoksa γ sıfır civarında olmalıdır ve yük yaklaşık sabit olmalıdır. Ağ bildirimleri düşmüş paketler tarafından talep aşırı olduğu zaman bağlantı daha az paylaşım kullanıyorsa hiçbir şey göndermez. Bağlantı hali hazırdaki limitleri bulabilmek için kendi bant genişliğini yükseltmek zorundadır örneğin başkası ile yolunuzu paylaşabilirsiniz ve her bir elde edilebilir bant genişliğinde bir pencerede birleşir. Eğer kapanırsa bant genişliğinin %50 si zayi olur. Yalnızca pencere boyutunuz büyümediği zaman büyüme politikası ne olmalıdır? İlk düşünülen simetrik, çoklu aktif artırım, daha uzun zaman sabiti imkânı, $W_i = bW_{i-1}, 1 < b \le 1 * d$ hatadır. Bunun için analitik sebep bu gerçeklerle yapmaktır ki ağı doyuma götürmek kolaydır. Ama düzeltmek zordur. Bu yüksek tahmin edilmiş bant genişliği maliyetlidir. Ama logaritmik olarak zaman sabiti hemen hemen ihmal edilebilir. Pencere boyutundaki sabit, küçük değişiklikler yapmakla tahmin etmeden en iyi yükselme politikasını belirleyeceğiz. Sıkışma olduğu zaman şu eşitlik geçerlidir:

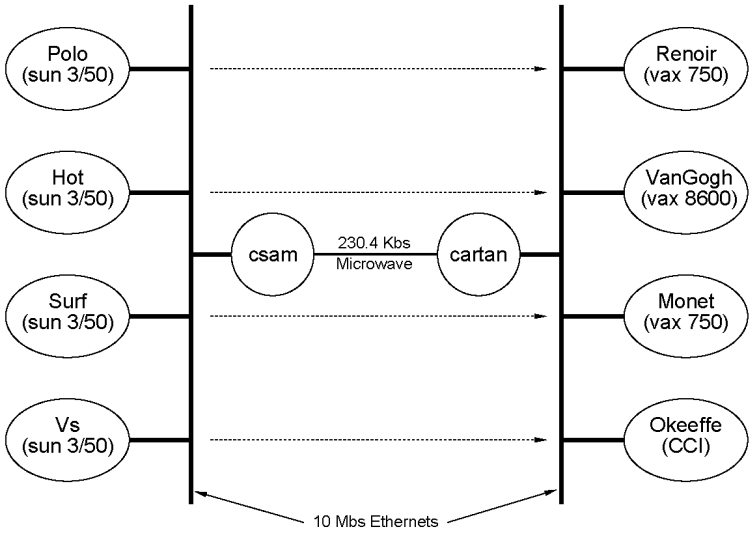
$$W_i = W_{i-1} + u \quad (u << W_{\max})$$

Hat bant genişliği, üst protokol örneğin en geniş yüklenmemiş yol için makul penceresi $W_{\max}$ eklenebilir artırımdır. Çoklu aktif düşüş politikası tavsiye edilir ve TCP de bu politika uygulanır. Bu iki uygulamanın tek farkı *d* ve *n* sabitlerinin seçimidir. Burada 0,5 ve 1 değerinin kullanıyoruz. Yazının devam eden kısmında bütün analiz yer almaktadır. Sıkışıklık kontrol algoritması için bu konu önemsiz gibi gözükebilir ama değildir. Yavaş başlama gibi 3 kot çizgisi vardır. 1-CWND ile Herhangi bir zaman her aşımında hali hazırdaki pencere boyutunun yarısını yap (bu çoklu aktif düşümdür). 2- yeni veri için bilgilendirme CWND yi 1/* CWND kadar artırır, bu eklenebilir artırımdır. 3- gönderirken alıcı reklâm penceresini ve CWND yi minimum gönder. Bu algoritmanın sıkışıklık giderme olduğunu not edin. Daha

önceden tanımlanan yavaş başlamaya dâhil değildir. Sinyal karışıklığını paket kaybettiğinde yeniden başlamaya sebep olacaktır. Yukarıdakilere ek olarak yavaş başlama nerdeyse hemen hemen gerekli olacaktır. Ama her sıkışıklık giderme ve yavaş başlamada zaman aşımı tarafından tetiklenir ve sıkışıklık penceresini artırır. Bunlar sıklıkla karışmıştır. Bunlar aslında tamamen farklı nesneler ile bağımsız algoritmalardır. Farkı anlayabilmek için iki algoritmada ayrı olarak ele alınmıştır. Fakat pratikte beraber uygulanmalıdır. Ek B de yavaş başlama sıkışıklık giderme algoritmalarını beraber tanımlanmıştır. Şekil 2.7 den 2.12 ye kadar sıkışıklık gidermeli ve gidermesiz TCP bağlantılarının davranışlarını gösterir. Sıkışıklığı benzetmek için Test şartlarına rağmen örneğin 16 KB pencereler iletilebilir. Bu test senaryosu pratiktekinden çokta uzak değildir. Arpanet IMP uçtan uca protokolü herhangi ağ geçidi çiftleri arasında 8 paketin gönderilmesine izin verir. Varsayılan 4,3 BSD boyutu 8 pakettir (4 KB). Bu ardışık gönderme Barkaly deki herhangi 2 kaynakta ve bitteki herhangi iki kaynakta UCB-MİT IMF yolunda ağ kapasitesini aşabilir ve gösterildiği gibi davranabilir. Kaynak [11], [12].

2.5. Ağ Geçidi Taraflı Sıkışlık Kontrolü

İletimin son noktasındaki algoritma ağ kapasitesinin aşmadığını garantiler. Bunlar bu kapasitede uygun paylaşımı garantilemez. Sadece ağ geçitlerindeki akışların sıkışıklığında paylaşımları kontrol etmek ve uygun yerleri kontrol etmek için yeterli bilgi var mı? Bit sonraki adımla ağ geçidinin sıkışıklık belirleme algoritmasını görürüz kaynak [7].



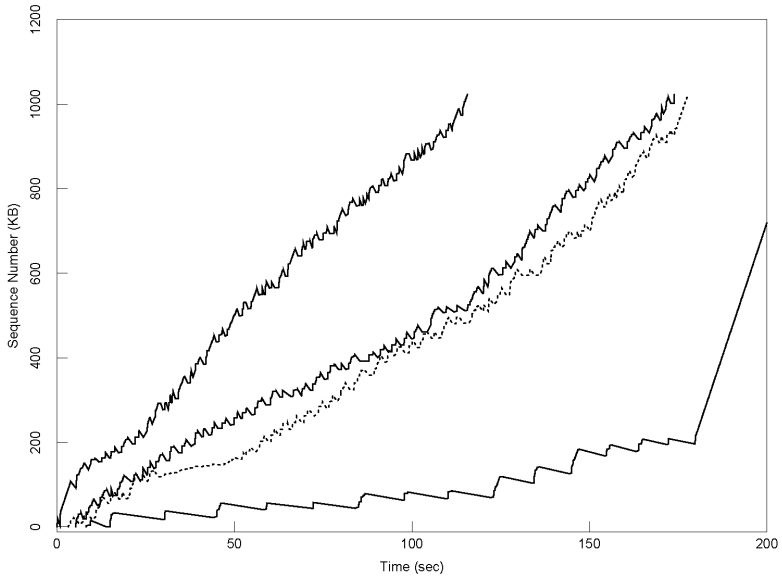
Şekil 2.7. Çoklu iletişim test kurulumu.

Çoklu ardışık TCP iletişim paylaşımı şişe boynu bağlantılı etkileşimi bu kurulumla test edilecek. 1 MB transfer (2048–512 byte veri transfer paketleri) 3 saniyede başlatılmıştır. LBL deki 4 makineden ayrı olarak UCB deki 4 makineye her makinede ki 1 iletim çifti yukarıda noktalı olarak gösterildiği gibi bütün trafik 234 KB/sn İP yönlendiricisine bağlanarak gitmektedir. LBL deki Cısam dan UCB deki Cartan İP yönlendiricisine mikro dalga bağlantı sorgusu 50 paketi tutabilir her bir bağlantı 16 KB (32-512 byte paket) pencereye verilir. Bu herhangi iki bağlantı mümkün olan tampon lamayı aşabilir ve 4 bağlantı % 160 oranında sorgu kapasitesini aşabilir.

Bu algoritmanın amacı mümkün olduğunca kısa süre son noktaya sinyali göndermektir. Ama ağ geçidinin trafiksiz kalacağı kadar erken değil. Paket düşümlerini sıkışıklık sinyali olarak kullanmaya devam etmeyi planladığımız zaman ağ geçidi kendi

kendini yanlış davranmaktan kurtarır. Sunucu basitçe düşmüş paketlere sahiptir. Paketlerin birden çok uygun bağlantı kullanmaya çalıştığını ağ geçidi söyler. Bu son nokta algoritmasına benzer. Ağ geçidi algoritması sıkışıklık gidermeyle son nokta değiştirilmese dahi sıkışıklığı azaltmalıdır. Ve düğümler en az sayıdaki paket düşümü ve uygun paylaşım bant genişliğine sahip olacaktır. Sıkışlık giderme uygulandığında

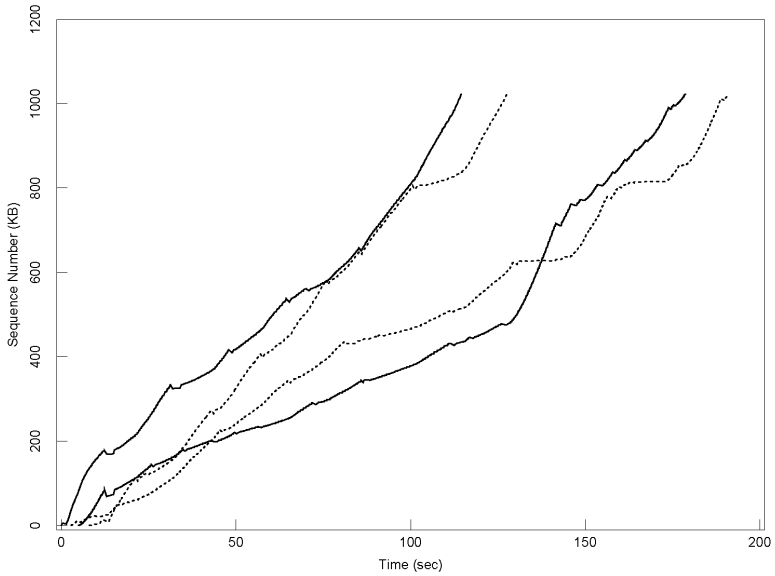
Logaritmik olarak sıkışıklık arttığında erken tanımlamak önemlidir. Eğer erken tanımlanmışsa göndericinin küçük artırımlarıyla pencereler bunu tamir eder. Bunun dışında çok büyük artırımlar ağa yeterli yedek kapasiteyi vermek için gereklidir. Ama trafiğin bir anda patlama eğilimi gerçekçi planlama saçma olmayan bir problemdir. Jain sorgu yeniden oluşturma noktaları arasındaki azami değere dayanarak bir şekil amaçlamıştır. Bu iyi bir patlama filtrelemesine dayanır. Ama yüksek yük altında veya uygun 2. Sıralı dinamikte dönüşüm problemi olduğunu düşünebiliriz. Armaks modelindeki dönüş zamanı / sorgu uzunluğu tahminindeki daha önceki çalışmalarımız gibi kullanmayı planlarız. Kaynak [13].



Şekil 2.8. Çoklu ardışık TCP ler sıkışıklık gidermesiz.

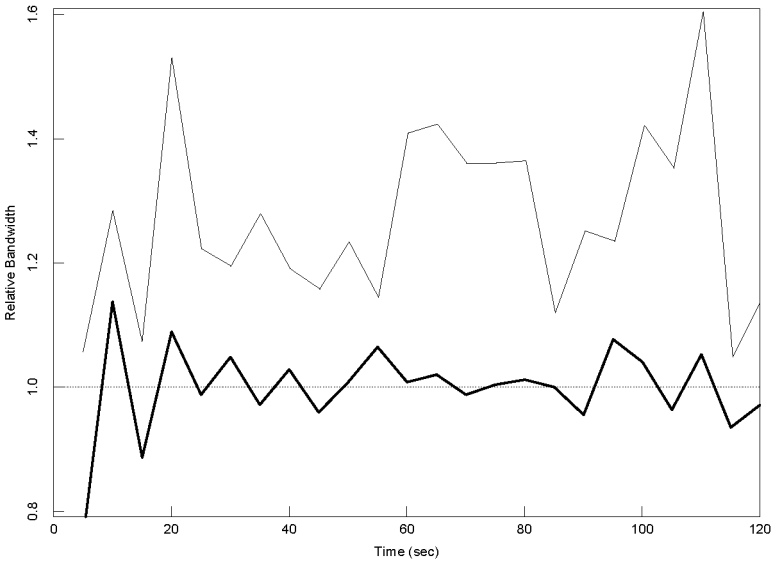
4 ayrı TCP haberleşmesinden sıkışıklık giderme yol üzerinde şekil 2.7 üzerinde gösterilmiştir. 11 000 paketin 4000 i yeniden iletim için gönderilmiştir (örneğin paketlerin yarısı yeniden iletilmiştir). Bağlantı bant genişliği 25 KB/sn olduğunda her bir 4 haberleşme 6 KB/sn la iletilmelidir. Bunların dışında bir haberleşme 8 KB/sn a sahiptir. İkisi 5 KB/sn bir diğeri 0,5 KB/sn ve 6 KB/sn kaybolmuştur.

Başlangıç sonuçları bu ilerlemenin yüksek yüklemede başarılı çalışacağını tavsiye eder, trafikte İkinci sıralı etkilerden muaftır ve her saniye kilo paketleri yeteri kadar yavaşlatmaz.



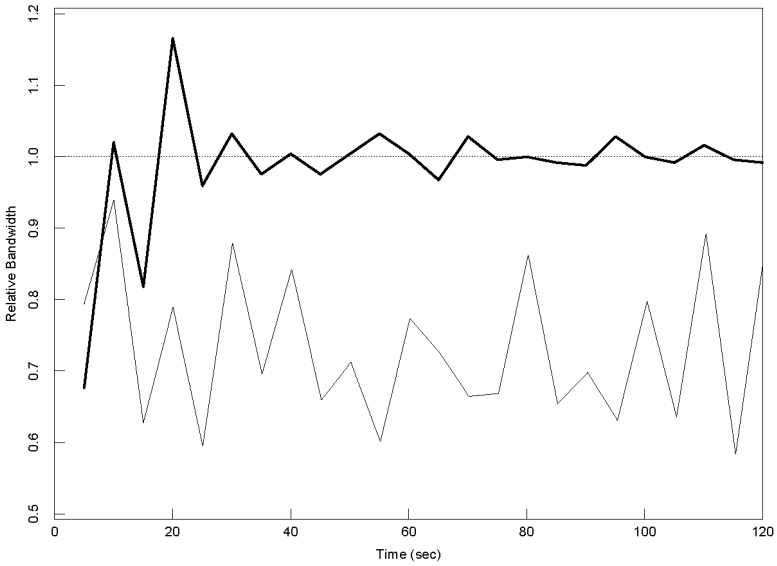
Şekil 2.9. Sıkışıklık gidermeli çoklu ardışık TCP ler.

Şekil 2.7 de gösterilen yol üzerinden sıkışıklık giderme kullanılarak 4 farklı TCP iletişimi gerçekleştirilmiştir. 8281 paketin 89 u yeniden iletim için gönderilmiştir. Örneğin paketlerin %1 yeniden gönderilmek zorunda kalınmıştır. İletişimlerin 2 tanesi 8 KB/sn ve 2 tanesi 4,5 KB/sn dır. Örneğin bütün bağlantı bant genişlikleri şekil 2.11 de hesaplanmıştır. Yüksek ve düşük bant genişliği göndericileri arasındaki fark alıcıdan kaynaklanır. 4,5 KB/sn' lik göndericiler 4,3 BSD lik alıcılarla haberleşir. Bu ACK nin %35 penceresinin solmasına veya 200 mili saniye geçmesine kadar gecikir. Örneğin 5 ya da 7 paket ACK gecikmesi maksimum değerde ise. Bu göndericinin her bir bilgilendirmede 5 ile 7 paket patlamanın iletilmesi manasına gelir. 8 KB/sn göndericiler 4,3 + BSD alıcılarla haberleşirler bu bilgilendirmenin en fazla bir paketini geciktir. Bilgilendirmenin saat kuralından dolayı yazan minimum bilgilendirme frekansının diğer bütün paketlerde olmasına inanır. Örneğin gönderici en son 3 paketi iletebilir: kayıp olma ihtimali aniden yükselir. Göndericinin Eski tip alıcılar ile konuşması 3 kez kayıp değeri yeniden gönderim beklemesinin daha fazla zaman harcayacağı anlamına gelir ve sıkışıklık gidermeden dolayı en büyük pencere boyutu daha küçülür (Daha yüksek kayıp değerleri 1,8 veya 0,500 üçe katına çıkabilir).



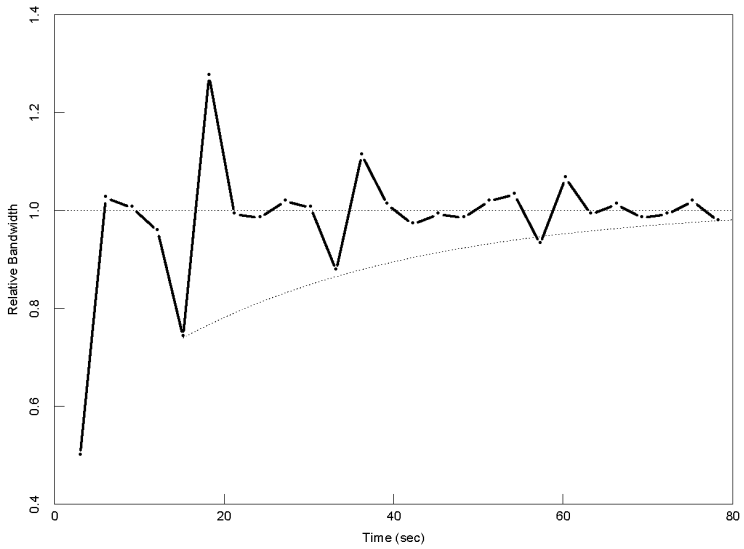
Şekil 2.10.Yeni ve eski TCP ler tarafından kullanılan toplam bant genişliği.

Sıkışıklık giderme kullanılmadan 4 adet gönderici tarafından kullanılan toplam bant genişliğini ince çizgi gösterir. Maksimum 5 saniye olan değerler ve 25 KB/sn bağlantı bant genişliğine normalleştirilir. Kablodaki göndereceğinden %25 daha fazla maksimum değerde göndericinin gönderebileceğini not edin. Kalın çizgi sıkışıklık giderme ile gönderici için aynı verileri içerir. Her TCP nin doğru pencere boyutunda küçük başlamadan dolayı bulduğunda İlk 5 sn için veri düşüktür. Sonra 20 sn civarında sıkışıklık kontrolü ayarlamadan dolayı ani artış meydana gelir. Kalan zamanda gönderici kanalın kablonun bant genişliğinde hareket eder (110 sn civarındaki aktivasyon bant genişliğidir). 80 sn civarındaki aktivite şekil 2.9 deki düz spotun yansımasıdır.



Şekil 2.11. Yeni ve eski TCP lerin etkili bant genişliği.

Şekil 2.10 eski TCP lerin şişe boyunu bant genişliğinden % 25 den fazla kullandığını göstermiştir. Bu şişe boynu sorgusu dolduğu zaman göndericilerin % 25 i çıkarılır. Eğer bu çıkarım yeniden iletilmişse 25 KB/sn nin tamamı bağlı bant genişliğine iletilir. Örneğin onların davranışları sosyal olmayabilir ama kendi kendini yıkıcı değildir. Ama şekil 2.8 de bağlı bant genişliklerinin % 25 civarı sayılmamıştır. Şimdi her 5 sn için girişlerin toplam veri bilgilendirmelerini maksimuma getirdi, bu etkili veya iletilmiş bağlantının bant genişliğini verir. İnce çizgi yine eski TCP yi gösterir. Bağlantı bant genişliğinin % 75 veri için kullanılır. Yeniden iletilme ihtiyacı duyulmayan yeniden iletim paketleri tarafından hatırlatmalar kullanılmalıdır. Kalın çizgide yeni TCP ler için alınmış bant genişliğini gösterir. Aynı yavaş başlama vardır ve bağlantı bant genişliğinde operasyonu uzun dönem boyunda takip eden bir geçici durum başlar.



Şekil 2.12. Pencere ayarlama detayları.

En uzun zaman 5 sn olduğundan dolayı eski TCP verilerinde tepeleri yumuşatmak gerekir. Sıkışıklık giderme pencere politikası şekil 2.10 da ve 2.11 de yapılması zordur. Burada 3 sn süreliğine sıkışıklık kontrolü için veri bilgilendirmeyi etkili bir şekilde gösteriyoruz. Paket düşürüldüğü zaman gönderici pencere dolana kadar gönderir. Sonra yeniden gönderimin zaman aşımı dolana kadar durur. Düşen paketten sonra alıcı bilgilendirme verisini gönderemediğinde bu çizimden büyüklüğü gönderenin pencere büyüklüğüne eşit olan negatif tepe görmeyi umarız. Eğer bir sonraki girişte yeniden gönderim gerçekleşirse aynı büyüklükteki pozitif tepe görmeyi umarız. Bu tepelerin yüksekliği gönderici tepe boyutunun direk ölçümüdür. Veri açıkça 15, 33 ve 57. saniyelerde 3 olayı da gösterir ve pencere boyutu logaritmik olarak düşer. Noktalı çizgiler 6 pencere boyutunda kareye dolar bu olayda belirtildiği gibi dolma zamanı değişkeni 28 sn dir. Ağ geçidindeki uzun zaman sabiti sıkışıklık giderme algoritmasının eksikliğinden kaynaklanır. Ağ geçidinde çalışan düşme algoritması ile zaman sabiti 4 sn civarında olur.

2.6. RRT ve Varyasyon Dönüşüm İçin Hızlı Algoritma

2.6.1. Teori

Asıl dönüş zamanı yaklaşımı için RFC 793 algoritması tahmin edici sınıfın en basit bir örneğidir. Geçen 20 yıl zarfında bu algoritma yaklaşımları kontrol teorisini devrimselleştirmiştir kaynak [14]. RTT nin (Run Trip Time – gidiş-dönüş zamanı) yeni ölçüm değeri *m* verilmiştir. TCP aşağıdaki formülle maksimum RTT yaklaşımına güncellenmiştir.

$$a \leftarrow (1-g) + gm$$

Bu formülde *g* kazanç demektir ve 0 ile 1 arasındadır, sinyalin gürültü gücüne bağlıdır. Bu daha hassas yapar ve daha hızlı hesapla sağlar. Aşağıdaki formülü elde edebilmek için g ile terimleri çarpıp yeniden sağlarsak

$$a \leftarrow a + g(m-a)$$

Bir sonraki ölçümün tahmini *a* değeri olarak düşünelim *(m-a)* bu tahmindeki hatadır. Ve yukarıdaki formül tahmin hatasındaki çeşitli parçalar ve eski tahmin değerine dayana yeni bir yaklaşım yapılabileceğini söyleriz yukarıdaki formülle. Tahmin hatası iki elemanın toplamıdır. Birincisi ölçümdeki gürültüden kaynaklanan rasgele, tahmin edilemeyen hata

ikicisi *a* nın yanlış seçiminden kaynaklanan hatalar. Rasgele hata E_r yi çağırır ve E_e yaklaşık hatasına

$$a \leftarrow a + gE_r + gE_e$$

eşittir. gE_e terimi a yı verir. Sağ taraflındaki yükselme gE_r nin rasgele taraftaki yükselmeyi verir. Birçok örnekten sonra rasgele vuruşlar birbirini iptal eder. Bu algoritma doğru maksimum bir noktada birbirine yaklaşmaya mail eder. Fakat g bir uzlaşmayı tasvir eder. Biz büyük g değeri istedik $E_e$yi etkilemeyecek şekilde ama küçük gE_r den kaynaklanan hasarları azaltması için. E_e terimi gerçek maksimum değere gittiğinde g nin hangi değerini kullandığınız önemli değildir. Hemen hemen her zaman daha büyük değerler yerine daha küçük değerleri kullanmak daha iyidir. Tipik olarak kazanç 0,1 ile 0,2 arasında seçilir. Kazancı toplamadan önce ham verinize uzun bir bakış için iyi bir fikirdir.
Açıktır ki doğru averaj ve anın standart sapması arasın da *a* rasgele olarak salınır. Aynı zamanda *a* logaritmik olarak doğru averaj değerinde birleşmeye değerinde 1/g zaman sabitinde. Böylelikle daha gE_r kararlı a yı verir. Doğru averajı elde etmek için daha uzun zaman harcanır.
M deki farklı ölçüm değerlerini almak istersek TCP yeniden gönderim zamanlayıcısına iyi bir değer hesaplamasını söyleyin. σ^2 değişimi kalıcı bir seçimdir. Çünkü matematiksel bazı özellikleri vardır. Değişimi hesaplamak *(m-a)* nın karesini almayı gerektirir. Böylelikle bunun için bir tahmin edici bir tam sayı değeriyle çarpılmasını gerektirir aynı zamanda birçok uygulama aynı ünitede ve a ve m deki gibi değişimi ister. Böylelikle değişimin karekökünü kullanmak için zorlanırız. Değişim ölçümü asıl tahmin veya sapma hatasını hesaplamak için kolaydır. $(m-a)$ nın ortalamasının maksimumu aynı zamanda

$$mdev^2 = \left(\sum |m-a|\right)^2 \geq \sum |m-a|^2 = \sigma^2$$

ilk sapma standart sapmadan daha tutucu yaklaşım değişimde olduğu zaman.
Sdev ve *mdev* arasında basit bir ilgi vardır genellikle örneğin tahmin hatası normal dağıtılmışsa $mdev^2 = \sqrt{\pi/2}sdev$ dir. *Sdev* den *mdev*'e gitmenin faktörü $(\sqrt{\pi/2} \approx 1.25)$ civarındadır. *Mdev sdev* in iyi bir yaklaşık değeridir ve hesaplaması daha kolaydır.

2.6.2. Pratik

Hızlı tahmin ediciler ortalama a için ve asıl sapma v için verilmiş olan ölçüm m yukarıdan takip eder. Her tahmin edici RFS 793 algoritmasının 2 durumun olduğu anlamına gelir.

$$Err \equiv m - a$$
$$a \leftarrow a + gErr$$
$$v \leftarrow v + g(|Err| - v)$$

daha hızlı hesaplamak için yukarıdaki eşitlikl tam sayı aritmetiğinde olmalıdır ama eşitlik g<1 parçasını içerir böylelikle bazı ihtiyaç duyulan ölçümler her şeyi tam sayı değeri olarak tutmaya ihtiyaç duyar. İkin karşılıklı kuvveti örneğin $g = 1/2^n$ (bazı *n* ler için) genel olarak iyi bir seçimdir. Ölçüm öteleme ile uygulanabildiğinde: ½ ile çarpmak aşağıdaki eşitliği verir.

$$2^n a \leftarrow 2^n a + Err$$
$$2^n v \leftarrow 2^n v + (|Err| - v)$$

Hatayı minimize etmek için *a* ve *v* nin, *sa* ve *sv* nin ölçülmüş versiyonlarını ölçülmemiş versiyonlarına göre tutmak gerekir. $g = .125 = \frac{1}{8}$ 'de tutmak c de bunu belirtmek

$$m -= (sa >> 3);$$

$$sa += m;$$

$$eger(m < 0)$$

$$m = -m;$$

$$m -= (sv >> 3);$$

$$sv += m;$$

a ve v için aynı kazancı kullanmak gerekli değildir. RFS 793 de 0,1 e yaklaşmak tavsiye edilir. Zamanlayıcıyı güçlendirmek için cevabı daha hızlı getirmek için RTT yi değiştirmek için *v* ye daha geniş kazanç vermek iyi bir fikirdir. Kısmen pencere gecikmelerinden dolayı zamanlayıcıyı güçlendirmek ve cevabı daha hızlı getirmek için RTT yi değiştirmek ve v ye daha geniş kazanç vermek iyi bir fikirdir. Özellikle pencere gecikmesinin yanlış seçiminden dolayı pencere boyutunun çarpımı RTT yapısının tam değeri vardır. Bunu filtrelemek için a tahmincisinde 1/g en azından pencere boyutu kadar geniş ve 1/g v tahmincisinde pencere boyutundan daha küçük olabilir.

0.25 kazancını kullanarak sapmada ve yeniden gönderim zamanlayıcısı hesaplamada, RTO a+4v olarak son zamanlayıcı kodu aşağıdakine benzer.

$$m -= (sa >> 3);$$

$$sa += m;$$

$$eger(m < 0)$$

$$m = -m;$$

$$m -= (sv >> 2);$$

$$sv += m;$$

$$rto = (sv >> 2) + sv;$$

Sıklıkla bu hesaplama yaklaşık RTO ya doğrulanır. *sa* kesmesinden dolayı m-a yı hesapladığında, *sa* bir sonraki değere yuvarlanmış doğru asıl değere yaklaşır. *Sv* de buna benzer şekildedir. Böylece ortalamada her birinde eğimin yarısı vardır. RTO hesaplaması bu değerin yarısına yuvarlanmalıdır ve saatin yaklaşımı ile rasgele faz göndermek için hesaba bir bölme eklemeye ihtiyacı vardır. Böylece 1.75 bölme eğimi yardımı 4v de yaklaşık olarak hesaplanan yarım bölme yuvarlanması + 1 bölme faz doğrulamasına eşittir.

2.7. Sıkışıklık Giderme Algoritması İle Yavaş Başlamanın Karışımı

Gönderici sıkışıklık kontrolü için iki algoritma arasında atlamak da iki durum değişkeni gönderici tutar: yavaş başlama/ sıkışıklık pencere, CWND ve eşik boyutu, *ssthresh.* Göndericinin çıkış rutini her zaman CWND nin minimumunu gönderir ve pencere alıcı tarafından sunulur. Zaman aşımında hali hazırdaki pencere boyutunun yarısı *ssthresh* de kayıt edilir. Bu sıkışıklık giderme algoritmasının çoklu aktif düşümünün parçasıdır. Daha sonra CWND bir pakete e ayarlanır bu yavaş başlamayı başlatır. Yeni veri bilgilendirildiğinde gönderici

$$eger\,(cwnd < ssthresh)$$
$$cwnd += 1;$$
$$\deg ilse$$
$$cwnd += 1/cwnd;$$

Böylelikle yavaş başlama Sıkışıklık giderme güvenli operasyon noktası olduğunu düşündüğünde biran önce başlar. Daha sonra sıkışıklık giderme biter ve yolda mümkün olan daha büyük bant genişliğinin olmasını sorgulamak için yavaşça pencere boyutunu artırır. Not edin ki eğer CWND tam sayı ölçeklendirilmediğinde ise yukarıdakinden başka durumlar yanlış işler. Bir paket parçası örneğin eğer maksimum pencere yol için w paketi ise CWND 0 ile w aralığını kapsamalıdır en azından 1/w çözümü ile. Paketlerin gönderimi maksimum iletim biriminden yol için daha küçük olduğunda etkinliği azaltır. Uygulayıcı dikkat etmelidir ki parçalı bölmeli CWND küçük paketler de gönderilimi sonuç vermez. Nedensel TCP uygulamasın da içeren saçma pencere giderme kodu küçük paketlerden korumalıdır. Ama bu noktada dikkatlice kontrol edilmelidir.

2.8. Tur Zamanı İle Pencere Ayarlama Etkileşimi

Bazı TCP bağlantıları çok düşük hızlarda olduğu zaman özellikle örneğin çevirmeli ağ yeniden iletim zamanı ve sıkışıklık pencere ayarlaması arasında bir etkileşim meydana getirebilir. Ağ yolları 2 sınıfa bölünebilir. Gecikme baskın, sakla ve ilet ve/veya aktarma gecikmeleri RTT yi belirler. Ve baskın bant genişliğinde bağlantı bant genişliği ve ortalama paket boyutu RTT yi belirler baskın bant genişliğinde bant genişliğinin yolu ve sıkışıklık giderme pencere artımı Δw RTT yi artırır.

$$\Delta R \approx \frac{\Delta w}{b}$$

Eğer RTT değişkeni yolu V küçük veri $\Delta R \;\; 4V$ yi aşabilir. Yeniden gönderim zaman aşımı meydana gelir ve birkaç turdan sonra SSTHRESH küçük değerlerle biter. RTO hesaplaması yavaş başlama sırasında çok büyük yeniden gönderim zaman aşımı tiplerinden korunmak için tasarlanmıştır. Özellikle RTT değişkeni ve RTO hesaplamasında 4 ile çarpılmıştır. Şimdi açıklanacak sebep yüzünden; yeniden gönderim i, RTO i yavaş başlamasının sonunda yeniden gönderim zaman aşımı hesaplanmışsa bir sonraki dönüşte asıl RTT eşit ya da küçüktür. Gecikmenin en kötü durumu pencereden kaynaklanan durumdur. Her bir çevrimde R ikiye katlar (pencere boyutu ikiye katlandığı zaman) buda $R_{i+1} = 2R_i$ (R_i yavaş başlamanın i. seferinin RTT değeridir) ama

$$V_i = R_i - R_{i-1} = R_i / 2$$

ve

$$\begin{aligned} rto_i &= R_i + 4V_i \\ &= 3R_i \\ &> 2R_i \\ &> R_{i+1} \end{aligned}$$

bu durumda yeniden gönderim zaman aşımı meydana gelmez. Pencereden büyümesinden dolayı yeniden gönderim zaman aşımı sıkışıklık giderme pencere artımından dolayı meydana gelebilir. Sadece paket artırımı pencerede değişebilir değişebildiğinde. Böylelikle paket boyutu s için birçok s–1 paketleri olabilir. Herhangi bir v artımı için yeteri kadar uzundur en son pencere artımımdan dolayı hiçbir şeyi bozmaz ama bu problem baskın bant genişliği

yolundan farklı olarak. Artırımlar 12 paketten daha fazla olduğunda yol için saçma büyük bir pencereye uygulanır. Asıl sapma tahmini v'nin filtre zamanın bozulma zamanı kazancı RTO hesaplamasındadır. Burada bir kişi bu zaman aşımını göz ardı etmelidir ve onların etkileri yol için daha uygun bir şeyler olması için basitçe pencereyi düşürecektir.

Bunla birlikte yavaş başlama ve sıkışıklık giderme bu tür yeniden gönderimleri tetiklememek için tasarımlanmıştır. Daha yüksek seviye protokoller ile etkileşim sıklıkla şöyledir: uygulama protokolleri SMTP gibi görüşme açısını vardır. Küçük paketler durup ve bekleyip değişebilir, örneğin bütün mail mesajları ve haberlerin metinleri gönderildiği zaman veri transferi ile devam edebilir. Maalesef görüşme sıkışıklık penceresini açar böylelikle veri transferinin başlangıcı ağa yavaş başlama olmadan birkaç paket düşer. Ve baskın bant genişliği yolunda RTO dan daha hızlı bu paketler sonucunda RTT artımı kaydedilebilir. Yavaş başlama eğer TCP uygulaması faz değişimini tespit ederse aynı zamanda bu problemi giderir. Bu algılama basittir çünkü en azından bir tur zamanı için hiçbir şey göndermediğimizden dolayı hat boştur. RTT yi görmek için diğer yol son gönderilenden sonra hattı boşaltmak için geçen zamandır. Böylelikle en azından bir RTT için hiçbir şey gönderilmemişse yavaş başlamayı güçlendirmek için bir paketi bir sonraki gönderim CWND yi ayarlar. Örneğin bağlantı durum değişkeni *lastsnd* son paketin gönderildiği zamanı tutarsa takip eden kod TCP çıkış rutininde erken görünmelidir:

$$\text{int } idle = (snd\_\max = snd\_una);$$

$$eger\,(idle\,now - lastsnd > rto)$$

$$cwnd = 1;$$

eğer iletimde hiçbir veri yoksa gönderilen bütün veriler bilgilendirilmişse sonuçsuzluk doğrudur. Böylece eğer "iletimde hiçbir şey yoksa ve uzun süredir hiçbir şey göndermemişse yavaş başlama". Bizim deneylerimiz ister hali hazırdaki RTT varsayımı, ister RTO yaklaşımı uzun süre için kullanılabilir.

2.9. Pencere Ayarlama Politikası

Düşürme terimi olarak ½ yi kullanmanın sebebi 7/8 e karşı olarak 15 de, aşağıdaki el dalgası vardır: paket düştüğünde ya başlıyor olursunuz veya düşüşten sonra yeniden başlıyor olun ya da değişiklik göstermez gönderim durumunda olursunuz. Eğer başlarsanız siz bilirsiniz ki hali hazırdaki pencere boyutunun yarısı çalışır. Örneğin pencerenin paketlerinin değeri kayıpsız değiştirilmiştir ki yavaş başlama bunu garanti eder. Bu sıkışıklık ta pencereyi en geniş boyutuna ayarlarsınız ki boyutunu yavaşça artırarak çalışırsınız. Eğer bağlantı şaşmadan çalışıyorsa ve bir paket düşmüşse yeni bir bağlantının başladığı muhtemeldir ve bant genişliğinizin bir kısmını alır. Genellikle $p<0.5$ ile ağımızı çalıştırırız. Böylece olasıdır ki bant genişliğini paylaşan iki iletişim açıkça vardır. Örneğin pencerenizi yarıya düşürmelisiniz çünkü bant genişliği sizin için yarıya düşmüştür ve aynı bant genişliğini paylaşan ikiden daha fazla iletişim varsa pencerenizi yarılamak muhafazakârdır. Bununla birlikte iki değişikliğin faktörü pencere boyutunda büyük performans cezası olarak görülür. Sistem terimlerinde maliyet ihmal edilebilir: sadece geniş sorgu oluştuğunda hali hazırda ki paketler düşürülür. İSO IP ile dahi "sıkışıklık tecrübeli" bit göndericileri güçlendirmek için ve pencereleri düşürmek için sorguya takılırız çünkü sorguyu dağıtmak için aşırı olmayan bant genişliği mümkün olmasıyla şişe boynu %100 kapasiteyle çalışmaktadır. Eğer paketler karışmışsa bazı göndericiler 2 RTT için kapanır. Açıkça sorguyu boşaltmak için zamana ihtiyaç duyulur eğer bu göndericiler doğru pencere boyutu ile yeniden başlarsa sorgu yeniden oluşmaz. Sistem herhangi bir şişe boynu bant genişliği kaybetmeden gecikme minimuma düşer. Bir paket artırımı 0.5 azaltışından daha az tatmin edicidir. Aslında bu kesinlikle hemen hemen çok geniştir. Eğer algoritma w pencere boyutunu birbirine yaklaştırırsa $O(w^2)$ paketleri

eklenebilir artırım politikası ile düşümler arasında vardır. %1 den küçük değerde ortalama düşüşler için vururuz ve buluruz ki arpa nette dört ağın test ettiğimiz en kötü durumu, pencereler 8-12 pakete yaklaşır buda bir paketin %1 artırımı için ortalama düşüş değerine götürür. Ama ağ geçidinde hiçbir şey yapmadığımız zaman pencere maksimuma yaklaştırırız. Ağ geçidi paketleri düşmeden kabul edebilir.

KAYNAKLAR

[1]. Karn, P., and Partridge, C. Estimating round-trip times in reliable transport protocols. InProceedings ofSIGCOMM '87 (Aug. 1987), ACM.

[2]. Borrelli, R., AND Coleman, C. Differential Equations. Prentice-Hall Inc., 1987.

[3]. Luenberger, D. G. Introduction to Dynamic Systems. John Wiley & Sons, 1979.

[4]. Jain, R. A timeout-based congestion control scheme for window flow-controlled networks. IEEE Journal on Selected Areas in Communications SAC-4,7 (Oct. 1986).

[5]. Zhang, L. Why TCP timers don't work well. In Proceedings ofSIGCOMM '86 (Aug. 1986), ACM

[6]. Jain, R. Divergence of timeout algorithms for packet retransmissions. InProceedings Fifth Annual International Phoenix Conference on Computers and Communications (Scottsdale, AZ, Mar. 1986).

[7]. Clark, D. Window andAcknowlegement Strategy in TCP. Arpanet Working Group Requests for Comment, DDN Network Information Center, SRI International, Menlo Park, CA, July 1982. RFC-813.

[8]. Edge, S. W. An adaptive timeout algorithm for retransmission across a packet switching network. In Proceedings ofSIGCOMM '83 (Mar. 1983), ACM.

[9]. Hajek, B. Stochastic approximation methods for decentralized control of multiaccess communications. IEEE Transactions on Information Theory IT-SI, 2 (Mar. 1985).

[10]. Jain, R., Ramakrishnan, K., and Chiu, D.-M. Congestion avoidance in computer networks with a connectionless network layer. Tech. Rep. DEC-TR-506, Digital Equipment Corporation, Aug. 1987.

[11]. Aldous, D. J. Ultimate instability of exponential back-off protocol for acknowledg-ment based transmission control of random access communication channels. IEEE Transactions on Information Theory IT-33, 2 (Mar. 1987).

[12]. Nagle, J. Congestion Control in IP/TCP Internetworks. Arpanet Working Group Requests for Comment, DDN Network Information Center, SRI International, Menlo Park, CA, Jan. 1984. RFC-896.

[13]. Feller, W. Probability Theory and its Applications, second ed., vol. II. John Wiley & Sons, 1971.

[14]. Ljung, L., and Soderstrom, T. Theory and Practice of Recursive Identification. MIT Press, 1983.

BÖLÜM 3

GÖZDEN GEÇİRİLMİŞ VEGAS

TCP VEGAS ın yaratıcı teknikleri son yıllarda birçok tartışmanın konusu olmuştur. Birçok çalışma TCP VEGAS ın TCP RENO ya kıyasla daha iyi performans sağladığını ortaya koymuştur. Ancak bu iki teknikten hangisinin performans kazanımlarından tamamen sorumlu olduğu henüz bilinmemektedir. Bu çalışma TCP VEGAS ın ayrıntılı bir performans değerlendirmesini sunmaktadır. TCP VEGAS ı çeşitli yeni mekanizmalara ayrıştırarak ve bu mekanizmaların her birinin performansını değerlendirerek şunu gösterdik: performans kazanımı esas olarak, TCP VEGAS ın yavaş başlangıç ve tıkanıklık giderilmesi için olan yeni teknikleri sayesinde başarılmaktadır. TCP VEGAS ın yaratıcı tıkanıklıktan kaçınma mekanizmasının ise sonuç üzerinde çok az bir etkisi olduğunu gösterdik. Dahası, bu mekanizmanın bazı problemler sergilediğini bulduk.

3.1. Giriş

TCP VEGAS TCP için ilk kez Brakmo tarafından sunulan yeni bir tasarımıdır. TCP VEGAS geliştirilmiş bir yeniden iletim stratejisini içerir. Bu strateji iyi parçalanmış tur ölçümlerine ve yavaş-başlangıç ve tıkanıklıktan kaçınma esnasında tıkanıklık tespiti için oluşturulan yeni mekanizmalara dayanır. Yaratıcı teknikler ve etkileyici performans kazanımları son yıllarda birçok tartışmanın konusu olmuştur. Bu çalışma TCP VEGAS ın tasarımına taze bir bakış sunmakta ve TCP VEGAS ın yeniliklerinin avantajlarına ışık tutmaya çalışmaktadır.

TCP RENO nun tıkanıklık tespiti ve kontrol mekanizmaları parçaların kaybını bir sinyal olarak kullanmaktadır. Bu parça kayıpları şebekede tıkanıklık olduğunu göstermektedir. Bu yüzden TCP RENO nun kayıplar olmadan önce tıkanıklığın başlangıç aşamalarını tespit edecek bir mekanizması yoktur. Dolayısıyla TCP RENO kayıpları engelleyemez. Dahası, TCP RENO reaktiftir, yani bağlantının mevcut band genişliğini bulmak için kayıplar üretmeye ihtiyacı vardır. Öbür taraftan, TCP VEGAS ın tıkanıklık tespit mekanizması aktiftir, yani çıktı oranındaki değişiklikleri gözlemleyerek tıkanıklıktaki başlangıcı tespit etmeye çalışır. TCP VEGAS bu çıktı ölçümlerinden tıkanıklık penceresi ayarlama politikasını çıkarır, bu da bağlantı kayıplar vermeden önce gönderme oranını azaltabilmeyi sağlar.

TCP VEGAS çeşitli değişik tekniklerin bir birleşimidir. Her bir teknik kendi başına bir tartışma konusudur. Daha önce yapılan tartışma ve çalışmalar ya yalnız belli bir mekanizma üzerinde yoğunlaşmış ya da TCP VEGAS ın bütün olarak davranışını değerlendirmeye çalışmıştır. Ancak asıl soru TCP VEGAS içerisindeki hangi tekniğin performans kazanımlarından sorumlu olduğudur. Bu soru şu ana kadar cevapsız kalmıştır. Bu soruyu cevaplandırmak için TCP VEGAS ı kendi algoritmalarına ayırdık ve bu algoritmaların her birinin performans üzerindeki etkileri değerlendirilmiştir.

3.2. TCP VEGAS

Daha önce yayınlanan ve TCP VEGAS ı tanımlayan çalışmalara göre TCP VEGAS TCP RENO dan şu sebeplerle farklıdır: TCP VEGAS yeniden transmisyon stratejisini etkileyen 3 yenilik getirmektedir. Birincisi, TCP VEGAS gönderilen her bir parça için RTT yi ölçer. Ölçümler iyi parçalanmış saat değerlerine dayanır. Bu ölçümler kullanılarak her bir parça için bir mola periyodu ölçülür. Kopya bir onay (ACK) alındığında TCP VEGAS mola süresinin bitip bitmediğini kontrol eder. Mola süresi bittiyse, parça yeniden iletilir. İkinci olarak, kopya olmayan bir ACK alınırsa ki, bu hızlı bir yeniden iletimden sonraki ilk veya ikincidir, TCP VEGAS zamanlayıcının bitip bitmediğini kontrol eder ve başka bir parçayı yeniden iletebilir. Üçüncüsü, birden fazla parça kaybı olduğunda ve birden fazla hızlı yeniden iletim gerçekleştiğinde, tıkanıklık penceresi sadece ilk hızlı yeniden iletim için azaltılır.
Tıkanıklık kaçınma mekanizması: TCP VEGAS tıkanıklıktan kaçınma esnasında tıkanıklık penceresini devamlı olarak artırmaz. Bunun yerine, başlangıç tıkanıklığını tespit etmeye çalışır. Bunu ölçülen çıktı ile beklenen çıktıyı karşılaştırarak yapar. Tıkanıklık penceresi bu iki değer ancak birbirine çok yakın olursa artırılır. Bu şu anlama gelir: eğer yeterli şebeke kapasitesi varsa beklenen sonuç, çıktı başarılabilir. Tıkanıklık penceresi, ölçülen çıktı beklenen çıktıdan az ise azaltılır; bu durum başlangıç tıkanıklığı için bir işaret olarak kabul edilir.
Geliştirilmiş yavaş-başlangıç mekanizması: Benzer bir tıkanıklık tespit mekanizması yavaş başlangıç sırasında uygulanır. Bunun amacı tıkanıklık kaçınma aşamasına ne zaman geçileceğine karar vermek içindir. Beklenen ve gerçekleşen çıktının sağlıklı bir karşılaştırmasını yapabilmek için tıkanıklık penceresinin sadece her bir diğer RTT kadar büyümesine izin verilir.

Kaynak [1] de, ek bir algoritma sunulmaktadır. Bu algoritma mevcut bant genişliğini yavaş başlangıç sırasında ACK aralığından alamaya çalışır. Ancak, bu algoritma deneyseldir ve TCP VEGAS ın değerlendirmesinde kullanılmamıştır. Dolayısıyla bizde bunu değerlendirmemizden çıkardık.

Hem kaynak [1] hem de [2] internette TCP VEGAS için %37 ile %71 arasında daha iyi çıktı rapor ediyorlar. Kayıplar ise beşte bir ile yarısı kadardır. Simülasyonlar bu ölçümleri teyit etmektedir, ayrıca şunu göstermektedir: VEGAS TCP RENO nun çıktısını kötü etkilememekte ve TCP VEGAS TCP RENO dan daha az doğru değildir.

3.3. Kaynak Taraması

TCP VEGAS ın tıkanıklık kaçınma konusundaki yeni teknikleri, bunun TCP performansı üzerindeki etkileri, rakip TCP RENO bağlantılarının varlığında TCP VEGAS ın davranışı daha önce araştırmacılar tarafından araştırılmış konulardır. Şimdi daha önce yapılan bu çalışmaları kısaca gözden geçirelim:

Ahn kaynak [3] de TCP VEGAS ile bazı canlı internet deneyleri gerçekleştirdi. Bir TCP RENO alıcısına % 20 daha hızlı ve bir TCP Tahoe alıcısına % 300 daha hızlı transfer gerçekleştiğini rapor ediyor. Her iki senaryo içinde TCP VEGAS ın daha az parçayı yeniden ilettiği, daha düşük RTT ortalaması ve varyansına sahip olduğu bulundu. WAN' da yapılan deneyler şunu ortaya çıkardı ki, TCP VEGAS yüksek tıkanıklık durumlarında daha yüksek çıktı verirken, TCP RENO TCP VEGAS ı düşük tıkanıklık durumunda geride bıraktı.

Akıcı bir model ve benzetimler ile MO kaynak [4] TCP VEGAS ın TCP RENO nun tersine uzun gecikmeli bağlantılara karşı eğilimli olmadığını ve TCP VEGAS ın TCP RENO nun varlığında doğru oranda bant genişliği alamadığını göstermiştir.

Hasegawa kaynak [5] TCP VEGAS ın tıkanıklık kaçınma mekanizmasının TCP RENO dan daha sağlam olduğunu bulan analitik bir model kullanmıştır. TCP VEGAS bağlantısının tıkanıklık penceresi sabit bir değere dönüşebilir. Ancak şunu da bulmuştur: bu mekanizma bazen değişik dolanım hızına sahip birçok bağlantıyı doğru bir şekilde sağlamada başarısız olur.

Bir Wan emülatörünün veya bir uydu bağlantısının yardımıyla, Zhang [6] uzun gecikme bağlantıları üzerinde çeşitli TCP versiyonlarının performansını araştırdı. TCP VEGAS TCP Tahoe ve TCP RENO nun ancak yarısı kadar çıktı iletti ancak diğer TCP lere oranla çok daha az yeniden iletim yaptı.

Ahn kaynak [7] yüksek hızlı geniş alan paket şebekelerin benzetimini hızlandırmak için yeni bir teknik geliştirdi. Değerlendirme bölümü TCP VEGAS ın küçültülmüş bir versiyonunun sonuçlarını sunmaktadır. Bu tıkanıklık tespiti ve pencere ayarlama tasarısını gigabit şebeke üzerinde göstermektedir. Deneylerde, TCP VEGAS TCP RENO nun ancak yarısı kadar çıktı elde etmektedir.

TCP VEGAS ın bu kadar sınırlanmış bir versiyonu Bolliger kaynak [8] tarafından da değerlendirilmiştir. TCP nin çeşitli versiyonları kullanıcı seviyesi protokolleri olarak uygulanmış ve internette değerlendirilmiştir. TCP VEGAS ın TCP RENO ya oranla birden fazla parça kaybından dolayı daha az molaya neden olduğu gösterilmiştir. Diğer taraftan, TCP VEGAS TCP RENO ya oranla daha fazla tetiklenmemiş molaya neden olmaktadır. Tetiklenmemiş molalar iyileşmeye girmek için kaçırılan fırsatları yansıtır. Bu çalışmada, TCP VEGAS ın çıktısı TCP RENO ya oranla çok az daha kötüdür.

Danzig, VEGAS ın yeni tıkanıklık kaçınma mekanizmasını içermeyen daha önce yayınlanmış bir versiyonunu değerlendirdi. Kaynak [2] deki iddiaları yeniden üretemeyecekleri için yazarlar TCP VEGAS ın yeni tıkanıklık kaçınma mekanizmasının performans gelişiminde önemli rolü olduğuna kanaat getirmişlerdir.

Son 3 çalışma biraz çelişkilidir, kaynak [7] ve [8] şu sonuca varmamıza neden olabilir: Tıkanıklıktan kaçınma esnasında TCP VEGAS ın yeni davranışının çıktı üzerinde olumsuz bir etkisi vardır. Fakat olumlu bir etkisi olacağını öngörmektedir. Maalesef, TCP VEGAS ın çıktı gelişimini gösteren çalışmalar rapor edilen hız artımlarından TCP VEGASın hangi algoritmalarının sorumlu olduğunu gösterememiştir. Bu çalışma bu soruya benzetimlere dayanarak yanıt aramaktadır. Ayrıntılı bir değerlendirmeye geçmeden önce sonraki bölüm benzetim ortamını tanıtmakta ve TCP VEGAS ve TCP RENO için bir başlangıç performans değerlendirmesi sunmaktadır.

3.4. Benzetim Ortamı

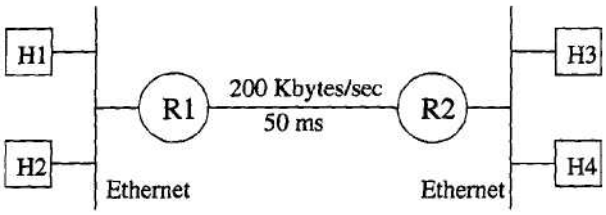
Bu bölüm TCP VEGAS ın yeni algoritmalarının etkisini incelemek maksadıyla kullanılan benzetim ortamını tanımlamaktadır.

3.4.1. Benzetim

Benzetimimizi *x-sim* adındaki, x-çekirdek tabanlı bir şebeke benzetimi üzerinde yaptık. Bu ortamda, gerçek *x*-çekirdek protokol uygulamaları benzetilmiş bir şebeke üzerinde çalıştırılır. Bizim x-sim seçimimiz şu iki gözleme dayanmaktadır: Birincisi, TCP VEGAS ı tanımlayan orijinal çalışmalardaki değerlendirmeler de x-sim ile yapılmıştır. Bu gerçek bizi TCP VEGAS ın farklı bir uygulamasını kullanarak yanlış bir yargıya varmama konusunda

güven sağlamaktadır. İkincisi, TCP VEGAS ın üretim koduna dayanan bir uygulamasını değerlendirmek istedik. Bu gereksinim x-sim tarafından sağlanmaktadır; çünkü bunun TCP VEGAS uygulaması doğrudan TCP RENO nun BSD uygulamasından alınmıştır.

TCP RENO ve *x*-çekirdekteki TCP VEGAS ın ilk uygulamalarında 2 değişiklik yaptık. Bu iki değişiklik TCP VEGAS ın yaratıcıları kaynak [9] tarafından bir çalışmada önerilmiş ve mevcut TCP RENO nun FreeBSD ve NetBSD sine uygulanmıştır. Bu değişikliklerden bir tanesi algoritmadaki bir modifikasyondur. Bu algoritma yeniden iletim mola değerini ölçer. Diğeri ise bir kontrolün tamiridir. Bu kontrol tıkanıklık penceresini hızlı iyileştirmeden sonra azaltmak içindir.



Şekil 3.1. Benzetim için ağ topolojisi.

3.4.2. Topoloji

Deneylerimiz için şekil 2.1 de sunulan topolojiyi kullandık. Daha önce yapılan çalışmalarla kıyaslanabilir olması için ilk VEGAS çalışmasındaki topolojinin tamamen aynısını seçtik. Aynı sebepten dolayı, kullanılan parça büyüklüğü 1.4 KB dır. Rotalayıcı kuyruk büyüklüğü 10 parçadır, rotalayıcı kuyruk disiplini FIFO dur. Ancak, tıkanıklıkla ilgili olmayan etkileri engellemek için daha büyük gönderici ve alıcı tampon büyüklükleri seçtik. (örneğin 50 KB yerine 128 KB) TCP alıcıları geciktirilmiş onayları kullanmamaktadır; TCP VEGAS ın tıkanıklık tespit mekanizması RTT deki değişimlere reaksiyon gösterdiği için geciktirilmiş onaylar performansı çok kötü etkileyebilir kaynak [10] de görüldüğü gibi. Kaynak [1] ve [2] deki bazı deneyleri tekrar ederek şebeke topolojisi ve benzetimi doğruladık. Bizim TCP RENO versiyonumuz orijinal versiyondan çok az daha kötü performans sergiledi. Bu farkın sebebi mola değerinin RTO daha muhafazakâr hesaplanmasıydı. Kaynak [11] de önerildiği gibi daha önceki versiyonda 2 ile çarpılan değer burada 4 ile çarpıldı.

3.5. Performans Değerlendirmesi

TCP VEGAS ın performansını kavramak için 1MB lık bir verinin sunucu HI dan sunucu H3 e transferini benzeşimledir. Bu farklı dereceler ve ters trafik tipleri için yapıldı. H2 den H4 e akan ters trafik TRAFFIC adı verilen, internet trafiğini benzetim eden ve Toplib kaynak [12] tabanlı bir x-çekirdek protokolüdür. Her tip deney 50 kez çalıştırıldı. Tablo 3.1 ve 3.2 bu deneylerin sonuçlarını göstermektedir. Düşük ters trafik durumunda ara varış zamanı 0.1 sn yüksek ters trafik durumunda ise 0.03 sn olmuştur. Çıktı dikkate alındığında TCP VEGAS TCP RENO dan her 4 senaryoda da daha iyi performans (% 40 dan % 120 ye kadar) göstermiştir. Dahası, TCP VEGAS TCP RENO dan % 6 ile % 65 daha az yeniden iletim yapmıştır.

Bu sonuçlar, TCP VEGAS ın kısmi olarak etkileyici gelişmeler sağladığı ve daha az yeniden iletim sonucunu verdiği yönünde rapor veren diğer çalışmaları doğrulamaktadır. Tablo 3.1 ve 3.2 bizim için daha ayrıntılı bir değerlendirme yapmamızı sağlayacak ve TCP VEGAS ın içindeki ayrı algoritmaların uygulamaları hakkında başlangıç noktası olacaktır.

AVERAGE THROUGHPUT [KB/S] PER CONNECTION.

	Background traffic			
	low		high	
	Reno	Vegas	Reno	Vegas
Reno	73.4	72.4	16.1	13.3
Vegas	105.5	101.4	35.1	29.2

Tablo 3.1.

AVERAGE RETRANSMISSIONS [KB] PER CONNECTION

	Background traffic			
	low		high	
	Reno	Vegas	Reno	Vegas
Reno	48.6	49.3	122.7	140.5
Vegas	16.8	18.8	113.0	131.9

Tablo 3.2.

3.6. TCP VEGAS' ın Algoritmaları

Algoritmaların değerlendirmesi için kaynak [13] daki kopyalar ile 2^k faktöriyel tasarımı yaklaşımını kullandık. Bu yöntem arayışı bize her birinin 2 seviyesi olan k faktörlerinin etkisine karar verme imkânı sağladı. TCP VEGAS da bu faktörler değişik algoritmalardır. Faktör seviyeleri açık ve kapalı durumlarıdır. Bunlar TCP VEGAS algoritmasının Kapalı olma durumunda TCP RENO nun kullanılabilmesi için kullanılıp kullanılmadığını gösterir.

Bir 2^k faktöriyel tasarımı her bir faktörün bağımsız olarak açılıp kapanabilmesini gerektirir. Bu yüzden, ilk önce TCP VEGAS ın kaynak kodunu güncellemek zorundaydık. Bunun amacı çeşitli algoritmaları birbirinden ayırmak ve her bir algoritmanın ayrı olarak seçilebilmesini sağlamaktır. Bu değişiklikler kaynak kodunun yakın incelemesini gerektirdi. Bu inceleme şunu ortaya çıkardı: TCP VEGAS kaynak [1] ve [2] de bahsedilen değişikliklerden biraz daha fazla değişiklik ihtiva etmektedir. TCP VEGAS daki yeni algoritmaların tam listesi aşağıda sunulmuştur.

A. Yavaş başlangıç sırasında tıkanıklık tespiti
B. Tıkanıklıktan kaçınma esnasında tıkanıklık tespiti
C. Daha saldırgan hızlı yeniden iletim mekanizması
D. Kopya olmayan ACK lar için ek yeniden iletimler
E. Çok fazla parça kaybı durumunda tıkanıklık penceresinin azalmasının önlenmesi
F. Bir iyileştirmeden sonra tıkanıklık penceresinin sadece 114 ile azalması.
G. Başlatma sırasında ve moladan sonra iki parça büyüklüğünde bir tıkanıklık penceresi (TCP RENO bu durumlarda tıkanıklık penceresinin büyüklüğünü bir parça olarak ayarlar).
H. Patlamadan kaçınma bir seferde 3 parçaya gönderilebilecek parça sayısını sınırlar.
I. Eğer gönderici uyum sağlayamazsa tıkanıklık penceresi artırılmaz, yani tıkanıklık penceresinin büyüklüğü ile yarım kalan veri miktarı arasındaki fark 2 en fazla büyüklükteki parçadan daha büyüktür.
J. Sivri sindirme çıktı oranını en fazla mevcut oranın 2 katı kadar sınırlar (bu algoritma normal olarak kapalıdır).

Algoritmaları birbirinden ayırırken, gerekli kod değişikliklerini minimum seviyede tuttuk. Bunun sebebi TCP VEGAS ın bizim uygulamamız ve ilki arasında davranışsal farklılıklardan kaçınmak içindir. Uygulamamızda şunu doğruladık: bizim TCP VEGAS versiyonumuz bütün algoritmaları kapatıldığında TCP RENO uygulamasıyla aynı sonuçları üretti. Benzer şekilde, TCP VEGAS versiyonumuz bütün algoritmaları açık durumda orijinal TCP VEGAS uygulamasıyla benzer sonuçları verdi.

3.7. Tarifnameden Sapmalar

Bölüm 3.6 TCP RENO ile ilgili daha önce tanımlanmamış değişiklikleri listeledi. Buna ilave olarak, TCP VEGAS ın kaynak koduyla ilgili incelememiz ve değerlendirmemiz bazı senaryoları ortaya çıkardı. Bu senaryolarda TCP VEGAS uygulaması, düşünülen şeyi veya orijinal çalışmalarda tanımlanan şeyleri pek başaramamaktadır.

3.7.1. Mola Davranışı

Yavaş başlangıçta ve tıkanıklıktan kaçınmada, TCP VEGAS tıkanıklık penceresinin güncellenmesi için gerekli stratejiyi değiştirip değiştirmeyeceğini belirlemek için her bir RTT yi 1 kez kontrol eder. Yavaş başlangıçta, tıkanıklık penceresinin üslü açılımını bırakıp bırakmadığını ve tıkanıklıktan kaçınmaya geçip geçmediğini kontrol eder. Tıkanıklıktan kaçınmada, tıkanıklık penceresinin lineer olarak artırılıp artırılmamasını, bir sonraki RTT esnasında sabit tutulup tutulmaması veya bir parça tarafından hemen azaltılıp azaltılmamasını kontrol eder. Tıkanıklıktan kaçınma esnasındaki bir mola durumunda, TCP VEGAS ın daha önce yayımlanan sürümü hemen üslü açılıma düşememekte, bunun yerine pencere sadece lineer olarak açılmaktadır. En kötü durumda, bu muhafazakâr açılış molanın artık onaylanmasından önce bütün veri gönderilene kadar muhtemelen tüm RTT ler için geçerli olur. Bir mola durumunda tıkanıklık penceresini güncellemesi için gerekli olan stratejisini hemen değiştirmesi için TCP VEGAS ı değiştirdik.

3.7.2. Taban RTT lerin Yeniden Ayarlanması

Yukarıda bahsedilen kontrol icra edilirken TCP VEGAS taban RTT yi son RTT esnasında sadece bir parça iletilmişse yeniden ayarlar. Bu yeniden başlama yardımıyla, TCP VEGAS rotalama değişiklikleriyle başa çıkabilir ki bu rotalama değişiklikleri minimum RTT yi artırır. TCP VEGAS tıkanıklık penceresi için minimum büyüklükte iki parça görevlendirdiği için, bu yeniden başlama göndericinin uyum sağlayamaması veya gönderecek veri olmaması durumunda ancak tetiklenir.

Ender durumlarda, bu yeniden başlama taban RTT sinin çok küçük bir değere ayarlanması neticesini verebilir. Bu numara mevcut şebeke durumlarıyla alakasız olabilir. Simülasyonumuzda hiç rotalama değişikliği olmadığı için ve göndericimizin her zaman gönderecek verisi olduğu için kodun parçasını etkisiz kıldık ve değerlendirmelerimiz için taban RTT yi yeniden başlattık.

SCENARIO FOR VIOLATED INVARIANTS.

Event	Eq.3	Eq.4	*beg_seq*	*snd_nxt*	*snd_una*
			5	10	5
timeout			5	5	5
send 5&6			5	7	5
9 is acked	−2	2	7	10	10
send 10-12			7	13	10
10 is acked	3	6	13	13	11

Tablo 3.3.

3.7.3. Sabit Bozulması

Tıkanıklıktan kaçınmada, TCP VEGAS ın tıkanıklık tespit tasarısı her bir RTTT yi kontrol eder. Kontrolün konusu şebeke şartlarının tıkanıklık penceresi ayarlama politikasında bir değişikliği öngörecek kadar değişip değişmediğidir. Tıkanıklık penceresinin nasıl

ayarlanması gerektiğine karar vermek için, TCP VEGAS beklenen çıktı ile ölçülen gerçek çıktıyı karşılaştırır. Beklenen çıktı şu şekilde hesaplanır:

$$\exp ected = \frac{windowSize}{baseRTT} \tag{3.1}$$

Pencere büyüklüğü halen iletim halindeki byte ların sayısıdır. Gerçek çıktı ise şu şekilde hesaplanır:

$$actual = \frac{rttLen}{rtt} \tag{3.2}$$

rttlen son RTT esnasında iletilen byte ların sayısını yansıtır. *rtt* ise son RTT esnasında kabul edilen parçaların ortalama RTT sidir.
Daha önce yayımlanan TCP VEGAS uygulamasında. Pencere büyüklüğü şu şekilde hesaplanmıştı:

$$snd.nxt - snd.una + \min(\max seg - acked, 0).^{6} \tag{3.3}$$

Maxseg en fazla parça büyüklüğü, cevaplanmış son ACK tarafından kabul edilen byte sayısıdır. Eşitlik (3.2) ile numaralandırılan *rttlen*, şu şekilde hesaplanır:

$$snd.nxt - beg.seq \tag{3.4}$$

begseq bir önceki gerçek ve beklenen hesaplama esnasındaki *sndnxt* nin değeridir. Begseq in onaylanması gerçek ve beklenenin hesaplanmasını tetikler. Eşitlik (3.6) ve (3.8) e göre aşağıdaki sabitler tutmalıdır:

$$\exp ected \geq actual \tag{3.5}$$

Tablo 3.3 bu sabitin tek bir kayıptan dolayı oluşacak bir mola durumunda nasıl ihlal edilebileceğini göstermektedir. Tablo 3.3 bu sabitin ihlaline sebebiyet veren olayların zaman serisini göstermektedir. Her bir olay için her bir olay işlendikten sonra *snd.nxt*, *snd.una* ve begseq in değerleri gösterilmektedir. Sütunların sıralaması (soldan sağa) değişkenlerin güncellendiği ve denklemlerin hesaplandığı sıraya benzer şekildedir. Moladan sonra ulaşan iki ACK gerçek ve beklenenin yeniden hesaplanmasını tetikler. Her iki durumda da sabit ihlal edilir yani beklenen gerçekten küçüktür.

İlk ihlal birden fazla parçayı onaylayan büyük bir ACK nın sonucudur. Bu problemi gidermek için çalışmamızda kullandığımız TCP VEGAS uygulamasındaki eşitlik (3.3) teki son terimi kaldırdık. İkinci problemin sebebi birden fazla RTT önce gönderilen veri üzerinden gerçek band genişliği hesaplanmasıydı. Begseq i bir moladan önce gönderilen verinin ACK tarafından onaylanması durumunda yeniden başlatarak çözdük. Bu şekilde, gerçek band genişliğinin hesabı moladan önce gönderilen veriyi içermeyecektir.

3.7.4. Tartışma

Bu tamirler TCP VEGAS ın performansını nasıl etkiler. Birincisi 7-A da belirtilen çözüm daha iyi performansa neden olabilir. Çünkü tıkanıklık penceresinin yavaş başlangıçta

uygun şekilde açılmasına izin verir. Çünkü eğer gönderici yanlışlıkla tıkanıklık penceresini tıkanıklıktan kaçınma stratejisine göre ayarlamaya devam etme durumundan daha hızlıdır bu.

İkincisi, tıkanıklıktan kaçınmada eğer tıkanıklık penceresi açılacaksa şu şart tutmalıdır (a pozitif ve genellikle 1 e ayarlanır).

$$(\exp ected - actual) * baseRTT < \alpha \tag{3.6}$$

Eğer TCP VEGAS ın değişmezi ihlal edilirse beklenen ve gerçek arasındaki fark negatiftir. Eşitlik (3.6) tutar ve tıkanıklık penceresi yanlışlıkla artırılabilir. Bu hareket düşünülenden daha girişken bir pencere açılımına sebep olabilir. Bu yüzden, ihlal problemini çözerek TCP VEGAS ın daha az girişken olmasını bekliyoruz.

Tablo 3.4 ve 3.5 tablo 3.1 ve 3.2 nin sonuçlarını tekrar ediyor ve ilave olarak bahsedilen tamirler yapılmış olan TCP VEGAS versiyonunun sonuçlarını da göstermektedir. Dikkat edilirse TCP VEGAS ve TCP RENO deneylerinde Tablo 3.4 ve 3.5 in 1. ve 3'üncü sıraları TCP VEGAS ı ters trafik olarak kullanılmıştır. TCP RENO deney sonuçlarının Tablo 1 ve 2 de sunulanlardan farklı olmasının sebebi budur. TCP VEGAS deneyleri için tablo 3.4 ve 3.5 in 2. sırası geliştirilmemiş, TCP VEGAS ileri ve ters trafik için kullanılmıştır.

Tamirler tüm 4 durum için az daha az çıktı, 4 durumdan 3 ünde de az daha fazla yeniden iletim sonucunu vermiştir. Hepsinden öte, TCP VEGAS* orijinal versiyonuna kıyasla benzer bir performansı başarmıştır. Çalışmanın geri kalanında TCP VEGAS ifadesi TCP VEGAS* yerine kullanılacaktır.

3.8.Çeşitli Algoritmaların Etkileri

3.8.1. Karmaşıklığın Azaltılması

Bölüm 3.6 da özetlendiği gibi, TCP VEGAS TCP RENO üzerinde 10 ilave algoritma kullanır. Tam bir $2^k r$ faktöriyel tasarımı şunları gerektirir: k=10 algoritmasının her bir mümkün birleşimi seçilmelidir ve bölüm 3.5 tanımlanan deney belirli bir ayarlama için r defa çalıştırılmalıdır kaynak [13]. Bu yöntem bize şunu sağlar: her bir algoritmanın etkisini ve algoritmaların tüm muhtemel etkileşimlerinin etkilerinin ölçmeyi sağlar. k=10 algoritmaları için deney $2^{10} - 1$ mümkün etkilerle sonuçlanır, çoğunluğu daha ziyade küçüktür.

AVERAGE THROUGHPUT [KB/S] PER CONNECTION.

	Background traffic			
	low		high	
	Reno	Vegas(')	Reno	Vegas(')
Reno	73.4	71.8	16.1	13.0
Vegas	105.5	101.4	35.1	29.2
Vegas'	103.7	99.6	34.6	28.4

Tablo 3.4.

AVERAGE RETRANSMISSIONS [KB] PER CONNECTION.

	Background traffic			
	low		high	
	Reno	Vegas(')	Reno	Vegas(')
Reno	48.6	49.5	122.7	144.8
Vegas	16.8	18.8	113.0	131.9
Vegas'	16.9	18.5	115.8	139.2

Tablo 3.5.

Karmaşıklığı azaltmak ve deneylerimizin açıklayıcılığını artırmak için algoritmaları etkiledikleri aşamalara (yavaş başlangıç, tıkanıklıktan kaçınma, iyileştirme vs.) göre 3 gruba kümelendirdik. $2^k r$ faktöriyel tasarımı her biri bir faktörü temsil edecek şekilde k=3 ile ayarladık. Faktör seviyelerindeki açık ve kapalı şu anlama gelir: ya belirli bir aşamayı etkileyen tüm algoritmalar on durumundadır ya da tümü kapalı durumundadır. Bu tasarım performansı $2^3 - 1$ ye etkileyen tüm muhtemel faktörleri ve faktörlerin etkileşimini azaltır. Algoritmalar şu şekilde kümelenmiştir.

Yavaş başlangıç: tıkanıklık tespiti (bölüm 3.6 da sunulan algoritma A) ve iki parça büyüklüğünde tıkanıklık penceresi (G) Tıkanıklıktan kaçınma: tıkanıklık tespiti (B)Tıkanıklık iyileştirme: daha girişken hızlı yeniden iletim stratejisi (C), yeni veri için ACK üzerinden yeniden iletim (D), tıkanıklık penceresinin 1/4 azaltılması (F), tıkanıklık penceresinin çok azaltışından kaçınma (E) patlamadan kaçınma algoritması (H), tıkanıklık penceresi artışı olmaması algoritması(I), ve sivri sindirme algoritması (J) her zaman kapalıdır.
Her bir deney 50 kez tekrarlanır. TCP VEGAS ın çıktısı üzerinde 3 aşamada ve kaynak [13] da tanımlanan yöntemi uygulayarak yeniden iletim sayıları üzerinde algoritmaların etkilerine karar verdik.

3.8.2. Çıktı İçin Sonuçlar

Z3 faktöriyel tasarımı aşağıdaki şekilde algoritmaların belirli bir birleşimi için y çıktısını ölçmemizi sağlar.

$$y = q_{mean} + q_{ss} \cdot x_{ss} + q_{ca} \cdot x_{ca} + q_{rec} \cdot x_{rec} +$$

aşama *i* deki tüm algoritmalar açık ise x_i 1 ve eğer kapalı ise –1 (*ss*: yavaş başlangıç, *ca*: tıkanıklıktan kaçınma, *rec*: iyileştirme, kurtarma), *q1* aşama *i* deki algoritmaların etkileridir ve *i* ve *J* aşamasındaki algoritmaların etkileşiminin etkisini gösterir. q_{mean} tüm deneylerin çıktısının ortalamasıdır.

THROUGHPUT [KB/S] (LOW BACKGROUND TRAFFIC).

	TCP Reno		TCP Vegas	
	Effect q	Percentage of variation	Effect q	Percentage of variation
mean	86.64		83.90	
ss	7.14	27.71	5.57	19.38
ca	2.06	2.30	2.25	3.17
rec	6.64	23.98	6.68	27.91
ss_ca	0.17[a]	0.02	0.46[a]	0.13
ss_rec	1.09	0.65	0.95	0.56
ca_rec	0.64	0.22	0.40[a]	0.10
ss_ca_rec	–0.68	0.25	–0.61	0.24
error		44.88		48.50
90%	0.59		0.57	

Tablo 3.6.

Tablo 3.6 düşük TCP RENO ve TCP VEGAS ters trafiğinin sonuçlarını sunmaktadır. Tablo 3.6 tüm $2^k r$=400 deneyleri için ortalama çıktıları rapor etmekte ve tüm faktörlerin *q* etkilerini ve ortalama çıktı üzerindeki etkileşimlerini rapor etmektedir. *Y* çıktısı için ortalamayı hesaplayabiliriz: örneğin, tüm algoritmaların yavaş başlangıçta, tıkanıklıktan kaçınmanın açık olduğu, kurtarma deki tüm algoritmaların kapalı olduğu ve ters trafik için TCP RENO' nun kullanıldığı bir yapılandırma için aşağıdaki şekilde hesaplanabilir:

$$q_{ss\_ca} \cdot x_{ss} x_{ca} + \cdots +$$
$$q_{ss\_ca\_rec} \cdot x_{ss} x_{ca} x_{rec}$$

Tablo 3.6 deki değişim yüzdesi kolonları y çıktısının değişiminin ne kadarının q etkisiyle açıklanabileceğini gösterir, dolayısıyla bir faktörün önemi için bir ölçüdür. Ölçümler 50 kez tekrarlandığından dolayı, deneysel hatalara atfedebileceğimiz toplam değişim yüzdesi

belirlenebilir. Hata satırı bu değişimi rapor eder. Dahası, % 90 satırında verilen değer ortalama çıktı için % 90 güven aralığında hesaba ve her bir etkinin hesabına imkân verir. O ihtiva eden güven aralıkları belirli bir faktörün veya faktör birleşiminin istatistiksel olarak belirgin olmadığı anlamına gelir.

Tablo 3.6 dan şu sonucu çıkardık: düşük TCP RENO ters trafiği için, sonuç üzerinde en büyük etkiye TCP VEGAS ın yavaş başlangıçtaki yeni algoritmaları sahiptir, daha sonra kurtarmadakiler gelir. Tıkanıklıktan kaçınma esnasında TCP VEGAS ın tıkanıklık tespit mekanizması değişimin yalnızca % 2 sinden sorumludur. Aşamalar arasındaki etkileşimler çıktı üzerinde yalnızca küçük bir etkiye sahiptir veya istatistiksel olarak önemli değildir. Çıktıdaki farklılaşmanın % 45 i deneysel hatalardan dolayıdır.

Düşük TCP VEGAS ters trafiği için, değişimin % 28 i kurtarma esnasındaki geliştirilmiş algoritmalarla açıklanabilir. Bunu yavaş başlangıç esnasındaki değişimler takip eder. Değişimin % 3 ü tıkanıklıktan kaçınma esnasındaki değişikliklerle açıklanabilir. Aşamalar arasındaki etkileşim yine küçüktür veya istatistiksel olarak önemli değildir. Değişimin hemen hemen yarısı deneysel hatalardan dolayıdır.

THROUGHPUT [KB/S] (HIGH BACKGROUND TRAFFIC).

	TCP Reno		TCP Vegas	
	Effect q	Percentage of variation	Effect q	Percentage of variation
mean	26.31		20.97	
ss	0.02[a]	0.00	0.07[a]	0.00
ca	−0.77	0.41	−0.69	0.42
rec	9.65	65.22	7.96	55.78
ss_ca	0.52	0.19	0.90	0.72
ss_rec	−0.96	0.64	−0.79	0.55
ca_rec	−0.53	0.19	−0.34[a]	0.10
ss_ca_rec	0.33[a]	0.07	0.37[a]	0.12
error		33.27		42.30
90%	0.45		0.45	

Tablo 3.7.

Yüksek ters trafik senaryolarının verisi Tablo 3.7 de görülmektedir. TCP RENO ters trafik durumunda, dominant etki iyileştirme esnasındaki değiştirilmiş davranıştır. Diğer tüm etkiler ya çok küçüktür ya da istatistiksel olarak önemli değildir. Şunu not etmek gerekir ki, TCP VEGAS ın yeni tıkanıklıktan kaçınma mekanizması, performans üzerinde küçük bir negatif etkiye sahiptir. Deneysel hatalar toplam değişimin 1/3 ünü oluşturmaktadır.

Yüksek TCP VEGAS ters trafik senaryosundaki sonuçlar TCP RENO senaryosundaki sonuçlara benzerdir, şöyle ki, iyileştirme esnasındaki değişim deneyde görülen değişimin çoğunu oluşturmaktadır. Yine tıkanıklıktan kaçınma esnasındaki değiştirilmiş davranışın etkisi negatiftir.

3.8.3. Yeniden İletimlerin Sonuçları

Tablo 3.8 düşük ters trafik için yeniden iletilen veri miktarı üzerinde üç aşamanın etkisini sunmaktadır. Hem TCP VEGAS hem de TCP RENO ters trafiği için, yavaş başlangıçtaki değişimler dominanttır. Bunu tıkanıklıktan kaçınmadaki değişimler takip eder. Şunu da not etmek gerekir ki, iyileştirmedeki modifikasyonlar ve yavaş başlangıç ve tıkanıklıktan kaçınmadaki geliştirmeler arasındaki etkileşimler yeniden iletilen veri miktarını artırmıştır.

RETRANSMISSIONS [KB] (LOW BACKGROUND TRAFFIC).

	TCP Reno		TCP Vegas	
	Effect q	Percentage of variation	Effect q	Percentage of variation
mean	29.02		32.75	
ss	−10.74	43.12	−12.43	54.91
ca	−8.65	27.97	−6.38	14.48
rec	3.08	3.56	3.05	3.31
ss_ca	5.42	10.96	2.41	2.06
ss_rec	0.32[a]	0.04	0.40[a]	0.06
ca_rec	−1.98	1.47	−1.53	0.83
ss_ca_rec	0.43	0.07	0.25[a]	0.02
error		12.82		24.34
90%	0.38		0.54	

Tablo 3.8.

Yüksek ters trafik durumunda, tablo 3.8 da da görüldüğü gibi deneysel hata hem TCP VEGAS hem de TCP RENO ters trafiği için değişimin % 90 ını açıklamaktadır. Aşamaların ayrı olarak yeniden iletim sayısı üzerindeki etkisi deneysel hataya kıyasla ihmal edilebilir seviyededir. Bu sürpriz değildir çünkü TCP VEGAS TCP RENO ya kıyasla ilk sırada yüksek ters trafik durumunda tablo 3.5 deki gibi yeniden iletim sayısını azaltma konusunda başarılı gözükmemektedir.

3.8.4. Sonuçlar

3.8.4.1 Yavaş Başlangıç

Düşük ters trafik senaryolarında yavaş başlangıçtaki değişimler önemlidir, özellikle ters trafik TCP RENO ise. Paket izlerinin bir incelemesi şunu ortaya çıkarmıştır: TCP VEGAS ın tıkanıklığa duyarlı pencere güncelleme stratejisi, başlangıç yavaş-başlangıcında molalardan kaçınma konusunda başarılıdır. TCP RENO nun tıkanıklık penceresinin daha hızlı ve tepkisiz açılımı bu aşamada mevcut band genişliğine fazla yüklenmeye ve parça kayıplarına sebep olabilir. Böyle bir zarar ancak bir mola ile giderilebilir. Ters trafik düşük olduğu için, transferler kısadır. Bu yüzden, bir mola çıktıyı kötü şekilde etkiler. TCP VEGAS yavaş başlangıçtaki başlangıç tıkanıklığını hissederek, molalardan kaçınabilir ve dolayısıyla TCP RENO dan daha iyi performans gösterir. Her bir algoritmanın bir faktörü temsil ettiği (A-G) daha detaylı 27 deneyin değerlendirmesi şunu göstermektedir: yavaş başlangıçtaki tıkanıklık tespiti gerçekte tüm algoritmalar içinde en yüksek pozitif etkiye sahiptir. (% 25) yavaş başlangıçtaki ikinci değişim ise ihmal edilebilir bir etkiye sahiptir. Başlangıç yavaş-başlangıcında mevcut bant genişliğine fazla yüklenilmesi problemi diğer araştırmacılar tarafından da tanınmıştır, TCP VEGAS ın yayınlanmasından beri bu problemi işleyen bazı çalışmalar yayımlanmıştır. Yavaş-başlangıçtaki mola ihtimalini azaltarak, tıkanıklık tespiti aynı zamanda yeniden iletim sayısını da azaltmayı başarır. Deney değişimin % 50 sinin bununla açıklanabileceğini göstermektedir. İlginç olan şudur ki, değişimin %3 ünden sorumlu olan G algoritması yeniden iletim miktarını artırmaktadır. Dolayısıyla tıkanıklık penceresini iki parça halinde başlatmak fazla saldırgan olabilir.

RETRANSMISSIONS [KB] (HIGH BACKGROUND TRAFFIC).

	TCP Reno		TCP Vegas	
	Effect q	Percentage of variation	Effect q	Percentage of variation
mean	120.84		144.02	
ss	2.55	1.09	5.58	3.45
ca	−2.21	0.81	−1.84[a]	0.37
rec	−4.79	3.83	−6.31	4.41
ss_ca	1.41[a]	0.33	1.45[a]	0.23
ss_rec	−3.24	1.75	−3.22	1.14
ca_rec	0.25[a]	0.01	−0.24[a]	0.01
ss_ca_rec	0.97[a]	0.16	−0.26[a]	0.01
error		92.02		90.39
90%	1.52		1.85	

Tablo 3.9.

AVERAGE THROUGHPUT [KB/S] (WAN SCENARIO).

	Background traffic	
	Reno	Vegas
Reno	15.2	14.7
Vegas	18.6	16.4

Tablo 3.10.

Yüksek ters trafik senaryosunda, yavaş-başlangıçtaki değişimlerin neredeyse hiç etkisi yoktur. Yüksek ve ters trafik senaryolarındaki değişimlerin etkileri konusundaki bu uyumsuzluk ve çelişki daha yakından bir incelemeyi gerektirmektedir. Bu maksatla, bir WAN senaryosu için benzetimlerimizi tekrar ettik. WAN senaryosunun topolojisi Bölüm 3.4.2 de tanımlanana benzerdir. Darboğaz bağlantısının gecikmesi 400 ms, bant genişliği 1.SM bit/sn ve kuyruk yönlendiricisinin büyüklüğü 50 parçadır. Yüksek ters trafiği benzettik. Tablo 3.10 WAN senaryosunda TCP VEGAS ve TCP RENO tarafından elde edilen çıktıları göstermektedir. TCP VEGAS ın performans gelişiminin orijinal topolojiye oranla daha az (% 10-20) etkili olduğunu not ettik. Tablo 3.11 üç aşamanın etkilerini ve TCP VEGAS ın çıktısı üzerindeki etkileşimleri listelemektedir. Yavaş başlangıçtaki değişimlerin çıktıyı olumsuz etkilediğini görmek ilginçtir. Bu gözlem yüksek ters trafik durumlarında şu anlama gelir: TCP VEGAS ın yavaş başlangıçta başlangıç tıkanıklığını hissetmesi ve tıkanıklıktan kaçınmaya geçmesi fazla muhafazakârdır ve performans gelişimleri (TCP RENO ya kıyasla) iyileştirmedeki değişimlere atfedilmelidir. WAN senaryosundaki yeniden iletilen veri miktarını incelediğimizde, yavaş başlangıç değişimlerinin yeniden iletim sayısını azaltmaya da yardım etmediğini bulduk.

3.8.4.2. Kurtarma

İyileştirmedeki değişimler çıktı üzerindeki en büyük etkiye sahiptir. Yavaş başlangıç değişimlerinin biraz daha etkili olduğu düşük TCP RENO ters trafik durumu hariçtir. TCP VEGAS ın daha girişken hızlı yeniden iletim politikası (C) nın tek başına performans kazanımından sorumlu olduğundan şüphe edilebilir. Ancak, 2^7 deneyinin değerlendirmesi şunu ortaya çıkarmaktadır: yüksek ters trafik durumunda tıkanıklık penceresini sadece 114 (F) kadar azaltmak en büyük etkiye sahiptir (%28 TCP RENO ters trafiği için ve yaklaşık % 7 TCP VEGAS ters trafiği için). Bunları yeni veri için ACK lar tarafından tetiklenen yeniden iletimler takip eder (D: RENO: %9, VEGAS: %2). Daha hızlı girişken yeniden iletim

politikasının etkisi (C) daha da küçüktür (RENO %3, VEGAS %2). İyileştirmedeki tıkanıklık penceresinin çok azaltılmasından kaçınan (E) algoritmasının çıktı üzerinde hiçbir etkisi yoktur.

THROUGHPUT [KB/s] (WAN SCENARIO).

	TCP Reno		TCP Vegas	
	Effect q	Percentage of variation	Effect q	Percentage of variation
mean	17.96		16.26	
ss	−2.87	12.31	−1.54	4.61
ca	−0.19[a]	0.05	−0.63	0.78
rec	4.80	34.29	3.25	20.53
ss_ca	0.39	0.22	0.05[a]	0.00
ss_rec	−1.74	4.52	−0.90	1.59
ca_rec	0.30[a]	0.13	0.16[a]	0.05
ss_ca_rec	0.01[a]	0.00	−0.20[a]	0.08
error		48.48		72.36
90%	0.37		0.40	

Tablo 3.11.

F algoritmasının TCP VEGAS ın performansını geliştirmesinin iki sebebi vardır. Birincisi, tıkanıklık penceresini yalnızca 114 daraltmak veya yarıya bölmek yerine iyileştirmeden sonra daha geniş bir tıkanıklık penceresi sonucunu verir. İkincisi, F algoritması iyileştirme davranışını değiştirir. Tıkanıklık penceresinin ikiye bölerek, TCP RENO yaklaşık yarım RTT kadar beklemelidir. Ta ki: yeteri kadar kopya ACK ların ulaşıp tıkanıklık penceresinin mevcut duran veriden daha fazla olmasını sağlayana kadar. Diğer taraftan, TCP VEGAS bir RTT nin sadece ¼ ü kadar beklemelidir. F algoritması kayda değer çıktı gelişimi ile sonuçlansa da bunu diğer bağlantıların pahasına yapabilir. Mesela, yüksek TCP RENO ters trafikte F değişimin % 28 inden sorumludur, fakat TCP VEGAS da yalnızca % 7 sinden sorumludur. Bu sonuçlar şuna işaret etmektedir: F algoritması VEGAS ın darboğaz band genişliğinden daha geniş bir pay almasına izin vermektedir. Bu mantık şu gözlemle de desteklenir: TCP RENO genellikle TCP VEGAS ters trafiği üzerinde çalıştırılırken daha düşük çıktı verir. TCP RENO nun gerçekte TCP VEGAS ters trafiği ile yarışırken kaybettiğini ve TCP VEGAS ın TCP RENO dan band genişliği çalabildiğini bulduk. Tıkanıklık iyileştirmesindeki değişimler en büyük etkiye sahip olduğu için bu asimetri veya adaletsizlik veya yanlışlık için ana olarak sorumludurlar.

D algoritması fazla parça kaybından dolayı molalardan kaçınmaya yardım eder. Kaynak [8] de gösterildiği gibi TCP RENO daki molaların çoğunluğu fazla parça kaybından kaynaklanır. Dolayısıyla, bu molaların sayısını azaltmak için yapılan değişiklikler çok faydalı olur. TCP VEGASın hızlı yeniden iletim politikasının sonuçları Jacobson un kaynak [14] kanıtı destekler. Jacobson yeni politikanın muhtemelen sadece ihmal edilebilir bir performans kazanımıyla sonuçlanacağını iddia etmiştir. E algoritmasının çıktı üzerinde neredeyse hiçbir etkisinin bulunmaması gerçeği şunu gösterir: D algoritması tarafından düzeltilemeyen fazla parça kaybı durumlarının bir mola vermeden devam ettirilmesi çok zordur. Bunun sebebi büyük olasılıkla daha fazla hızlı yeniden iletimin tetiklenebilmesi için tıkanıklık penceresinin çok küçük olmasıdır. Özetle, TCP VEGAS ın tıkanıklık iyileştirmesi için olan tekniklerinin özellikle fazla parça kaybı ile ilgili olan D algoritmasının çok etkili olduğu görülmüş ve TCP RENO ya kıyasla etkileyici bir performans sergilemesinden ana olarak sorumlu olduğu tespit edilmiştir. Etkili olmasına rağmen, F algoritması doğruluk açısından problemli olabilir. Şunu da not etmeliyiz ki, TCP RENO nun fazla parça kaybı ile mücadelede olan problemleri diğer araştırmacılar tarafından da işaret edilmiştir. Kaynak [5], [10], [15], [16]. Yakın geçmişte, bu

problemleri çözmek için TCP RENO nun verisini ve tıkanıklık iyileştirme mekanizmasını iyileştirmek için birkaç çözüm önerisi getirilmiştir.

3.8.4.3. Tıkanıklıktan Kaçınma

Deneylerimiz gösterdi ki, TCP VEGAS ın muhtemelen en yaratıcı özelliği, yani tıkanıklıktan kaçınma esnasındaki tıkanıklık tespit mekanizması, gerçekte en az etkisi olandır. Hatta TCP ters trafik senaryosunda etkisi negatiftir. Tablo 3.12 bu sonuçları özetlemektedir. Bu tablo Tablo 3.9 daki sonuçları tekrar etmekte, ilave olarak TCP VEGAS ın bir versiyonunun çıktılarını göstermektedir. Bu sürüm, tıkanıklıktan kaçınmada tıkanıklık tespit mekanizmasını ayırmaktadır. Bu rakamlar TCP VEGAS ın tıkanıklıktan kaçınma mekanizmasının yalnızca orta seviyede etkili olduğunu anlatmaktadır. Dahası, yalnızca TCP VEGAS ın yeni tıkanıklıktan kaçınma mekanizmasını içeren bir TCP VEGAS versiyonu TCP RENO ya kıyasla düşük ters trafik yüklemelerinde çok küçük bir gelişme kaydetmiştir ve hatta yüksek ters yükleme durumlarında daha düşük çıktı sonucunu vermiştir.

TCP VEGAS ın tıkanıklıktan kaçınma mekanizması yüzünden daha iyi performans göstermesi beklenir, çünkü: TCP VEGAS paket kaybından kaçınmak için tıkanıklık penceresini aktif olarak küçük miktarlarda bir kerede bir parça kadar azaltabilir. Bir paket kaybı bir tıkanıklık penceresinde büyük miktarda azalmaya neden olur. Bu yüzden, paket kaybından sonra, küçük orandaki azalmaların çıktıyı oranın yarıya bölünmesinden daha az etkilemesi beklenir. Ancak, bu çalışma ve diğer çalışmaların sonuçları kaynak [7], [8] bu umudu kırar. Sonuçta, deneye dayalı, deneysel kanıtlar tıkanıklıktan kaçınma mekanizmasının çok muhafazakâr olduğunu göstermektedir.

Yeniden iletilen verinin miktarı üzerindeki etki dikkate alındığında TCP VEGAS ın tıkanıklık tespit mekanizmasını oldukça başarılı bulduk. Ancak, şunu da not etmek gerekir ki, hem yüksek hem de alçak ters trafik için, diğer faktörler azalmaya bu mekanizmadan daha fazla katkıda bulunmaktadır.

AVERAGE THROUGHPUT [KB/S] PER CONNECTION.

	Background traffic			
	low		high	
	Reno	Vegas	Reno	Vegas
Reno	73.4	71.8	16.1	13.0
Vegas	105.5	101.4	35.1	29.2
Vegas w/o *ca*	99.3	94.6	35.5	28.0
Vegas only *ca*	74.5	73.4	15.3	11.3

Tablo 3.12.

3.9. Tıkanıklıktan Kaçınmanın Problemleri

Sekizinci bölümde gösterildiği gibi, TCP VEGAS ın yeni tıkanıklıktan kaçınma mekanizmasının çıktı üzerindeki etkisi orta seviyededir. Bu bölüm bu mekanizmanın doğruluk problemleri de sergileyebileceğini göstermektedir.

3.9.1. Eski Bağlantıların Yanlış Ele Alınması

Tıkanıklıktan kaçınmada, TCP VEGAS aşağıdaki şart karşılandığında tıkanıklık penceresini azaltmaya başlar. P pozitiftir ve genellikle 3 e ayarlanmıştır, terimlerin tanımları için bölüm 8-c ye bakın:

$$\left(\frac{windowSize}{baseRTT} - \frac{rttLen}{rtt} \right) * baseRTT > \beta$$

pencere büyüklüğünün rttlen e eşit olduğunu farz edersek, iletim halindeki parçaların sayısı son RTT esnasında gönderilen parçaların sayısına tekabül eder. Bu varsayım TCP Vega dengeye ulaştığında geçerlidir. Yukarıdaki durum aşağıdaki için doğrudur:

$$windowSize > \frac{\beta}{1 - \frac{baseRTT}{rtt}}$$

Bir TCP VEGAS bağlantısının tıkanıklık olmayan bir şebekede başlatıldığı bir senaryoyu düşünün. Bu bağlantının baseRTT1 i minimum mümkün R1T ye oldukça yakındır. Eğer şebekede sonradan tıkanıklık olursa ölçülen RTT artar ve *baseRTT/rtti* azalır. Bizim benzetim topolojimiz için, 0.5 den küçük faktörler gözlemledik. Şimdi, ikinci bir bağlantının başlatıldığını farz edin. Şebekede tıkanıklık olduğu için, ikinci bağlantının tahmini *baseRTT2* si *baseRTT1* den daha büyük, bu yüzden base*RTTIrtt2* de *baseRTT/rtti* den daha büyüktür (*rtt1=rrr2* olduğu kabul edilerek). Bu şu anlama gelir: tıkanıklık penceresi büyüklüğünün azalmasını tetikleyen pencere büyüklüğü kritik değeri ikinci bağlantıda birinci bağlantıdan daha büyüktür. Bu yüzden, ikinci bağlantı birinci bağlantıdan daha yüksek band genişlikleri başarır.

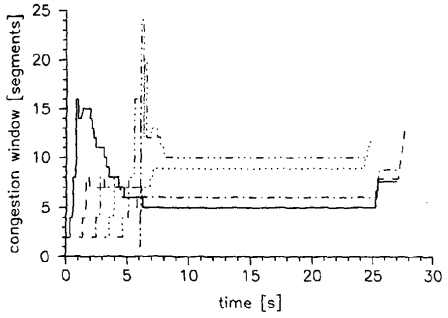


Fig. 2. Connections sharing bottleneck.

Şekil 3.2. Darboğaz paylaşımın ilişkisi.

Şekil 3.2 TCP VEGAS ın tıkanıklıktan kaçınma mekanizmasının doğruluk problemini göstermektedir. Grafik ayrı ayrı zamanlara göre düzenlenmiş 5 bağlantının tıkanıklık pencere büyüklüklerini göstermektedir. Bağlantılar bir darboğazı paylaşır ve 1 saniyelik aralıklarla başlatılır. İlk bağlantı en fazla diğer bağlantıların sebep olduğu tıkanıklığa reaksiyon gösterir ve dengede, onun tıkanıklık penceresi en küçüktür. Diğer taraftan, en son başlatılan bağlantı en büyük tıkanıklık penceresini başarır ve darboğaz band genişliğinin en geniş payını alır. İkinci bağlantının tıkanıklık penceresinin büyüklüğü dengedeki ilk bağlantınınkine benzerdir. Şunu da not etmek gerekir ki; t=3 sn deki ilk bağlantının tıkanıklık penceresi büyüklüğü, dengedeki son bağlantının tıkanıklık penceresinin büyüklüğüne tekabül eder. Ancak, ilk bağlantı t=3 sn de tıkanıklık penceresini azaltmaya zorlanırken, son bağlantı pencere büyüklüğünü ayarlamak zorunda değildir.

Tıkanıklık penceresini artırmayı tetikleyen algoritma benzer bir problem yaşar. Aşağıdaki şart sağlandığında tıkanıklık penceresi artırılır ve a genellikle 1 e ayarlanır.

$$windowSize > \frac{\alpha}{1 - \frac{baseRTT}{rtt}}$$

Bir A bağlantısı için sağ taraftaki terim daha büyük olduğundan, tıkanık bir şebekede başlatıldığından, daha sonra bir B bağlantısı için, şebeke tıkanık değilken başlatıldığından, şart B tipi bağlantılardan ziyade A tipi bağlantılar tarafından sağlanır. Yine, bu şu anlama gelir: A tipi bağlantılar adil olmayan bir bağlantı band genişliği payı elde eder.

Şunu da not etmek gerekir ki; TCP RENO nun tıkanıklıktan kaçınma stratejisi de adaleti garanti etmez. Ancak, her bağlantı zaman zaman kayıplar verdiğinden ve kendinden güdülenen paket kayıplarından ötürü, en azından diğer bağlantılar için yetişme şansı vardır. TCP VEGAS da durum böyle olmayabilir, çünkü TCP VEGAS özellikle kendinden güdülenen kayıpları engellemeye çalışır.

3.9.2. Israrlı Tıkanıklık

Bölüm 3.9.2 de tartışılan probleme ilave olarak, ısrarlı tıkanıklık durumlarında ki TCP VEGAS davranışı vardır. Böyle bir durumda, bir TCP VEGAS bağlantısı *baseRTT* yi olduğundan fazla tahmin eder ve onun a ve 8 parçaları arasında iletimde olduğunu düşündüğü halde, gerçekte, iletim halinde daha pek çok parçası vardır. Problemin detaylı bir tanımı kaynak [21] de vardır.

3.9.3. Tartışma

Bölüm 3.9.2 de tanımlanan problemi yenmek için öneriler olmuştur. Bu önerilerden biri, baseR1T yi ısrarlı tıkanıklık durumunda daha büyük bir değere ayarlamaktır örneğin kaynak [21] de olduğu gibi. Bölüm 3.9.1 daki problem benzer bir şekilde yenilebilir. *baseRTT* yi yeniden ayarlamak rotalama değişiklikleri durumunda yeterli bir ölçü olabilse de; minimum RTT nin gerçekte daha büyük olabileceği yerde, iki problemin olacağı durumlarda yeterli bir çözüm değildir. Çünkü temel RTT tanım olarak bağlantı tıkanık değilken bir parçanın RTT sidir. Dolayısıyla, *baseRTT* yi minimum ölçülen RTT den daha büyük bir değere ayarlamak bu tanımı ihlal eder ve tıkanıklıktan kaçınma mekanizmasının teorik temeline gölge düşürür.

3.10. Sonuçlar

Bizim TCP VEGAS değerlendirmemiz daha önce yapılan kaynak [2], [3] ve TCP VEGAS ın TCP RENO dan önemli derecede daha yüksek çıktıyı başarabileceğini gösteren çalışmaları doğrulamaktadır. Daha önce yapılan çalışmalara ilave olarak, TCP VEGAS üzerinde yaptığımız derin analizler, TCP VEGAS ın yaratıcıları tarafından performans konusunda önerilen TCP VEGAS ın çeşitli algoritmalarının ve mekanizmalarının etkisini belirlememizi sağladı.

Deneyimiz gösterdi ki; TCP VEGAS ın yavaş başlangıç ve sıkışıklıktan kurtarma teknikleri çıktı üzerinde en fazla etkiye sahiptir. Çünkü fazla parça kaybından dolayı oluşan molalardan kaçınabilirler. Bu yüzden, TCP VEGAS TCP RENO nun çok bilinen bir problemini halletmekte oldukça başarılı görünmektedir. Ancak, TCP VEGAS ın en yaratıcı özelliği olan tıkanıklıktan kaçınma esnasındaki tıkanıklık tespit mekanizması ya çok az etkiye veya hatta negatif etkiye bile sahiptir. Dahası, tıkanıklıktan kaçınma mekanizmasının rakip bağlantılar arasında doğrulukla ilgili problemler sergileyebildiğini bulduk. Sonuç olarak, TCP VEGAS ve TCP RENO nun bir sonuca varacak karşılaştırmalı değerlendirmesi için TCP

VEGAS ın yavaş başlangıç ve tıkanıklık iyileştirme tekniklerinin TCP RENO daki benzerleriyle karşılaştırılması gerekir. Biz tartışmayı TCP VEGAS ı geliştirenler tarafından incelenen bir senaryo ile sınırladık. Değişik senaryolar hala değişik incelemeleri gerektirebilir.

TCP gibi nakil protokolleri performans üzerindeki etkileri ve birbirleriyle etkileşimleri genellikle anlaşılamayan birçok karmaşık algoritmaları içine alır, örneğin tıkanıklık kontrolü için, veri kurtarma için. Bizim faktör analizi yaklaşımımız böyle bir protokolün çeşitli algoritmalarının etkinliği ve bu algoritmaların etkileşimi konusuna bir ışık tutmamızı sağladı. Gelecekteki protokol geliştiricileri bu tip performans analizi ve deneylerine imkân sağlamak için tasarım ayrıştırma ve uygulamada erken davranma konusunda cesaretlendirmek istiyoruz.

KAYNAKLAR

[1]. L.S. Brakmo and L.L. Peterson. TCP Vegas: End to End Congestion Avoidance on a global Intemet. IEEE Journal on Selected Areas in Communications. 13(8):1465 480,Oct 1995.

[2]. L. S. Brakmo, S. W. O'Malley, and L. Peterson. TCP Vegas: New Techniques for Congestion Detection and Avoidance. In Proc. of ACM SIGCOMM '94. pages 24-35, London, October 1994.

[3]. J. S. Ahn, P. Dan& 2. Liu, and L. Yan. Evaluation of TCP Vegas: Emulation and Experiment. In Proceedings of ACM SIGCOMM '95, pages 185-195, August 1995.

[4]. J. MO, R.J. La, V. Anantharam, and J. Walrand. Analysis and Comparison of TCP Reno and Vegas. In Pmceeditigs of INFOCOM '99. pages 1556-1563, March 1999.

[5]. G. Hasegawa, M. Murata, and H. Miyahara. Fairness and Stability of Congestion Control Mechanisms of TCP. In Proceedings of ZNFOCOM '99, pages 1329-1336, March 1999.

[6]. Y. Zhang, E. Yan, and S.K. Dao. A Measurement of TCP over long-Delay Network. In Proceedings of 6th Inti, Con$ on Telecommunication Systems, pages 498-504, March 1998.

[7]. J.S. Ahn and P.B. Danzig. Packet Network Simulation: Speedup and Accuracy Versus 'Iiming Granularity. IEEE Transactions on Networking, 4(5):743-757, October 1996.

[8]. J. Bolliger, U. Hengartner, and T. Gross. The Effectiveness of End-to-End Congestion Control Mechanisms, Technical Report 313, ETH Wrich, February 1999.

[9]. L.S. Brakmo and L.L. Peterson. Performance Problems in BSD4.4 TCP. Computer Conununicatiori Review, 25(5):69-86, Oct 1995.

[10]. S. Floyd and T. Henderson. RFC 2582: The NewReno Modification to n=p's Fast Recovery Algorithm, April 1999.

[11]. V. Jacobson. Congestion Avoidance and Control. In Proceedings of ACM SIGCOMM 88, pages 314-329, August 1988.

[12]. PB. Danzig and S. Jamin. A Library of TCP Internetwork traffic Characteristics. Technical Report 91495, Computer Science Department, USC, 1991.

[13|. R.Jain. TIie An ofComputer Systems Performalice Analysis. John Wiley & Sons, Inc., 1991.

[14|. V. Jacobson. problems with Arizona's vegas, March 1994. end2end-tf mailinglist.

[15]. S. Floyd. TCP and Successive Fast Retransmits, February 1995. ftp://ftp.ee.lbl.gov/papersifastretrans.ps,

[16]. J.C. Hoe. Improving the Start-up Behavior of a Congestion Control Scheme for TCP.

BÖLÜM 4

TCP VE SORGU YÖNETİM ALGORTİMASININ DUALİTY MODELİ

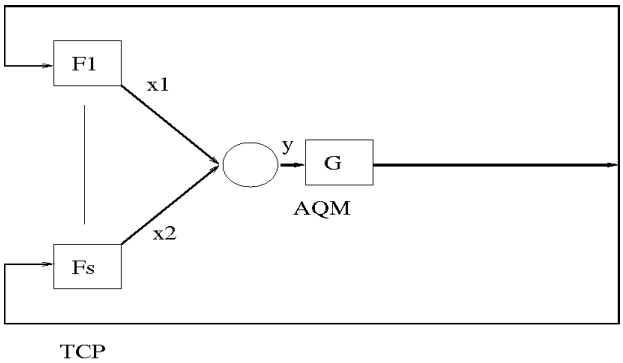
Sıkışıklık kontrolü ağ kaynaklarını paylaşmak için dağıtılmış algoritmadır. İki elemana sahiptir. İlki kendi yolundaki sıkışıklığa cevap veren dinamik olarak ayarlanan değerin kaynak algoritması (pencere boyut), ikincisi tam ve kesin olarak sıkışıklık ölçümleri ve geri gönderimleri tam veya kesin olarak kaynakların bu bağlantıyı güncelleyen bağlantı algoritmasıdır. Kaynak algoritması hali hazırdaki internette kaynak algoritması TCP tarafından taşınır. Ve bağlantı algoritmaları aktif sorgu yönetim AQM şemaları örneğin DROP TAİL veya RED tarafından taşınır. Farklı protokoller farklı ölçümler sıkışıklığı ölçmek için kullanılır. Örneğin TCP RENO kaynak [1], [2] ve bunların değişkenleri kayıp olasılığını sıkışıklık ölçümü olarak kullanır ve TCP VEGAS kaynak [3], sorgu gecikmesi sıkışıklık ölçümü olarak kullanır kaynak [4]. Her biri tam olarak bağlantıda güncellenir ve tam olarak Uçtan uca kayıp veya gecikmeyle kaynağa geri besleme yapılır. Bu yazımızda uçtan uca sıkışıklık kontrolünün genel bir modelini sunacağız ve çeşitli TCP-AQM protokolleri tarafından belirlenen kapalı döngülü sistemlerin eşitlik özelliklerini anlamak için uygulayacağız. Temel fikir sıkışıklık kontrol işlemini düşünmektir. Dağıtılmış hesaplama kaynak ve hat tarafından ağ üzerinde gerçek zamanlı olarak küresel en uyguluk problemini çözmek için kaynak [5] incelenmiştir. Buradaki elemanlar toplam kayan araç kaynak konularını kapasite değişkenlerine maksimize etmek içindir. Kaynak değerlerini birincil değişkenler olarak derleyeceğiz. Sıkışıklık ölçümlerini çift değişkenler olara ve TCP-AQM protokollerini dağıtılmış birincil çift algoritma olarak bu en uygun şekle sokma problemi çözmek için ve ilgili çift problemi çözmek için kullanacağız. Farklı protokoller örneğin RENO VEGAS RED REM gibi 1 bunların hepsi aynı ilk örnek problemi farklı araç fonksiyonları ile çözer ve bu fonksiyonları açıkça sunacağız. Daha fazla olarak bütün bu protokoller Eşitlikteki çift problemi çözmek için sıkışıklık ölçümlerini üretir (langrange çarpanı).

Bu model geniş ağın TCP-AQM kontrolü altındaki geniş ağın eşitlik özelliği için kullanılır. Örneğin gecikme sorgu uzunluğu, kayıp ihtimalleri. Aşağıda en uygun şekle sokma problemleri için kaynak [4] çalışarak anlaşılabilir. Eğer problem konkav program olursa bu özellikler nümerik etkili olarak hesaplanabilir.

Araç en üst düzeye çıkarma ile TCP-AQM algoritmaları arasında her iki yönde gidilebilir. Genel araç fonksiyonu ile başlayacağız örneğin: kendi uygulamamızda gibi ve TCP-AQM algoritmasını toplam araçları maksimize etmek için kaynak [5], [6], [7], [8], [9] olduğu gibi sunacağız. Zıt olarak TCP-AQM algoritmasını ve aşağıdaki araç fonksiyonlarını belirlemek için ters algoritma tasarlayabiliriz. Bu uçtan uca kontrol durumudur: uçtan uca sıkışıklık ölçümleri alınır alınmaz TCP algoritması birleşik bağlantı sıkışıklık ölçümlerinin toplamına ulaşır.

4.1. TCP-AQM' nin DUALİTY Modeli:

Geri dönüş sıkışıklık kontrolünde kaynaklar kendi yollarında ki cevabın sıkışıklık bilgisine göre kendi değerlerini ayarlar. Bu bir geri beslemedir ister kesin tampon artışı olsun ister tur gecikmesi olsun. Farklı şemalar farklı sıkışıklık ölçümleri benimser. Örneğin TCP RENO paket kayıpları tarafından sıkışıklığı ölçer, TCP VEGAS sorgulayarak sorgu gecikme kullanarak, RED (Rasgele Erken Belirleme) sorgu uzunluğu ile ve REM (Rasgele Logaritmik İşaretleme) performans ölçümü ile eşleri bozulan ölçümler ile ölçer (Örneğin kayıplar ve gecikmeler). RED veya REN ile bu ölçümler ister paket düşümleri olsun ister ihtimal işaretlemeleri olsun haritalanır. Bu sıkışıklık ölçümleri kontrol döngüsünü kapatarak kaynak değerlerine cevabın geri dönüşünü verir. Anahtar fikir birincil değişken olarak kaynak değerlerini düşünmektir Sıkışıklık ölçümü veya kayıp / ihtimal işaretleme eşitliği çift değişken olarak ele alınır.



Şekil 4.1.TCP-AQM nin çiftli modeli.

TCP	Duality Model	
Reno	F_s	$\dot{x}_s = \frac{1-p(t)}{\tau_s^2} - \frac{2}{3}p(t)x_s^2(t)$
	U_s	$\frac{\sqrt{3/2}}{\tau_s}\tan^{-1}\left(\frac{\tau_s x_s}{\sqrt{3/2}}\right)$
Vegas	F_s	$\dot{x}_s = \left\{ \begin{array}{ll} \frac{1}{\tau_s^2} & \text{if } x_s(t) < \overline{x}_s(t) \\ -\frac{1}{\tau_s^2} & \text{if } x_s(t) > \overline{x}_s(t) \end{array} \right\}$
	U_s	$\alpha_s d_s \log x_s$

Şekil 4.2. TCP'nin Duality modeli.

Bağlantıların seti olarak L ağa modellenmiştir. Belirli kapasiteler $c = (c_l, l \in L)$ olarak belirlenmiştir. Kaynağın C seti tarafından s ile indekslenmiş paylaşılmıştır. Her s kaynağı bağlantının $L_s \in L$ nı kullanır. L_s , $L \times S$ matrisini belirler.

$$R_{ls} = \begin{cases} 1 & eger\ l \in L_s \\ 0 & diger\ durumlarda \end{cases}$$

Her s kaynağı ile ilgili $x_s(t)$ iletim değeri t zamanında paket/saniyedir. Her l bağlantısı ile ilgili sayısal sıkışıklık ölçümü $p_l(t) \geq 0$ t zamanındadır. Kaynak [10] ün yazılışı $y_l(t) = \sum_s R_{ls} x_s(t)$ l bağlantısında toplam kaynak değeri olsun. $q_s(t) = \sum_l R_{ls} p_s(t)$ Uçtan uca kaynak için sıkışıklık ölçümü olsun. Vektör yazılışı;

$$y(t) = Rx(t) \quad veya \quad q(t) = R^T p(t)$$

burada $\Re_+^{|S|}$de $x(t) = (x_s(t), s \in S)$ ve $q(t) = (q_s(t), s \in S)$dır. Ve $\Re_+^{|S|}$de $p(t) = (p_l(t), l \in L)$ ve $y(t) = (y_l(t), l \in L)$dır. S kaynağı $x_s(t)$ kendi değerini belirler ve uçtan uca sıkışıklık ölçümü $q_s(t)$ yolunun, fakat *x(t)* veya *p(t)* vektörü değil, *q(t)* nin diğer bileşenleri değil. Benzer olarak l hattı yalnızca yerel sıkışıklık $p_l(t)$ ve akış değeri $y_l(t)$ olarak belirlenir.

$x_s(t)$ kaynak değeri her devir için F_s fonksiyonuna $x_s(t)$ ve $q_s(t)$ ye bağlı olarak ayarlanır: bütün s ler için

$$x_s(t+1) = F_s(x_s(t), q_s(t)) \tag{4.1}$$

dir

Bağlantı sıkışıklık ölçümü $p_l(t)$ $p_l(t)$ ve $y_l(t)$ ye bağlı olarak her periyotta ayarlanır. Ve bazı iç vektör değişkenleri $v_l(t)$ örneğin l bağlantısındaki sorgu uzunluğundaki gibi bu bazı fonksiyonlar ($G_s H_l$) ile modellenir. Bütün l ler için

$$p_l(t+1) = G_l(y_l(t), p_l(t), v_l(t)) \tag{4.2}$$

$$v_l(t+1) = H_l(y_l(t), p_l(t), v_l(t)) \tag{4.3}$$

G_l negatif olmayan böylelikle $p_l(t) \geq 0$ dır. Burada F_s TCP algoritmasını (RENO veya VEGAS) modeller ve ($G_s H_l$) ve AQM modeli: bir sonraki bölümü görünüz. Biz genellikle AQM yi G_l

$$x_s = F_s(x_s, q_s)$$

olarak düşünürüz. İç değişken $v_l(t)$ kesin referans olmadan ve H_l adaptasyonu olmadan eşitlik (4.1) ve (4.3) *(x,p)* varsayarız. Eşitlik (4.1) in tamamlanmış noktası x_s eşitlik değeri ile uçtan uca sıkışıklık ölçümü q_s arasında bir ilişkiyi tam olarak belirler.

$$q_s = f_s(x_s) > 0 \tag{4.4}$$

F_s devamlı değişken ve $\partial F_s / \partial_{q_s} \neq 0$ inde açık A seti: $A := \{(x_s, q_s) | x_s > 0, q_s > 0\}$ formül olduğunu varsayalım. Sonra kesin fonksiyon teoreminden tek devamlı değiştirilebilir f_s fonksiyonu $\{x_s > 0\}$ dan $\{q_s > 0\}$ dan örneğin eşitlik (4.4) gibi var olsun x_s ve q_s arasındaki tabloyu genişletmek için A nın kapanması (4.5) eşitliği ile belirlenir sonsuzdur.

$$f_s(0) = \inf\{q_s \geq 0 | F_s(0, q_s) = 0\} \tag{4.5}$$

$(x_s, 0)$ noktası = $F_s(x_s, 0) = x_s$ ise eşitlik (4.6) belirlenir.

$$f_s(x_s) = 0 \tag{4.6}$$

Her s kaynağı için araç fonksiyonu belirlenir eşitlik (4.7) deki gibi

$$U_s(x_s) = \int f_s(x_s)dx_s \qquad x_s \geq 0 \tag{4.7}$$

değişkene bağlı olarak tektir. İntegrallenebilir olması U_s nin devamlı fonksiyon olduğunu gösterir. Böylelikle $f_s(x_s) = q_s \geq 0$ bütün x_s değerleri için. U_s Azalmayandır. f_s Azalmayan fonksiyonu iddia etmek mantıklıdır. Daha sert daha fazla sıkışıklık daha düşük değerlere eşittir. Bu U_s yi konkav yapar ve eğer f_s keskin inişli ise U_s keskin, keskin konkavdır 2. türev $U_s''(x_s) < 0$ olana kadardır. Araç fonksiyonun artışı açgözlü kaynağa uygulanır. Daha büyük değerler daha yüksek araçlara yöneltir ve konkavlık azalan geri dönüşleri sağlar.

Şimdi toplam araç eşitliklerini kaynak [11] deki maksimize etmenin problemini düşünün

$$\max_{x\geq 0} \sum U_s(x_s) \;\; Rx \leq c'ne \tag{4.8}$$

Sınırlama söyler ki her l bağlantısında akım değeri y_l , c_l kapasitesini aşamaz. En uygun değer vektörü x^* eşitlik (4.8) deki nesne fonksiyonu devamlı ve uygun çözümü olduğu zaman elde edilebilir. Tektir eğer U_s keskin konkav ise. Paylaşılan bağlantı boyunca kaynak çiftli ise(kapasite sınırlaması) x^*ın çözümü direk olarak bununla birlikte mümkün olan bütün kaynakların koordinasyonunu gerektirir. Ve büyük ağlarda olanaklı değildir. Eşitlik (4.1) ve (4.3) deki anlamanın anahtarı *x(t)* yi öncelikli değişken, *p(t)* yi çiftli değişkeni ve $(F,G) = (F_s G_l, s \in S, l \in L)$ dağıtılmış öncelikli çiftli algoritma eşitlik (4.8) deki öncelikli problemi çözmek ve Langrange çiftini çözmek için düşünülür.

$$\min_{p\geq 0} \sum_s \max_{x_s \geq 0}(U_s(x_s) - x_s q_s) + \sum_l p_l c_l \tag{4.9}$$

Böylelikle çift değişken ağdaki sıkışıklığın kesin ölçümüdür. Çift problem öncelikli problem uygun olduğu zaman en uygun çözüme sahiptir. Eşitlik (4.1)-(4.3) birincil ve çiftli problemin çözümü olarak tanımlayacağız ve *(F,G)* her birincil ve çiftli değişkende beraber yaklaşacaktır her problemi çözmek için.

(F,G,H) ın tanımını özetleyebiliriz.

C1: bütün $s \in S$ ve $l \in L$, F_s ve G_l nin negatif olmayan fonksiyonlar için (4.1)-(4.3) ün eşitlik noktalarını içerir.

C2: bütün $s \in S$ F_s devamlı değişken ve $\partial F_s / \partial_{q_s} \neq 0$ ve $\{(x_s, q_s) | x_s > 0, q_s > 0\}$ f_s 4 numaralı fonksiyonda yükselmeyendir.

C3: eğer $p_l = G_l(y_l, p_l, v_l)$ ve $v_l = H_l(y_l, p_l, v_l)$ $y_l \leq c_l$ eğer $p_l > 0$ sa.

C4: her $s \in S$, f_s keskin azalan için C1 durumu $(x(t), p(t)) \geq 0$ ı ve $x^*, p^* \geq 0$ garanti eder. C2 U_s araç fonksiyonunu ve varlığını garanti eder. C3 birincil uygun tamamlayıcı (x^*, p^*) gevşekliğini garantiler. Son olarak C4 durumu x^* en uygun tekliğine garanti eder.

*Teorem1:*_C1 ve C2 deki tahminleri düşünün. (x^*, p^*) eşitlik (4.1)-(4.3) ün olsun. Daha sonra (x^*, p^*) eşitlik (4.8) deki birincil problemi çözer ve eşitlik (4.9) daki çiftli problem ile eşitlik (4.7) tarafından verilmiş olan araç fonksiyonunu da çözer. Eğer sadece C3

tutulursa. Daha fazla olarak C4 önermesini göz önünde bulundurursak Us keskin konkav olur ve en uygun değer vektörü x^* tek olur

İspat: Eşitlik (4.7) deki U_s nin tanımlamasından sonra C4 ele alındığı zaman ikinci iddia ispatlanır. Böylelikle ilk iddiayı ispatlamamız yeterlidir. DUALİTY teorisi ile Kaynak [11] *(x*,p*)* öncelikli çift en uygundur. Eğer x^* sadece öncelikli uygunsa, p^* çiftli uygunsa, tamamlayıcı sarkıklık devam ediyorsa ve

$$x^* = \arg\max_{x \geq 0} L(x, p^*) \tag{4.10}$$

8 in Lagrangian' nı L olduğu zaman

$$L(x,p) = \sum_s U_s(x_s) + \sum_l p_l (c_l - \sum_s R_{ls} x_s)$$

Böylelikle ilk iddia yı ispat etmek için (4.10) numaralı eşitliği oluşturmamız gereklidir. Şimdi

$$\begin{aligned} &\max_{x \geq 0} L(x, p^*) \\ &= \max_{x \geq 0} \sum_s U_s(x_s) + \sum_l p_l^* \left(c_l - \sum_s R_{ls} x_s \right) \\ &= \sum_s \max_{x \geq 0} \left(U_s(x_s) - x_s \sum_l R_{ls} p_l^* \right) + \sum_l p_l^* c_l \end{aligned}$$

U_s nin yapılması ile eşitlik (4.7) ve (4.4) den şunu elde ederiz ki herhangi bir eşitlik için $x_s^* > 0, (x^*, p^*)$ olduğu zaman

$$U_s'(x_s^*) = f_s(x_s^*) = q_s^* = \sum_l R_{ls} p_l^* \tag{4.11}$$

Eğer $q_s^* = 0$ eşitlik (11) eşitlik (6) ile elde edilir. Eğer $x_s^* = 0$ eşitlik (5) den elde edilirse

$$U_s'(0) = f_s(0) = q_s^* \tag{4.12}$$

fakat eşitlik (11)-(12)

$$\frac{\partial L}{\partial x_s}(x^*, p^*) \leq 0$$

'e uygulanır ki

Eğer $x_s^* > 0$ olduğu zaman. Böylelikle $L(x, p^*)$ konkavdır x de. Karush-kuhntucer durumunda gerekli ve uygundur x^* $L(x, p^*)$ $x \geq 0$ da maksimize edebilmek için. Böylelikle ispat tamamlanmış olur.

Birçok TCP-AQM protokollerinde farklı dağıtılışmış öncelikli çift algoritmalarla küresel en uygun şekle sokma eşitlik (4.8) ve eşitlik (4.9) çiftli problemlerini farklı araç fonksiyonları *Us* ile modellenebilir. Bu hesaplamada kaynaklar ve bağlantılar tarafından internet üzerinde gerçek zamanlı sıkışıklık kontrol formunda taşınır. Teorem 1 bu tip yaklaşımları kabul eden (F,G,H) geniş protokol sınıflarını karakterize eder. Bu izah uçtan uca kontrolün sonucudur. Uçtan uca sıkışıklık ölçümü alınır alınmaz TCP algoritması bağlantıların sıkışıklı ölçümlerinin toplamı gibi davranır. Bazı TCP ve AQM algoritmasındaki iddialarda (C1-C3 iddiasında gibi bölüm2 de) tipik olarak yeterlidir. *Us* araç fonksiyonun tanımı TCP algoritması sadece *Fs* ye dayalıdır. AQM nin rolü (G,H) 1-3 deki problemin tamamlayıcı gevşekliğinden emin olmak içindir ki memnun etmiştir (durum 3). Tamamlayıcı gevşeklik için basit bir yaklaşım vardır: AQM giriş değerini kapasiteye bütün şişe boynu bağlantıları için araçlanmayı maksimize etmeyi seçmek zorundadır. Herhangi bir AQM sorgu işlemlerini kararlı hale getirir. Bu özellik çift problemi çözen langrang çarpanını p^* ı üretir.

Takip eden bölümde teorem 1 i REM ile RED ile TCP ren oyu yaklaştıracağız teorem 1 uygulayarak ve TCP VEGAS DROP TAİL ile. İlk olarak (F,G,H)'ı protokol tanımlamalarından çıkaracağız ve daha sonra kesin olarak uygun hale getiren *Us* protokolü araç fonksiyonunu çıkarmak için eşitlik (4.7) yi kullanacağız. Tablo 4.1 özetlenmiştir.

TCP		
Reno-1	$F_s(x_s(t), q_s(t))$	$\left[x_s(t) + \frac{1-q_s(t)}{D_s^2} - \frac{2}{3} q_s(t) x_s^2(t)\right]^+$
	Utility	$\frac{\sqrt{3/2}}{D_s} \tan^{-1}\left(\sqrt{\frac{2}{3}}\, x_s D_s\right)$
Reno-2	$F_s(x_s(t), q_s(t))$	$\left[x_s(t) + \frac{1-x_s(t)D_s q_s(t)}{D_s^2} - \frac{2}{3} q_s(t) x_s^2(t)\right]^+$
	Utility	$\frac{1}{D_s} \log \frac{x_s D_s}{2x_s D_s + 3}$
Vegas	$F_s(x_s(t), q_s(t))$	$\begin{cases} x_s(t) + \frac{1}{D_s^2} & \text{if } x_s(t) < \overline{x}_s(t) \\ x_s(t) - \frac{1}{D_s^2} & \text{if } x_s(t) > \overline{x}_s(t) \\ x_s(t) & \text{otherwise} \end{cases}$
	Utility	$\alpha_s d_s \log x_s$
AQM		
RED	$G_l(y_l(t), p_l(t), v_l(t))$	$\begin{cases} 0 & r_l(t+1) \le \underline{b}_l \\ \rho_1(r_l(t+1) - \underline{b}_l) & \underline{b}_l \le r_l(t+1) \le \overline{b}_l \\ \rho_2(r_l(t+1) - \overline{b}_l) + m_l & \overline{b}_l \le r_l(t+1) \le 2\overline{b}_l \\ 1 & r_l(t+1) \ge 2\overline{b}_l \end{cases}$
	$H_l(y_l(t), p_l(t), v_l(t))$	$b_l(t+1) = [b_l(t) + y_l(t) - c_l]^+$ $r_l(t+1) = (1-\alpha_l) r_l(t) + \alpha_l b_l(t)$
REM	$G_l(y_l(t), p_l(t), v_l(t))$	$1 - \phi^{r_l(t+1)}$
	$H_l(y_l(t), p_l(t), v_l(t))$	$b_l(t+1) = [b_l(t) + y_l(t) - c_l]^+$ $r_l(t+1) = [r_l(t) + \gamma(\alpha_l b_l(t) + y_l(t) - c_l)]^+$
Delay	$G_l(y_l(t), p_l(t), v_l(t))$	$p_l(t+1) = [p_l(t) + \frac{y_l(t)}{c_l} - 1]^+$

Tablo 4.1. TCP-AQM algoritmasının model yazılışları.

4.2. RENO/AQM

TCP için biz sadece sıkışıklı giderme fazını modelleriz ve diğer önemli yaklaşımları ihmal ederiz. Örneğin yavaş başlama ve hızlı gönderim/hızlı düzeltme. AQM için sıkışıklığın ölçüsü ile sıkışıklık ölçüsünün geri dönüşü arasında ayırt edicilik kullanışlıdır. TCP RENO örneğin kayıp olasılığını sıkışıklığın ölçüsü olarak kullanır. Bu sıkışıklık ölçümünün değeri Ya paketleri düşürerek ya da bu olasılık ile ECN bitini ayarlayarak kaynağa geri beslenir. Bu yazıda sıkışıklık ölçümünün tasarımı ve onun eşitlik özelliği ile ilgileneceğiz. Ve bizim AQM modelimiz geri besleme mekanizmasını yakalamaz. İster paketlerin düşümü ister ECN bitini ayarlamak için İşaretleme metodunu kullanırız.

4.2.1. F,S,H Modeli

Bu alt bölümde TCP RED, RENO, REM modellerini sunacağız. Bu modellerin uygulanması aşağıdaki alt bölümde verilmiştir. AIMD nin ortalama hareketini sadece modelleyeceğiz ve TCP RENO [2] ile örneğin NEW RENO SACK ve benzeri arasında ki farkları çıkarmayacağız. Bütün bu protokoller eğer tur zamanında işaret yoksa her bir tur zamanında pencereyi artırır ve bunun dışında pencereyi yarılar. İki çeşit çoklu aktif azaltma vardır. RENO nun eski versiyonlarında işaretleme belirlendiğinde her seferinde pencere yarılanır. RENO nun yeni versiyonlarında tur zamanında bir ya da daha fazla işaret ve ise bir kere yarılanır. RENO nun eski versiyonlarına RENO–1 yeni versiyonlarına RENO–2 diyeceğiz. Aşağıda görüldüğü gibi farklı araç fonksiyonlarına ve kayıpsızlık özelliklerine

açıkça sahiptirler. Her sürüm için paket işaretleme olasılığını sıkışıklığın ölçümü olarak çıkarırız.

DROP TAİL altında tam tamponlama ile varmış olan paket düşürülür. Markala olasılığının dinamikleri için uygun olan ifadeyi bilmeyiz. Kullanılan kayıp değerin modeli örneğin kaynak [12] ve [9] de tamponsuz sorgu içindir. $p(t+1) = \left[1 - c / \sum_s x_s(t)\right]^+$. bu model eşitlik (4.8) i çözme ceza fonksiyonu gelişimi için uygundur ama DUALİTY gelişimi için uygunluk bağımlılığından dolayı değil. Böylece RED e REM için sadece model sunarız.

$w_s(t)$ pencere boyutu olsun D_s (Yayılma + eşitlik sorgu gecikmesi) sabit olduğunu iddia ettiğimiz tur zamanı eşitliği olsun. Literatürde alışıldığı şekliyle örneğin 1 kaynak [11], [13] $x_s(t) = w_s(t) / D_s$ tarafından belirlenmiş olsun ve t zamanında kaynak değeri olsun. Zaman birimi birçok tur zamanının sıralamasındadır ve kaynak değeri $x_s(t)$ zaman çizelgesinde ortalama değer olmalıdır. Dinamikler tur zamanı çizelgesinden daha küçüktür akışkan model tarafından yakalanmamıştır.

4.2.1.1.RENO–1

L hattında t zamanında $p_l(t)$ işaretleme olasılığı olsun. Anahtar yaklaşımı yaparız ki uçtan uca işaretleme olasılığı $q_s(t)$ kaynak algoritması bağlantı işaretleme olasılığının toplamı olarak davranır

$$q_s(t) = \sum_l R_{ls} p_l(t) \tag{4.13}$$

$p_l(t)$ küçük olduğunda bu anlamlıdır. $q_s(t) = 1 - \prod_{l \in L}(1 - p_l(t)) \cong \sum_{l \in L} p_l(t)$ olduğunda T periyodunda $x_s(t)$ paketlerinin değerini her bir birim zamanı için iletir ve bilgilendirmeleri yaklaşık aynı değerde alır(pozitif ve negatif). İddia edilir ki bütün paketler bilgilendirilmiştir. Ortalamada s kaynağı $x_s(t)$ $(1 - q_s(t))$ pozitif bilgilendirme sayısını her bir birim zamanı için alır ve her pozitif bilgilendirme $w_s(t)$ pencereyi büyültür $(1/x_s(t)$ ile), $x_s(t)$ $q_s(t)$ negatif bilgilendirme birim zaman için ve pencerenin yararlanmasında ortalama alır. Böylece t periyodunda penceredeki net değişim kabaca

$$x_s(t)(1 - q_s(t)).\frac{1}{w_s(t)} - x_s(t)q_s(t)\frac{1}{2}.\frac{4w_s(t)}{3}$$

Daha sonra RENO–1 in *Fs* kaynak algoritması aşağıdaki formülle verilir.

$$x_s(t+1) = \left[x_s(t) + \frac{1 - q_s(t)}{D_s^2} - \frac{2}{3} q_s(t) x_s^2(t) \right]^+ \tag{4.14}$$

İkinci derecen terim özellikle belirtir ki eğer değer çift kat olursa çoklu aktif düşüş frekansın iki katı büyüklüğün iki katıyla meydana gelir.

4.2.1.2 RENO -2

Pencere RENO–2 artırımı her bir tur zamanı D_s için 1 le eğer işaretleme yoksa eğer bir ya da daha fazla işaretleme varsa her bir tur zamanında pencere bir kere yarılar. Bunu aşağıdaki gibi modelleriz: her bir t periyodunda (bir kısım tur zamanın sıralamasında), pencere $1/D_s$ ile $1 - \hat{q}_s(t)$ olasılığı ile artırılır ve $2w_s(t)/3D_s$ ile azaltılır $q_s(t)$ olasılığı ile,

$\hat{q}_s(t)$uçtan uca olasılık olduğunda s yolunda en azından bir paket t periyodunda işaretlenmiştir. Tekrar $p_l(t)$olasılık olarak not edelim ki paket t periyodunda işaretlenmiştir ve $\hat{q}_s(t)$uçtan uca paket işaretleme olasılığı 13 tarafından verilir. $\hat{q}_s(t)$ yi aşağıdaki gibi modelleriz

$$\hat{q}_s(t) = w_s(t)q_s(t)$$

$w_s(t)$ pencere boyutu olduğunda. Bu sağlanır eğer paketler aynı pencerede birbirinden bağımsız olarak işaretlenmişse ve paket işaretleme olasılığı $q_s(t)$küçükse $\hat{q}_s(t) = 1-(1-q_s(t))$durumunda. Daha sonra t periyodunda pencere boyunda ki değişim

$$\frac{1}{D_s}(1-\hat{q}_s(t)) - \frac{2w_s(t)}{3D_s}\hat{q}_s(t)$$

$$= \frac{1-w_s(t)q_s(t)}{D_s} - \frac{2}{3}q_s(t)x_s(t)w_s(t)$$

böylelikle RENO-2 kaynak algoritması $F_s(x_s(t)q_s(t))$

$$x_s(t+1)$$

$$= \left[x_s(t) + \frac{1-x_s(t)q_s(t)D_s}{D_s^2} - \frac{2}{3}q_s(t)x_s^2(t) \right]^+ \tag{4.15}$$

İle verilir.

4.2.1.3. RED

RED kaynak [14] iki iç değişken ihtiva eder. Ani sorgu boyu $b_s(t)$ve ortalama sorgu boyu $r_l(t)$

$$b_l(t+1) = [b_l(t) + y_l(t) - c_l]^+ \tag{4.16}$$

$$r_l(t+1) = (1-\alpha_1)r_l(t) + \alpha_l b_l(t) \tag{4.17}$$

‘e göre güncellenmişlerdir.
$\alpha_l \in (0,1)$ olduğu durumda. Daha sonra (bunun ılımlı versiyonu) RED $p_l(t)$olasılığı ile paketi işaretler. Böylece $r_l(t)$ nin doğrusal büyüyen fonksiyonu:

$$p_l(t) = \begin{cases} 0 & r_l(t) \le 2b_l \\ \rho_1(r_l(t)-b_l) & b_l \le r_l(t) \le \bar{b}_l \\ \rho_2(r_l(t)-b_l)+m_l & \bar{b}_l \le r_l(t) \le 2\bar{b}_l \\ 1 & r_l(t) \ge 2\bar{b}_l \end{cases} \tag{4.18}$$

$$\rho_l = \frac{m_l}{\bar{b}_l - b_l} \quad ve \quad \rho_2 = \frac{1-m_l}{\bar{b}_l}$$

olduğunda
Bu eşitlikler (4.16)-(4.18) bu G,H modelini RED için belirler.

4.2.1.4 REM

REM kaynak [15] aynı zamanda iki iç değişken ihtiva eder. Ani sorgu uzunluğu $b_l(t)$ve $r_l(t)$ücret olarak adlandırılan bir değer. RED deki gibi $b_l(t)$eşitlik (4.16) tarafından modellenmiştir. $r_l(t)$ücreti

$$r_l(t+1) = \left[r_l(t) + \gamma(\alpha_l \beta_l(t) + y_l(t) - c_l\right]^+ \tag{4.19}$$

$\gamma > 1$ ve $\alpha_1 > 1$sabit olduğunda güncellenir. Paketleri işaretler olasılıkla ki $r_l(t)$ücretinde logaritmiktir.

$$p_l(t) = 1 - \phi^{-r_l(t)} \tag{4.20}$$

$\phi > 1$olduğunda. Pratikte eşitlik (19) la

$$r_l(t+1) = \left[r_l(t) + \gamma(\alpha_l(b_l(t) - \hat{b}_l) + y_l(t) - c_l\right]^+$$

ile yer değiştirebilir. $\hat{b}_l \geq 0$ hedef eşitlik geri bildirim olduğunda. Daha büyük $\hat{b}_l$ daha genellikle daha yüksek araçlaşmaya yöneltir. Özellikle geniş bir şekilde sorgu dalgalandığında kaynak [15]. Bu sürüm ile eşitlik sorgu uzunluğu teorem 3 de aşağıda $b_l^* = b_l$ dir 19 $b_l = 0$ ayarlanmasına dayanır. Logaritmik işaretleme olasılığı eşitlik (4.20) yaklaşık uçtan uca fiyat $r_l(t)$ s kaynağında $\sum_{l \in L}$ için yararlıdır. RENO tarafından kullanılmadığı zaman diğer yükseltme fonksiyonları kullanılabilir kaynak [15] de açıklandığı gibi. Örneğin işaretleme olasılığı $r_l(t)$ fiyatında doğrusal olabilir.

$$p_l(t) = \min\{pr_l(t), 1\} \tag{4.21}$$

bazı $\rho > 0$ sabitleri için. 0 olamayan hedef sorgu büyüklüğü b_l^* versiyonları ile ve doğrusal işaretleme olasılığı kaynak [16] un kontrolüne denktir. Diğer verilen AQM' nin uyarlanabilir sanal sorgusu gibi kaynak [9] aynı zamanda eşitlik (4.2) – (4.3) ün formunda modellenebilir. Eşitlik (4.16),(4.19), ve (4.20) –(4.21) eşitlileri (G,H) REM için model belirler.

RENO'nun Araç Fonksiyonları:

Bu alt bölümde RENO–1 ve RENO–2 nin araç fonksiyonlarını çıkaracağız. Göstereceğiz ki RED veya REM ile hem birincil hem de çiftli problem çözülür. Not edin ki bu at bölümlerin sonuçları RENO–1 ve RENO–2 kaynaklarına ve hem RED hem REN bağlantılarını içeren ağa uygulanır.
Yardımcı önerme 2: (F,G,H) fonksiyonları RENO -1 ve RENO -2, RED ve REM (14–21 eşitlikleri) modeli c1 c2 c4 durumlarını karşılar.

İspat: açıkça c1 durumu belirlenmiştir. Hem RENO–1 hem RENO–2 durumu için $x_s > 0$, Fs devamlı diferansiyelleşebilir ve $\partial F_s / \partial q_s \neq 0$ olduğunda RENO–1 için

$$q_s = \frac{3}{2x_s^2 D_s^2 + 3} =: f_s(x_s) \tag{4.22}$$

RENO–2 için

$$q_s = \frac{3}{x_s D_s (2x_s^2 D_s^2 + 3)} =: f_s(x_s) \tag{4.23}$$

Böylece $f_s(x_s)$hem RENO–1 hem de RENO–2 için açıkça düşer. Onların araç fonksiyonlarının keskin konkavlığı uygulanır. Böylece c2 ve c4 durumları sağlanır. Eşitlik (4.22) ve (4.7) yi karşılaştırarak, RENO–1 in araç fonksiyonu eşitlik (4.14)

$$U_s(x_s) = \frac{\sqrt{3/2}}{D_s} \tan^{-1}\left(\sqrt{\frac{2}{3} x_s D_s}\right) \tag{4.24}$$

benzer olarak RENO–2 nın araç fonksiyonu (4.15)

$$U_s(x_s) = \frac{1}{D_s} \log \frac{x_s D_s}{2x_s D_s + 3} \tag{4.25}$$

dir.

Not edin ki RENO–1 ve RENO–2 nin araç fonksiyonları VEGAS tan farklı olarak kullanılır. 0 bant genişliği almak için birçok şişe boynu bağlantıları kat eden kaynaklar için bu mümkündür (uçtan uca fiyat 1 birim olduğunda). Aşağıdaki sonuçlar RENO ya RED veya REM ile teorem–1 uygulanır. Bu çıkarır ki RED ile eşitlik sorgu uzunluğu örnekteki probleme dayanır (ağ topolojisi, yönlendirme kaynakların sayısı vb.) ve RED parametreleri ve böylece yük arttığı sürece kaçınılmaz büyüme gerçekleşir. RED parametreleri statik ya da dinamik olarak ayarlanabilir. Eşitlik sorgu uzunluğunu küçültmek için ama sadece potansiyel kararlı olmayışının sarf edilmesi ile; aşağıdaki örnekleri görünüz. Karşılaştırmada REM ile eşitlik sorgu uzunluğu yükten bağımsız 0 dır.

Teorem 2.1 RENO–1 ve RENO–2 kaynaklarının ve RED ve REM bağlantılarını içeren ağın eşitliği (x^*, p^*)olsun. Daha sonra (x^*, p^*)birincil eşitlik (4.8) ve çiftli problem eşitlik (4.9) eşitlik (4.24) de RENO–1 için ve eşitlik (25) de verilen RENO–2 kaynakları için verilen araç fonksiyonları ile çözer. Daha fazla olarak eşitlik değer vektörü x^* tektir. Eğer l bağlantısı RED de uygulanırsa daha sonra eşitlik sorgu uzunluğu b_l^* $b_l^* > b_l$- yi sağlar $p_l^* > 0$ ile. Eğer l bağlantısı REM e uygulanırsa daha sonra $b_l^* = 0$dır.

İspat: yardımcı önerme 2 tarafından c1 c2 c4 14-21 in kombinasyonları tarafından sağlanır. Verilen (x^*, p^*)eşitliği birincil çiftli en uygun olduğunu göstermek içindir. C3 ün aynı zamanda sağlandığını kontrol etmemiz gerekir. 16 dan $y_l^* \le c_l$hem RED hemde ren ile ve böylece birinci uygunluk sağlanır. Düşün ki $p_l^* > 0$ eğer l hattı RED de uygulanırsa eşitlik (4.17) ve (4.18) den

$$b_l^* = r_l^* > b_l \ge 0 \tag{4.26}$$

oluşur. Ama $b_l^* > 0$ $y_l^* = c_l$ de uygulanır. Eğer l hattı REM de uygulanırsa $p_l^* > 0$ve $r_l^* > 0$da uygulanır.

Böylece eşitlik (4.19) dan

$$\alpha_l b_l^* + y_l^* - c_l = 0 \tag{4.27}$$

Oluşur. $y_l^* \le c_l$ yi biliriz. Eğer $y_l^* \le c_l$ ise eşitlik (4.16) $b_l^* = 0$ a uygulanır. Ama bu eşitlik (4.27) yede yalanlanır. Böylece $y_l^* = c_l$ (ve $b_l^* = 0$). Hem RED hem ren ile tamamlayıcı gevşeklik gösterilir ve böylece c3 sağlanır ve (x^*, p^*) birincil çiftli en uygundur.

Daha fazla olarak eşitlik (4.26) gösterir ki $b_l^* > b_l$ - $p_l^* > 0$ RED ile olduğunda. REM ile takip eden elemanlar gösterir ki $b_l^* = 0$dır. Bu ispatı tamamlar.

NOTLAR

Eşitlik (4.22) ve (4.23) ilişkileri uygulanır ki RENO–1 ve RENO–2 büyük D_s ile kaynakları birbirinden ayırır. İyi bilindiği gibi birçok daha önceki çalışmalarda örneğin kaynak [13], [14], [17], [18] daha fazla olarak eşitlik (4.22) RENO 1 için

$$x_s = \frac{\sqrt{3/2}}{D_s}\sqrt{\frac{1-q_s}{q_s}} \cong \frac{\sqrt{3/2}}{D_s}\frac{1}{\sqrt{q_s}}$$

gibi yeniden yazılabilir. q_s olasılığı küçük olduğunda daha önceden geniş bir şekilde ilişki belirlenmiştir. Bazı yazarlar [16] [19]iddia eder ki RENO penceresini 1 ile büyütür her tur zamanında belirleyici olarak. Bu 1 in yerine eşitlik(4.14) de $(1-q_s(t))$ yi yerine koymaya dayanır. Bu model $x_s = \sqrt{3/2}/D_s\sqrt{q_s}$ ü verir

$$U_s(x_s) = -\frac{3/2}{x_s D_s^2} \tag{4.28}$$

Uygun araç fonksiyonu ile [19] ve [12] de kullanıldığı gibi (sabit terimleri ihmal ederek). RENO–2 için eşitlik (4.23)

$$q_s = \frac{3}{x_s D_s (2x_s D_s + 3)} \cong \frac{3}{2x_s^2 D_s^2}$$

yaklaştırılabilir. $2x_s D_s \ge 3$ olduğunda veya q_s küçük olduğunda. Daha sonra RENO–2 eşitlik (4.28) de verilen RENO1 deki gibi aynı araç fonksiyonuna sahiptir. DUALİTY teorisi ile verilen çift en uygun *p, x* değer vektörü

$$x_s = U_s'^{-1}(q_s) \tag{4.29}$$

da verilir. $q_s = \sum_l R_{ls} p_l$ olduğunda uygun değer vektörüdür. RENO nun değer ayarlaması işlemiş eşitlik (4.14) veya (4.15) bu stratejinin yumuşatılmış versiyonu olarak düşünülebilir. Aşağıdaki durumda $\bar{x}_s(y) = U_s'^{-1}(q_s)$ eşitlik (4.29) tarafından belirlenen hedef değeri olsun. Verilen RENO–1 in veya RENO–2 nin araç fonksiyonunu kullanarak daha sonra RENO1 için eşitlik (4.24) kullanılarak aşağıdaki

$$\bar{x}_s(y) = U_s'^{-1}(q_s) = \frac{1}{D_s}\sqrt{\frac{3}{2}\frac{1-q_s}{q_s(t)}}$$

Elde ediliyor. Değer artırımını (14) hedef değerinin $\bar{x}_s(t)$ terimleri olarak yeniden yazılabilir.

$$x_s(t+1) = \left[x_s(t) + \frac{2q_s(t)}{3}\bar{x}_s^2(t) - x_s^2(t)\right]^+$$

Böylece değeri atamadan $x_s(t+1)$ direk olarak $\bar{x}_s(t)$ hedef değerine bir adımda RENO–1 hali hazırdaki $x_s(t)$ değerine gider hedef değer $\bar{x}_s(t)$ den onların karesinin farkının orantısal değerini ekleyerek, RENO 2 için 23 den hedef değer $2q_s(t)\bar{x}_s^2(t) - x_s^2(t))/3$ karşılamalıdır. Böylece aşağıdaki gibi $\bar{x}_s(t)$ hedef değerinin terimleri olarak yazılabilir.

$$q_s(t) = \frac{3}{\bar{x}_s(t)D_s(\bar{x}_s(t)D_s + 3)}$$

Böylece eşitlik (4.25) de *Fs* hedef değeri $\bar{x}_s(t)$ nin terimleri olarak yeniden aşağıdaki gibi yazılabilir.

$$x_s(t+1)$$
$$= \left[x_s(t) + \frac{1}{D_s^2}\left(1 - \frac{x_s(t)(2x_s(t)D_s + 3)}{\bar{x}_s(t)(2\bar{x}_s(t)D_s + 3)} \right) \right]$$

Örneğin değeri yükseltin eğer $x_s(t) < \bar{x}_s(t)$ ve diğer durumlarda.

3.Bu gelişme burada alınmış kaynak [20] de ki gibi takip eder. $G_l H_l$ tarafından tamamen sorgu yönetim mekanizması modellenmiştir. Kaynak [21] deki model işaretleme olasılık fonksiyonunu F_s nin bir parçası olarak ihtiva eder, işaretleme araç fonksiyonuna AQM ye aynı zamanda TCP algoritmasına bağlıdır.

4.3. VEGAS / DROP-TAİL

VEGAS ın çiftli modeli kaynak [4] de geliştirilmiş ve doğrulanmıştır. Bu bölümde asıl sonuçları özetleyeceğiz. Eşitlik sorgu uzunluğunu ayarlamak için yeterince geniş tampon boyutu olan durumu düşünürüz. Böylelikle VEGAS kaynakları tek eşitliğe gelir. Bu durumda eşitlikte paket kaybı olamaz. [4] de gösterilmiştir ki VEGAS sorgu gecikmesini sıkışıklık ölçümü olarak kullanır. $p_l(t) = b_l(t)/c_l$ $b_l(t)$t periyodundaki sorgu uzunluğu olduğunda. Güncelleme kuralları böylelikle $G_l(y_l(t), p_l(t))$

$$p_l(t+1) = \left[p_l(t) + \frac{y_l(t)}{c_l} - 1 \right]^+ \tag{4.30}$$

Tarafından verilmiştir (eşitlik (4.16) nın her iki tarafı *cl* ye bölünmesi ile elde edilir). Böylece VEGAS için AQM her herhangi bir iç değişkene bağlı kalmaz. Verilen $\bar{x}_s(t)$

$$\bar{x}_s(t) = \frac{\alpha_s d_s}{q_s(t)} \tag{4.31}$$

tarafından verilmiş ve hedef değeri olsun, α_s VEGAS ın parametresi olduğunda ve d_s s kaynağının tur zaman yayılma gecikmesi olduğunda. Kaynak değeri için güncelleme kuralı daha sonra $F_s(x_s(t), q_s(t))$ olarak

$$x_s(t+1) = x_s(t) + \frac{1}{D_s^2} 1(\bar{x}_s(t) - x_s(t)) \tag{4.32}$$

tarafından verilir.

$1(z) = 1\ eger\ z > 0, -1$ Olduğunda eğer z>0,- $x_s = \bar{x}_s = \alpha_s d_s / q_s 1$ eğer z<0 ve 0 eğer z=0 ise. Eşitlikte $U_s'(x_s) = \alpha_s d_s / x_s$ veya $U_s(x_s) = \alpha_s d_s \log x_s$ elde ederiz. Aşağıdaki sonuçlar 18 de ispatlanır. Kısmen uygulanır ki her bir bağlantıda basit konkav programı çözerek sorgu uzunluğunu hesaplayabiliriz.

Teorem4. VEGAS/DROP TAİL ın (x^*, p^*) eşitlik (4.30)-(4.32) de modellendiği gibi birincil (4.8) ve çiftli problem (4.9) eşitliği ile çözülür, araç fonksiyonu Us

$U_s(x_s) = \alpha_s d_s \log x_s$

Tarafından verilen. Daha fazla x^* tektir ve iyi orantılanmıştır. L bağlantılarında ki eşitlik sorgu uzunluğu $c_l p_l^*$ dır. Teorem de verilen araç fonksiyonu ile VEGAS ın değer ayarlaması eşitlik (4.32) eşitlik (4.29) un yumuşatılmış versiyonu olarak teoremde verilen araç fonksiyonu ile çıkarılabilir. $x_s(t+1)$ değerini ayarlamak yerine bir adımda $\bar{x}_s(t)$ hedef değerine (4.29) tarafından belirlenir, VEGAS $x_s(t)$ hali hazırdaki değerine gider. $1/D_s^2$ ile her adımda hedef değer $\bar{x}_s(t)$ yaklaşır.

KAYNAKLAR

[1]. V. Jacobson. Congestion avoidance and control. Proceedings of SIGCOMM'88, ACM, August 1988. An updated version is available via ftp://ftp.ee.lbl.gov/papers/ congavoid.ps.Z.

[2]. Stevens. TCP/IP illustrated: the protocols, volume 1. Addison-Wesley, 1999. 15th printing.

[3]. Lawrence S. Brakmo and Larry L. Peterson. TCP Vegas: end-to-end congestion avoidance on a global Internet. IEEE Journal on Selected Areas in Communications, 13(8): 1465— 80, October 1995. http://cs.princeton.edu/nsg/papers/ jsac-vegas.ps.

[4]. Steven H. Low, Larry Peterson, and Limin Wang. Understanding Vegas: a duality model. J. of ACM, 49(2):207-235, March 2002. http://netlab.caltech.edu.

[5]. Frank P. Kelly, Aman Maulloo, and David Tan. Rate control for communication networks: Shadow prices, proportional fairness and stability. Journal of Operations Research Society, 49(3):237-252, March 1998

[6]. Steven H. Low and David E. Lapsley. Optimization flow control, I: basic algorithm and convergence. IEEE/ACM Transactions on Networking, 7(6):861—874, December 1999. http://netlab.caltech.edu.

[7]. Massoulie and J. Roberts. Bandwidth sharing: objectives and algorithms. In Infocom'99, March 1999.

[8]. Jeonghoon Mo and Jean Walrand. Fair end-to-end window-based congestion control. IEEE/ACM Transactions on Networking, 8(5):556-567, October 2000.

[9]. Srisankar Kunniyur and R. Srikant. End—to-end congestion control schemes: utility functions, random losses and ECN marks. In Proceedings of IEEE Infocom, March 2000. http: //www.ieee-infocom.org/2000/papers/401.ps.

[10]. ernando Paganini, John C. Doyle, and Steven H. Low. Scalable laws for stable network congestion control. In Proceedings of Conference on Decision and Control, December 2001. http://www.ee.ucla.edu/~paganini.

[11]. D. Bertsekas. Nonlinear Programming. Athena Scientific, 1995.

[12]. R. J. Gibbens and F. P. Kelly. Resource pricing and the evolution of congestion control. Automatica, 35:1969—1985, 1999.

[13]. atthew Mathis, Jeffrey Semke, Jamshid Mahdavi, and Te-unis Ott. The macroscopic behavior of the TCP congestion avoidance algorithm. ACM Computer Communication Review, 27(3), July 1997. http://www.psc.edu/networking/ papers/model_ccr97.ps.

[14]. S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. IEEE/ACM Trans. on Networking, 1(4):397-413, August 1993. ftp://ftp.ee.lbl. gov/papers/early.ps.gz.

[15]. Sanjeewa Athuraliya, Victor H. Li, Steven H. Low, andQinghe Yin. REM: active queue management. IEEE Network, 15(3):48-53, May/June 2001. Extended version inProceedings of ITC17, Salvador, Brazil, September 2001. http://netlab.caltech.edu.

[16]. Chris Hollot, Vishal Misra, Don Towsley, and Wei-Bo Gong. On designing improved controllers for AQM routers supporting TCP flows. In Proceedings of IEEE Infocom, April 2001. http://www-net.cs.umass.edu/papers/ papers.html.

[17]. S. Floyd. Connections with multiple congested gateways in packet—switched networks, Part I: one—way traffic. Computer Communications Review, 21(5), October 1991.

[18]. T. V. Lakshman and Upamanyu Madhow. The performance of TCP/IP for networks with high bandwidth—delay products and random loss. IEEE/ACM Transactions on Networking, 5(3):336—350, June 1997. http://www.ece.ucsb. edu/Faculty/Madhow/Publications/ton97.ps.

[19]. Deepak Bansal and Hari Balakrishnan. Binomial congestion control algorithms. In Proceedings of IEEE Infocom, April 2001.

[20]. Steven H. Low, Fernando Paganini, and John C. Doyle. Internet congestion control. IEEE Control Systems Magazine, 22(1):28-43, February 2002.

[21]. Steven H. Low. A duality model of TCP flow controls. In Proceedings of ITC Specialist Seminar on IP Traffic Measurement, Modeling and Management, September 18-20 2000.

BÖLÜM 5

VEGAS - DUALİTY MODEL

Ağdaki küresel en uygun olma problemini çözmek için dağılmış öncelikli çift algoritma kaynak tarafından taşınır. TCP IP VEGAS sıkışıklık kontrol mekanizmasını çoklu bağlantılı ve çoklu kaynaklı modelini tanımlarız. Bu model gecikmelerin ve TCP VEGAS ın kayıp özelliklerinin kökenini anlamamızı sağlar. Ağ uygun tamponlama olduğu zaman ağ kapasitesinin uygun ağırlıkta olduğu varsayılarak VEGAS kararlığını belirtir. Bundan korunmak için REM aktif sorgu yönetimini ne kadar kullanacağımız tavsiye eder ve bu sıkışıklıktan kaynaklanan sonuçların mekanizmasını açıklar.

5.1. VEGAS Modeli

Bu VEGAS modelinin gösterildiği bölümdür ve VEGAS ın nesnelerini gösterir. VEGAS ın algoritması çiftli bir metottur problemlerin çoğunu çözmek için. Bu çabanın hedefi VEGAS ın kararlılığını daha iyi anlamakla anlamakla olur.

5.1.1. Ön Hazırlık

Yönlendirmeli ağ çoklu yönlü L tarafından modellenmiştir. İletim kapasitesi c_l ile $l \in L$ eleman ve sonsuz tamponlama boşluğu. s kaynaklar tarafından paylaşılmıştır. *s* kaynağı l(s) alt setine dönüşür $L(s)$ alt eleman hatlarını $L(s) \subseteq L$ dönüştürür. *Zs* değeri iletildiğinde $U_s(x_s)$ değerine ulaşır.(örneğin her saniyedeki paketler) Tur zamanı çiftleme gecikmesi s kaynağı için *ds* olsun. Her L hattı için $\overline{S}_s(l) = \{s \in S \mid l \in L(s)\}$ L hattını kullanan kaynak setleri olsun. Tanımlama $l \in L(s)$, $s \in S(l)$ olduğu zamandır.

VEGAS ın bir tercümesine göre kaynak gerçek değeri ve tahmin edilen değerinin farkını gösterir ve penceresindeki artırımlar veya azatlımlar bir sonraki tur zamanında alfa s parametresinden daha büyük veya daha küçük olmasına bağlı olarak. Eğer fark sıfırsa pencere boyutu değişmez. Senkron parçaları zaman modeline göre bunu modelleriz $W_s(t)$ zamanında pencerenin kaynağı olsun ve $D_s(t)$ uygun tur zamanı olsun (sorgu gecikmesi + çiftleme) her bir ayrı zaman için $1/D_s(t)$ (bir paketin pencere boyunun değişimini modelleriz her dönüş zamanı için her ayrık zaman da $1/D_s(t)$ nin değişimi tarafından). Bu s kaynağı aşağıdakine göre penceresini ayarlar VEGAS kaynak algoritması

$$w_s(t+1)=\begin{cases} w_s(t)+\dfrac{1}{D_s(t)} \; if \; \dfrac{w_s(t)}{d_s}-\dfrac{w_s(t)}{D_s}<\alpha_s \\ w_s(t)+\dfrac{1}{D_s(t)} \; if \; \dfrac{w_s(t)}{d_s}-\dfrac{w_s(t)}{D_s}>\alpha_s \\ w_s(t) \quad else \end{cases} \tag{5.1}$$

kaynak yazıda [1] $W_s(t)/D_s$ beklene değer olarak belirtilmiştir. $w_s(t)/d_s$ asıl değerdir. $w_s(t)/d_s - w_s(t)/D_s(t)$ Arasındaki fark DİFF dir. Asıl uygulama tur zamanı çiftleme gecikmesi d_s daha sonra belirlenecek tur zamanı en az değeri tarafından tahmin edilir. α_s birimi KB/sn yi söyler. α_s in önemi bölüm 3 de açıkladık. [1] in açıklamasında DİFF i α_s ve β_s arasından tutmak için pencereyi ayarlar. $\alpha_s < \beta_s$ için $\alpha_s = \beta_s$ Olduğunu basitçe öne süreriz. Bu VEGAS ın özünü oluşturur. $\alpha_s < \beta_s$ Kullanımın etkisi [2] tarafından açıklanmıştır.

$w_s(t) := w_s(t)/D_s(t)$ bant genişliğini t zamanında s zamanı için ifade etsin. $W_s(t)$ pencere boyutu - Bant genişliği gecikme ürünü $d_s w_s(t)$ s yolunda tamponlanan toplam geri bildirim a eşittir. Buda kaynak[1] deki durumla d_s nin çarpılmasıdır. Görüyoruz ki pencerenin kaynak artırımı ya da azatlımı toplam geri bildirim $w_s(t) - d_s x_s(t)$ $\alpha_s d_s$ den daha küçük ya da büyük olması bağlıdır. Bu ikinci VEGAS algoritma yaklaşımıdır. Bölüm 5.1 de 3. yaklaşımı açıklayacağız. Eşitlik (5.1) sadece kaynak dinamiklerini belirler ve ağ davranışını tamamen açıklamaz. $D_s(t)$ dönüş zamanı gecikmelerini belirleyen bağlantı dinamiklerini ihtiva eder aynı zamanda. $D_s(t)$ gecikmesi s kaynağının $w_s(t)$ penceresine sadece değildir aynı zamanda diğer kaynakların paylaşılan bağlantı üzerinde olmasına da bağlıdır.

5.1.2. Vegasın Nesneleri

VEGAS eşitliğini şu anda yorumlayabiliyoruz. Detaylı ağ dinamiklerini ihtiyacı olmadığı için takip eden alt başlıkta bütün özelliklerini erteliyoruz.

Algoritma birleştiği zamanı $w^* = \left(w_s^* . s \in S\right)$ ve dönüş zamanı eşitliğini gösterir. $D^* = \left(D_s^* . s \in S\right)$ gerçekleşir.

$$\frac{w_s^*}{d_s} - \frac{w_s^*}{D_s^*} = \alpha_s \; bütün \; s \in S \; için \tag{5.2}$$

İlk sonucumuz gösterir ki VEGAS kaynakları sahiptir eşitlik (5.3) ile

$$U_s(x_s) = \alpha_s d_s \log x_s \tag{5.3}$$

Eşitlik (5.1) fayda fonksiyonlarıdır. Bundan daha fazla VEGAS ın nesneleri kaynak değerlerini seçmek içindir. $x = \left(x_s, s \in S\right)$

$$\max_{x \geq 0} \sum_s U_s(x_s) \tag{5.4}$$

$$\sum_{s \in S(l)} x_s \leq c_l . \; l \in L \tag{5.5}$$

U_s hizmet fonksiyonu tam olarak iç bükey artışlıdır. VEGAS kaynağının çok arttığı anlamına

gelir (fayda değer artışı). Ama azaltan bir geri dönüş vardır (dış bükeylik). Bağımlılık belirtir ki toplam kaynak değeri herhangi bir hatta kapasiteyi aşamaz. Eşitlik (5.4) ve (5.5) öncelikli problemi belirttik. X değer vektörü bağımlılığı mümkün kılabileceğini belirtti ve mümkün olan x öncelikli uygunluğa uygunluk olarak adlandırılır (veya basitçe uygun). Tek uygun değer vektörü objektif fonksiyonun kesin iç bükey olduğunda mevcuttur ve devamlıdır, mümkün çözüm seti yoğundur.

Teorem. 1: $w^* = \left(w_s^*, s \in S\right)$ VEGAS ın eşitliği olsun $D^* = \left(D_s^*, s \in S\right)$ dönüş zamanı eşitliği olsun ki (2) eşitliğini gerçekleştirsin. Paketler birebir bütün hatlara sunulsun sonra kaynak değerlerinin eşirliği $x^* = \left(x_s^*, s \in S\right)$ formül tarafından belirlensin. Bu değer eşitlik (5.3) ve (5.5) için Tek uygun çözüm dür.

İspat: Karush-Kuhn-Tucker teoremi tarafından uygun kaynak değer vektörü $x^* \geq 0$ Sadece ve sadece $p^* = \left(p_s^*, l \in L\right) \geq 0$ olduğu zaman uygundur. Bütün s ler içinde

$$U_s'(x_s^*) = \frac{\alpha_s d_s}{x_s^*} = \sum_{l \in L(s)} p_l^* \tag{5.6}$$

Bütün l ler için $p_l^* = 0$ dır . Bağlantının toplam kaynak değeri $\sum_{s \in S(l)} x_s^* < c_l$ kapasitesinden kesin küçüktür. (tamamlayan sarkıklık). *p** vektörünü bağlantılar sağlar ki Geri bildirim un eşitliğini ispat ediyoruz. Ve bu eşitlik uygun değerdedir.

b_l^* geri bildirim eşitliğinin l bağlantısındaki değeri olsun. b_l^* ın parçası s kaynağına ait olan ilk giriş ilk çıkış servis disiplini altında hat kapasitesi olduğu zaman $\frac{x_s^+}{c_l} b_l^* \; c_l$ dır. Böylelikle kaynak s $\sum_{l \in L(s)} \frac{x_s^+}{c_l} b_l^*$ eşitliğinde onun yolunda içerir. Pencere boyutu bant genişliği gecikme ürünü + toplam yoldaki geri bildirime eşitliği zaman

$$w_s^* - x_s^* d_s = \sum_{l \in L(s)} \frac{x_s^*}{c_l} p_l^* \tag{5.7}$$

Olur. Böylelikle eşitlik (5.2) de $w_s^* = w_s^* / D_s^*$

$$\alpha_s \; w_s^* = \frac{w_s^*}{d_s} - \frac{w_s^*}{D_s^*} = \frac{1}{d_s}(w_s^* - x_s^* d_s) = \frac{1}{d_s}\left(\sum_{l \in L(s)} \frac{x_s^*}{c_l} p_l^* \right)$$

Eşitliğini elde ederiz son eşitlik (5.7) den takip ederiz.

$$p_l^* = \frac{b_l^*}{c_l}$$

'nü belirlemekle ve terimleri yeniden düzenleyerek formül 6 ya ulaşırız.. Açıkça *x** geri bildirim sınırsız büyüdüğü zaman *x** uygun olmalıdır. Eşitlik (5.7) nin tersini söylemlidir. Hat l bağlantıdaki $p_l^* = 0$ geri bildirim eşitliği. Eğer toplam kaynak değeri kapasiteden kesinlikle küçükse tamamlayıcı sarkıklık durumu aynı zamanda gerçekleşir.

5.1.3. Çift Problem

Teorem 1 de VEGAS birbirine yaklaşırsa eşitlik (5.3 – 5.5) deki uygunluk problemini eşitliğin çözdüğü savunulur. Burada soru VEGAS algoritmasının veya daha karışık olan

eşitlik (5.16—5.19) belirlemelerinin mi kalacağıdır. Aslında birbirine yaklaşacağıdır. Birbirine sert yaklaşma analizi zordur çünkü güncelleme fonksiyonu doğrusal ve devamlı değildir. Kaynak [3] analizinde iki kaynağın tek bağlantıda paylaşımını görürüz. Bu alt bölümde 4 ve 5 deki çift problem için ve sonrasında VEGAS algoritmasını eğim atma algoritması ele alacağız. Bu eğim atış algoritması yeterince küçük adım ölçüleri ile birbirine yaklaşma sağlandığı zaman ispat edilebilir. Bunu Yaklaşık algoritma yaklaşımı olarak ima etmediğimizde eğim atış algoritmasının kaydı için VEGAS ı tavsiye eder. Bundan daha fazla bu izah VEGAS ağ modelinin tamamına yöneltir bir sonraki bölümde açıklanacağı gibi kaynak[4] duality gelişmelerini görmekle başlayacağız. Kaynak [4]; [5] eşitlik (5.4) ve (5.5) i ve çift problemi çözmek için ve tartılmış eğim fırlatma algoritmasını tanıtır. Bundan sonraki alt başlıkta VEGAS algoritmasını tanıtacağız. Bu algoritmanın yaklaşık versiyonu olarak. Her bir l bağlantıya bağlı olarak *Pl* çift değişkendir. Eşitlik (5.4) ve (5.5) Lagrangian kaynak [6] ve eşitlik (5.4) ve (5.5) in çift problem belirtildiği gibi:

$$L(x,p) = \sum_s U_s(x_s) - \sum_l p_l \Big(\sum_{s \in S(l)} x_s - c_l \Big)$$

$$= \sum_s \left(U_s(x_s) - x_s \sum_{l \in L(s)} p_l \right) + \sum_l p_l c_l$$

Eşitlik (5.4) ve (5.5) in çift probleminin fonksiyon $D(p) := \max_{x \geq 0} L(x,p)$ gibidir. İlk terimin x_s den artırılabilir olduğunu fark edin ve

$$\max_{x_s \geq 0} \sum_s (U_s(x_s) - x_s \sum_{l \in L(s)} p_l) = - \sum_s \max_{x_s \geq 0} (U_s(x_s) - x_s \sum_{l \in L(s)} p_l)$$

Böylelikle çift problem çift vektörü seçmektir. $p = (p_l, l \in L)$öyle ki

$$B_s(p^s) = \max_{x_s \geq 0} U_s(x_s) - x_s p^s \tag{5.9}$$

$$p^s = \sum_{l \in L(s)} p_l \tag{5.10}$$

olduğunda

$$\min_{p \geq 0} D(p) := \sum_s B_s(p^*) + \sum_l p_l c_l) \tag{5.8}$$

çiftli değişken p_l yi l bağlantısındaki her birim bant genişliği fiyatı olarak açıklayacağız. Sonra p_s eşitlik (5.10) da s yolunda her bant genişliğinin fiyatıdır. Böylelikle $x_s(p^s)$ eşitlik (5.9) da kaynak s ye bant genişliği maliyetini temsil eder. $x_s . U_s(x_s) - x_s p^s$ net x_s değerindeki iletimin karıdır. Ve $B_s(p^s)$maksimum s kazancı p5 (ölçülebilir) verilen fiyatta gerçekleştirilebilen maksimum s kazancı olarak belirtilir. Verilen fiyat vektörü $p \geq 0$ s kaynağı tek $x_s(p^s)$değerini ikna edebilir ki formül9 u maksimum değere getirir sadece lokal bilgilere dayanarak. Daha fazla olarak Duality teori tarafından kaynak [6] fiyat vektörü *p** eğer çiftli uygun eşitlik (5.8) e minimize ise $x_s(p^{*s})$ilk uygun değeri aynı zamanda tek uygun değeri olur.

Bu yazının kalanında bağlantı fiyatını p_l olarak ifade edeciğiz. $p^s = \sum_{l \in L(s)} p_l$ yol fiyatı(s kaynağının) ve vektör p $p = (p_l l \in L)$ basitçe fiyattır. VEGAS için *pl* bağlantı fiyatı l bağlantısındaki sorgu gecikmesi olarak döner: aşağıdaki bölümü görün. Uygun p* gölge fiyattır (Lagrangian çarpımı). p_l^* yorumlayıcısı ile uç artırım toplam $\sum_s U_s(x_s)$ l nin cl kapasitesindeki uç artırımı için tekrarlayan eğim izdüşümü algoritması kaynak [5] deki çift

problemi çözmek için tasarlanmıştır. Çift objektif fonksiyonu d nin $\nabla D(p(t))$eğimini not edin. Fiyatlar eğim $D(p(t)$ yi küçültmek için ters yönde yararlanmıştır.

$$p(t+1) = \left| p(t) - \gamma \Theta \nabla D(p(t)) \right|^{+}$$

Burada $\gamma > 0$ sabittir. $\Theta = diag(\theta l \in L)$ Pozitif dik ölçülür matristir ve $|z|^{+} = \max\{0.z\}$ elemandır. Çift problemin yapısı dağıtılmış merkezleşme ve yukarıdaki algoritmaların yapısına müsaade eder.

$x_s(t)$ nin yerine koyulan p, $p(t)$ ile eşitlik (5.9–5.10) maksimize eden tek kaynak değerini gösterir ve $x^l(t) = \sum_{s \in S(l)} x_s(t)$ l deki toplam kaynak değerini gösteri. $p^s(t) = \sum_{l \in L(l)} p_l(t)$ s kaynağını yol fiyatını gösterir. Sonra $U_s(x_x) = \alpha_s d_s \log x_s$ VEGAS ın araç fonksiyonu ile eğim atış algoritması yaklaşım uygulaması takip etmesi [6] olarak görülebilir.

$$x_s(t) = \frac{\alpha_s d_s}{p^s(t)} \tag{5.12}$$

olduğu zaman

$$p(t+1) = \left[p_l(t) + \gamma \theta_1 (x^l p(t)) - c_l \right]^{+} \tag{5.11}$$

Açıklamak için $x^l(t)$ l bağlantısında bant genişliği için talepleri göstereceğini not edin ve c_l mevcut olanı gösterir. Böylelikle fiyat arz ve talep kanunlarına göre ayarlanır. Eğer arz talebi aşarsa fiyatı yükseltir değilse düşürür. [4] algoritmasında $\theta_l = 1$ özel bir durum vardır. γ büyütme faktörü kaynak [5] deki zaman bağımlı Hessian matrisi $\nabla^2 D(p(t))$ tersi olarak seçilmiştir. Eşitlik (5.12) tarafında kaynak s eşitlik (5.9-5.10) un tek yükseltici değerine atar. Ekonomideki arz fonksiyonu gibidir. $p^s(t)$ yol fiyatı ne kadar büyükse (yol ne kadar sıkışmışsa) kaynak değerleri o kadar küçüktür. Algoritmanın merkezleştirilmemiş doğası her hat ve har kaynak sadece lokal bilgileri tek başlarına güncellendiğine dikkat edin. Takip eden sonuçlar söyler ki ölçülmüş eğim atış algoritması eşitlik (5.11) ve (5.12) tarafından belirlenir. Tekil uygun kaynak değerine gider. VEGAS ı Bu algoritmanın yaklaşık versiyonu olarak düşünüldüğünde bu teorem kararlılığının altını çizer.

Teorem 2: θ_l adım boyutu yetirince küçük olduğunda herhangi bir $x(0) \geq 0$ başlangıç değerinde başlarken ve $p(0) \geq 0$ fiyatları her limit noktası ($x^*.p^*$) *(x(t),p(t))* durumun eşitlik (5.10) ve (5.12) tarafından oluşturulur Kaynak [5].

İspatı aynı zamanda ($x^*.p^*$). ın uygunluğu garantileyen adım büyüklüğünün sınırını açık eder. $\bar{L} := \max_{s \in S} |L(s)|$ ve $\bar{S} := \max_{l \in L} |S(l)|$ belirle. $\bar{L}$ kaynak tarafından kullanılan en uzun yolun uzunluğudur. Ve $\bar{S}$ en sıkışık bağlantıyı paylaşan kaynakların sayısıdır. $x_s(t)$ kaynak değerini üstten sınırlı olduğunu düşünün şöyle ki $\alpha_s d_s / x_s^2(t) \geq \alpha d / \bar{x}^2$ bütün s ler için. Büyüklük faktörü $\theta_l = 1/c_l$ olduğu zamana VEGAS algoritmasını özelleştirin. Düşünün ki hat kapasitesi alttan sınırlı. Teorem 2.2 nin sonucu aşağıdadır.

$$\gamma = \frac{2\alpha dc}{\bar{L}\bar{S}\bar{x}^2}$$

5.1.4. Vegas Algortiması

VEGAS algoritmasını yaklaşık olarak ölçülmüş eğim fırlatma algoritması olarak düşünürüz. Bu algoritma adapte olabilen sıkışıklık kontrolünden daha yakındır. Bağlantı algoritması (5.11) sıkışıklık ölçümünü $p_l(t)$ $p^s(t)$ hesaplanır ve kaynak algoritması eşitlik (5.12) iletim değerini sıkışıklık geri dönüş $p^s(t)$ ye adapte eder. Bu algoritmayı uygulamak için VEGAS-kaynak bağımlı mekanizma iki kabulü göstermelidir. 1-Bağlantı ücretlerinin nasıl hesaplandığı ve 2- yol ücretlerine her bir kaynak için onların değerlerini ayarlamak için nasıl geri dönüldüğü. Bunları göreceğiz ilki ücret ayarlaması eşitlik (5.11) Her bir bağlantıdaki tampon işlem tarafından gerçekleşir. 2. yol ücretleri kesin olarak dönüş zamanı boyunca kaynağa geri beslenir. Verilen yol ücreti $p^s(t)$, s kaynağı yaklaşık Eşitlik (5.12) nin versiyonudur.

Özellikle l bağlantısının giriş değerini s kaynağından $x_s(t)$ de *t* zamanında düşünün. Daha sonra *l* bağlantısındaki toplam giriş değeri $x^l(t) = \sum_{s \in S(l)} x_s(t)$ olsun ve *l* bağlantısında tampon aralığı $b_l(t)$, $b_l(t+1) = \left[b_l(t) + x^l(t) - c_l\right]^+$ 'e dayandırılarak oluşur.

Her iki tarafı *cl* ye bölmekle

$$\frac{b_l(t+1)}{c_l} = \left[\frac{b_l(t)}{c_l} + \frac{1}{c_l}(x^l(t) - c_l)\right]^+ \tag{5.13}$$

eşitlik elde edilir.

$p_l(t) = b_l(t)/c_l$ belirlerken eşitlik (5.13) (5.11) in aynısıdır y=1 ile ve ölçüm faktörü $\theta_l(t) = 1/c_l$ olduğu zaman. $x_s(t)$, $x^l(t)$ kaynak değeri hariç tutulur. Eşitlik (5.12) den kesin olarak farklıdır. Daha sonra açıklandığı gibi…

1 den tekrar çağrılır ki VEGAS algoritması $w_s(t)$

$$w_s(t) - x_s(t)d_s < \alpha_s d_s \quad veya \; w_s(t) - x_s(t)d_s > \alpha_s d_s \tag{5.14}$$

eşitliğe dayanarak güncellenir.

2.1 in ispatındaki gibi bu değer geri bildirime bağlanır böylelikle ücretler yolda

$$w_s(t) - x_s(t)d_s = x_s(t)\sum_{l \in L(s)} \frac{b_l(t)}{c_l} = x_s(t)\sum_{l \in L(s)} p_l(t) = x_s(t)p^s(t) \tag{5.15}$$

eşitlik gibidir.

Eşitlik (5.14) numaralı koşulda

$$x_s(t) < \frac{\alpha_s d_s}{p^s(t)} \quad veya \quad x_s(t) > \frac{\alpha_s d_s}{p^s(t)}$$

oluşur.

VEGAS kaynağı haki hazırda ki kaynak değeri $x_s(t)$ ye hedef değeri $\alpha_s d_s / p^s(t)$ yi karşılaştırır. Pencere artırılmış veya azaltılmış $1/D_s(t)$ ile karşılaştırılır. Bir sonraki periyotta hali hazırdaki kaynak değeri $x_s(t)$ hedef değerinden $\alpha_s d_s / p^s(t)$ daha küçük ya da daha büyük olmasına dayandırılır. Ayarlamada eşitlik (5.12) algoritması hedef değerine bu değeri ayarlar.

Özetle VEGAS algoritmasını aşağıdaki doğrusal olmayan değerlerle göstermiş olduk.

VEGAS Ağ Modeli

$$v_s(t) := \frac{1}{d_s + p^s(t)} \left\{1(x_s(t)p^s(t) < \alpha_s d_s) - 1(x_s(t)p^s(t) > \alpha_s d_s)\right\} \tag{5.18}$$

$$x_s(t) := \frac{w_s(t)}{d_s + p^s(t)} \tag{5.19}$$

olduğunda

$$p_l(t+1) = \left[p_l(t) + \frac{1}{c_l}(x^l(t) - c_l) \right]^+ \quad bütün\, l\, linkleri\, için \tag{5.16}$$

$$w_s(t+1) = \left[w_s(t) + v_s(t) \right]^+ \quad bütün\, s\, kaynakları\, için \tag{5.17}$$

Böylelikle $D_s(t) := d_s + p^s(t)$ s nin tur zamanıdır, $[z]^+ = \max\{0, z\}$ dir, ve fonksiyon 1 (A) göstericisinden 1 e eşittir. Eğer A doğru ve 0 sa. Bu lineer olmayan sistem eğim fırlatma algoritması eşitlik (5.11-5.12) nin yaklaşık versiyonu (5.8)-(5.10) daki çift problemi için olarak düşünülebilir.. Eşitlik (5.16)-(5.19) un tamamlanmış noktası tarafından verilmiştir ve $x_s^* p^{*s} = \alpha_s d_s$ $x_s^* = 0$ kadar ve $x^{*l} \le c_l$ İle eğer $p_l^* > 0$ Eşitliği ile karşılar Tam olarak Karush-Kuhn-Tucker durumu önceki eşitlik (5.3)-(5.5) problemi için mevcuttur. Böylelikle eşitlik toplam araç fonksiyonuna maksimize olur teorem 2. 1 de açıklandığı gibi.

5.1.5. Notlar

Teorem 2.2 deki uygun durumda kararlılık için $\gamma > 0$ a ihtiyaç duyar yeterince küçük olmasına. Kaynak VEGAS algoritması bunun la birlikte $\gamma = 1$ i idea eder (eşitlik (5.11)-(5.16) yı karşılaştırın). Şimdi y yi yeniden tanıtmak için bir yol tanımlayın. Her iki tarafı eşitlik (5.13) ile çarparsak $\gamma > 1$ tarafından ve $p_l(t) = \gamma \frac{b_l(t)}{c_l}$ ile belirleyerek,

$$p_l(t+1) = \left[p_l(t) + \gamma \frac{1}{c_l}(x^l(p(t)) - c_l) \right]^+$$

oluşturulur.

Burada fiyatlarda ağırlıklı sorgu gecikmesi kullanılarak, fiyat ihtiyaç duyulmayan y nın adımı ile bulunur. Daha sonra (15)

$$\gamma(w_s(t) - x_s(t)d_s) = x_s(t) \sum_{l \in L(s)} \gamma \frac{b_l(t)}{c_l} = x_s(t) p^s(t) \tag{5.20}$$

ye geliştirir.

Araç fonksiyonunu modifikasyonlar güncellememelidir aynı zamanda eşitlik değeri $w_s(t)$ eşitlik (5.18) e dayanarak ayarlanmalıdır. Böylelikle eşitlikte $p^{*s}(t) = \alpha_s d_s / x_s^*$ $\gamma = 1$ için bu ihtiyaç eşitlik (5.20) ile beraber duruma bağlı VEGAS algoritmasını (14) den

$$w_s(t) - x_s(t)d_s < \frac{\alpha_s}{\gamma} d_s \quad veya \quad w_s(t) - x_s(t)d_s > \frac{\alpha_s}{\gamma} d$$

geliştirir.

Bu büyüklük α_s yi $1/\gamma$ kez daha büyük kullanmak için örneğin 10 KB/sn in birimi olarak kullanmak yerine KB/sn α_s için kullanmak. γ (α_s nin birimi) bütün kaynaklarda aynı olduğunu not edin. Daha küçük gama kaynak değerlerinin dönüşümünü kesinleştirir, ama dönüşüm daha yavaş olur. Kararlılık ve cevap verilebilirlilik arasındaki duyarlılık mevcuttur herhangi bir geri dönüş kontrol sisteminde. Daha küçük γ aynı zamanda daha büyük eşitlik geri bildirim eşitliğine sebep olur $b_l(t) = c_l p_l(t)/\gamma$ olduğunda. Bu zorluk ücret hesaplamadan tampon işlemi tekleştirmekle işaretleme tanıtılarak üstesinden gelinebilir.

α_s kullanmak için bu değerler $1/\gamma$kere daha büyük 10 KB/sn birim kullanır alfa s için KB/sn ın dışında γ (α_s nin birimi) bütün kaynaklarla aynı olmalıdır. Daha küçük y değeri kaynak değerlerinin dönüşümüdür ama dönüşüm daha küçük olur. Kararlılık ile cevap verilebilirliği arasında gerilme vardır herhangi bir geri dönüş sisteminde. Daha küçük γ aynı zamanda daha büyük ger bildirimine sebep olur. $b_l(t) = c_l p_l(t)/\gamma$ olduğunda. Bu zorluk fiyat hesaplamadan tampon işlemi çiftleştirmeme için işaretleme tanımlanarak aşılabilir.

5.2. Gecikme, Kayıpsız Tam ve Kayıp

5.2.1. Gecikme

Daha önceki bölümde VEGAS algoritmasının 2 eşitlik yaklaşımı tanımlanmıştır. İlki kaynağı değerini ayarlar böylelikle asıl değerini bulmak için α_s ile β_s KB/sn arasında tahmin edildiği değerinden daha küçüktür. α_s (1/ds tipik olarak) ve β_s (tipik olarak 3/ ds) VEGAS algoritmasının parametresi iken. Bu umulan değer maksimum mümkün olan hali hazırdaki pencere boyutudur. Yolda herhangi bir sorgulama yoksa gerçekleşir. Bu değerin mantığı maksimum ağ alt araçlandırmasına çok yakındır. Ve sıkışıklıktan daha uzaktır. 2. yaklaşım VEGAS kaynağı $\alpha_s d_s$ (tipik olarak 1) ile $\beta_s d_s$ (tipik olarak 3) arasında ki yolda tamponlanan paketlerin sayısını korumak için değerini ayarlar. Böylece ekstra kapasite avantajı sağlar mümkün olduğu zaman. Duality model 3. yaklaşımı tavsiye eder. Tamponlama işlemin dinamikleri l hattında (eşitlik (5.11) ve (5.13) karşılaştırarak) bir bağ kurar

$$p_l(t) = \frac{b_l(t)}{c_l} \tag{5.21}$$

Bu da bağlantı ücreti $p_l(t)$ hatlarında sorgu gecikmesinin t zamanındaki paket varışı ile yüzleşmesidir. Yol ücreti $p^s(t) = \sum_{l \in L(s)} p_l(t)$ uçtan uca sorgu gecikmesidir (dağınık gecikme hariç). Bu sıkışıklık sinyalidir ve kaynak değerini ayarlar ve kaynak tur zamanı ile (tahmin edilen) üretim gecikmesi arasındaki fark alınarak hesaplanır. Daha sonra eşitlik (5.12) VEGAS kaynak setine uygulanır. Sorgu gecikmesinin yayılımının gecikmesinin değeri orantısaldır. Orantısal sabit α_s ile β_s arasındadır. Ne kadar büyük sorgu gecikmesi olursa o kadar keskin sıkışıklık ve değer düşümü olur. Bu VEGAS ın yaklaşımı ren ile birlikte kullanıldığında VEGAS ı değiştirmek için kullanılır. Aşağıdaki bölümü görünüz. Aynı zamanda eşitlik (5.12) den takip eder ki bant genişliği sorgu gecikmesi genişliği kaynağın ürünü yolda tamponlanmış ekstra paketler.

$$x_s^* p^{*s} = \alpha_s d_s \tag{5.22}$$

Sorgu teorisinde Little in kanunudur. Eşitlik (5.22) ye ilişkin p^{*s} ye uygulanır. p^{*s} Artmalıdır. x_s^* kaynakların sayısıyla böylelikle azalmalıdır. Yolda tamponlanan bazı ekstra paketleri tutmak için her kaynağın kakıştığı yeniden oluşmasıdır bu.

5.2.2. Kayıpsız Tam

Buna rağmen bunu zamanda tanımadık. VEGAS ın iki eşit uygun uygulaması vardır. Her biri algoritmada belirsizliğin farklı yaklaşımlarındandır. İlki asıl koda dayanan alfa s ve beta s parametrelerini belirler her tur zamanı için byte olarak. İkincisi kaynak [9] deki yazısına dayanarak her inicin biten terimleri olarak belirler. Bu iki byte'ın (paketlerin) bu iki uygulama iyilik üzerinde kesin etkisi vardır: ikinci kayırma kaynakları geniş yayılan gecikmesiyle.

Bizim modellememizin terimlerinde "log" araç fonksiyonu (5.1) x^* eşitlik değeri iyi yayılır kaynak [7], [8] herhangi diğer uygun değer vektörü x için

$$\sum_s \alpha_s d_s \frac{x_s - x_s^*}{x^*} \le 0$$

e sahibiz. ilk uygulaması alfa $\alpha_s = \alpha / d_s$ kaynak yayılım gecikmesine ters olarak orantılıdır. daha sonra $U_s(x_s) = \alpha_s d_s \log x_s = \alpha \log x_s$ araç fonksiyonları bütün kaynaklar için uygundur ve eşitlik değeri iyi oranlıdır. ve yayılım gecikmesinin bağımsızlığıdır. Buna oransal iyilik (Pf) uygulaması deriz. 2. uygulama $\alpha_s = \alpha$ eşitliğine sahiptir bütün kaynaklar için. Daha sonra araç fonksiyonları ve eşitlik değerleri oransal iyi olur, kaynakların oransal ağırlığı ile yayılım gecikmeleri eşitlik (5.22) uygulanır ki eğer iki kaynak r ve s yüzleri yol fiyatı ile aynıysa örneğin tek sıkışık bağlantılı ağda eşitlik değeri yayılım gecikmesine orantılıdır.

$$\frac{x_r^*}{d_r} = \frac{x_s^*}{d_s}$$

Çoklu sıkışıklıklı bağlantılı ağda yayılım gecikmesi tarafından araca ağırlık verir. Eğer yayılma gecikmesi kaynak yolunda sıkışıklık bağlantıların sayısına orantılı ise ikinci uygulama olan ağarlıklı orantısal iyiliği çağırırız (WPF). Bunun pencereyi eşitlemeye çalışan TCP RENO ile uyumu

$$x_r^* D_r^* = x_s^* D_s^*$$

ve böylelikle bant genişliğinin yarısında kaynak 2 kere tur zamanı gecikmesi gönderir. Bu bağlantılar aleyhinde davranma yüksek yayılma gecikmesi ile kaynaklarda iyi bilinir örneğin kaynak [10,…,14] aslında aynı yöntem kayıp ihtimalini çiftli değişken olarak varsayarak burada geliştirilmiştir. [5; Low et al. 2002] RENO ya uygulanmıştır. RENO

$$U_s(x_s) = \frac{\sqrt{2}}{D_s} \tan^{-1} \frac{x_s D_s}{\sqrt{2}}_s$$

araç fonksiyonuna gösterildiği gibi sahiptir.

5.2.3. Kayıp

Eşitlik geri bildirimi $b_l^* = p_l^* c_l$ ile uygun hale getirmek için sağlanan L bağlantılarının tamponu yeterince büyüktür. VEGAS kaynağı eşitlik (5.5) deki uygunluk durumuna borçlu olan eşitlikte herhangi bir kayıptan dolayı etkilenmeyecektir. Paketler kaybolana dek pencere boyutunu doğrusal olarak artırmakla hangi pencerenin çarpılabilir azaltılmış olduğuna bağlı olarak ağın ek kapasitesi için araştıran bu TCP-RENO ya bir ayarlamadır. Belirlenmiş tur zamanı ve akıllıca ona reaksiyon göstermeyle bu da dikkatlice açılmış sıkışıklık bilgisi tarafındandır. VEGAS devamlı boşaltma döngüsünden ve sıkışıklıktan düzeltmeden kaçınır. Bu kaynak [1] de deneysel sonuçlardan teyit edilmiştir. Kaynak [1] ve [14] da belirtildiği gibi

eğer tampon yeterince geniş değilse eşitlik ulaşılamaz kayıp kaçınılmaz olur ve VEGAS RENO döner. Bunun sebebi eşitliğe ulaşma çabasındandır. VEGAS kaynaklarının hepsi kendi yollarında tamponu ağ da artırarak $\alpha_s d_s$ paketlerin sayısına ulaşmaya çalışır. Bu aklın kabul edebileceği şekilde VEGAS ın RENO üzerindeki performans geliştirmeleri detaylı deney çalışmaları göreli çeşitli mekanizmalar ile belirlemesini açıklar. Bu çalışma belirlemiştir ki VEGAS ın kayıp düzeltme mekanizması sıkışıklı giderme mekanizması değildir. Çok büyük bir yardım sağlamıştır. Bu açıkça eşitliğe ulaşmaktan VEGAS korumak için olabildiğince küçük tampon olursa umulmalıdır. [Hengartner et al. 2000] de yönlendirici tampon büyüklü 10 parçadır: yönlendirici tampon büyüklüğü 19 parçadır, arka taraf trafiği ile kolayca doldurulabilir. VEGAS geri bildirimi için küçük boşluk bırakılabilir. Tampon büyüklüğünün etkisi ve VEGAS ın yeniden gönderimi aşağıda benzeterek gösterilmiştir.

5.3. Daimi Sıkışıklık:

Bu bölüm daimi sıkışıklık problemini açıklamaktadır. Hem VEGASın tampon işleminin sömürücüsü fiyat belirlemek için hem de yayılma gecikmesini tahmin etmek için ihtiyaçlarını açıklar. Gelecek bölümde rasgele logaritmik işaretleme (REM) tarafından nasıl meydana geldiği açıklanacak kaynak [15]. ECN kaynak [16; 17]da daha önceki açıklandığı formda gibi.

5.3.1. Fiyat ve Geri Bildirimin Eşitlenmesi

VEGAS tampon işleminde güvenir $p_l(t) = b_l(t)/c_l$ fiyatını hesaplamak için eşitlik fiyatları sıkışıklık kontrol algoritmasına dayanmaz ama problem durumu yalnızca: ağ topolojisi, bağlantı kapasitesi, kaynakların sayısı, onların yönlendirmeler iş ve araç fonksiyonları gibi. Kaynakların sayısı artarsa eşitlik ücreti ve geri bildirim $(b_l^* = p_l^* c_l)$ yap. Eğer her kaynak $\alpha_s d_s = \alpha$ paketlerini ağda tamponlanmış olarak tutarsa eşitlik back logu alfa n paketleri olur, kaynağın n numarası lineerken

5.3.2. Yayılma Gecikmesi Tahmini

Modelimizde iddia ediyoruz ki kaynak d_s tur zamanı yayılma gecikmesini bilir pratikte minimum tur zamanı değerini belirler daha sonradan anlaşılacağı gibi. Hata yönlendirme değişikliği olduğu zaman oluşabilir veya yeni bir bağlantı başladığında kaynak [3]. İlk olarak yönlendirme daha uzun yayılma gecikmesinden hali hazırdaki yönlendirmeye değiştiğinde, kaynak onu büyültmek zorunda olduğunda yeni yayılma gecikmesi büyümüş tur zamanı olarak alınır. Daha sonra penceresini küçültür.

İkinci olarak kaynak başladığında tur zamanını belirler sorgu gecikmesi yolundaki paketlere dayanarak sorgu gecikmesi dâhilinde. d_s yayılma gecikmesini yüksek tahmin eder ve $\alpha_s d_s$ paketlerinden daha fazla yolda koymaya çalışır. Daimi sıkışıklığa önderlik eder. Kararlılık ve iyilikten yaklaşık hataların etkisini göreceğiz.

Her s kaynağı yaklaşık $\hat{d}_s(t) := (1+\varepsilon_s) d_s(t)$kendisinin tur zamanı yayılma gecikmesi d_s 1 deki VEGAS algoritmasında olduğunu düşünün. ε_s farklı kaynaklar için farklı olabilen yüzde hata. Doğal olarak $-1 < \varepsilon_s \le D_s(t)/d_s(t) - 1$ bütün t ler için iddea ederiz ki yaklaşım $0 < d_s(t) \le D_s(t)$ sağlar. Eşitlik pencereleri $w^* = (w_s^*, s \in S)$ ve ilgili eşitlik tur zamanı $D^* = (D_s^*, s \in S)$

$$\frac{w_s^*}{\hat{d}_s} - \frac{w_s^*}{D_s^*} = \alpha_s \quad bürün\ s \in S\ ler\ için \tag{5.23}$$

doğrular. Bir sonraki sonuç söyler ki yaklaşım hatası (3) den

$$U_s(x_s) = (1+\varepsilon_s)\alpha_s d_s \log x_s + \varepsilon_s d_s x_s \tag{5.24}$$

araç fonksiyonunu etkili bir şekilde değiştirir.

*Teorem 5.1***:** Eş yayılma gecikme tahmininde hata yüzdesi olsun. $w^* = \left(w_s^*, s \in S\right)$ VEGASaın eşitlik penceresi olsun ve $D^* = \left(D_s^*, s \in S\right)$ ilgili eşitlik tur zamanı olsun örneğin eşitlik (5.23) ü sağlasın. İlk giren ilk çıkar prensibine göre bütün hatlarda paketlerin sunulmuş olduğunu düşünün. Daha sonra eşitlik kaynağı değeri $x^* = \left(x_s^*, s \in S\right)$ $x^* = w_s^* / D_s^*$ tarafından belirlenmiştir. bu eşitlik (5.24) de verilen araç fonksiyonu ile eşitlik (5.4) ve (5.5) in en uygun çözümüdür

İspat: teorem 2.1 in ispatına dayanır

$$U_s'(x_s^*) = \frac{(1+\varepsilon_s)\alpha_s d_s}{x_s^*} + \varepsilon_s d_s = \sum_{l \in L(s)} p_l^* \tag{5.25}$$

Eşitlik (5.6) (5.25) eşitliğiyle değiştirildiğini düşünün. Bağlantılarda eşitlik geri bildirimi *p**, gibi bir vektörü sağladığını göstermek için ve böylelikle eşitlik değerleri uygundur. Yaklaşık yayılma gecikmesini $\hat{d}_s^* = (1+\varepsilon_s)d_s^*$ d_s^*ın eşitlik (5.23) deki doğru değerlerinde

$$\alpha_s = \frac{w_s^*}{(1+\varepsilon_s)d_s} - \frac{w_s^*}{D_s^*}$$

elde etmek için düşünün. $w_s^* - x_s^* d_s = x_s^* \sum_{l \in L(s)} b_l^* / c_l$ Kullanarak

$$(1+\varepsilon_s)\alpha_s d_s = (w_s^* - d_s x_s^*) - \varepsilon_s d_s x_s^* = \left(\sum_{l \in L(s)} \frac{p_l^*}{c_l} - \varepsilon_s d_s \right) x_s^*$$

sahip oluruz.

bu eşitlik (5.25) i kabul eder $p_l^* = \frac{b_l^*}{c_l}$ ü belirleyerek ve terimleri yeniden düzenleyerek.

Teorem 2.1 in ispatında *x* * uygun olmalıdır ve sınırsız geri bildirimin dışında büyüyebilmelidir. Tamamlayıcı gevşeklik durumunu yalanladığı doğrulanmalıdır. Böylelikle ispat tamamlanır.

Teorem 5.1 anlamı iki parçalıdır. İlki yanlış yayılma gecikmesi uygular ve VEGAS algoritmasının kararlılığını düzenini bozmaz. Değerler basit olarak değişik eşitlik peşine gider. İkincisi yayılma gecikmesi yaklaşımında nispi hatayı bildiğimiz zaman yeni eşitlik değerleri hesaplamamıza izin verir ve böylelikle iyiliği tayin eder. Bu tip bilgiler elde olmadığında kaliteli hata yaklaşımın etkisinin kıymetini belirler

Örneğin r ve s kaynağının aynı yol ücretinde olduğunu gördüğünüz düşünün. Eğer 0 yaklaşım hatası varsa onların eşitlik değeri ağırlıkları ile orantılıdır.

$$\frac{a_r d_r}{x_r^*} = \frac{a_s d_s}{x_s^*}$$

hata ile onların değeri eşitlik (5.26) de anlatılmıştır.

$$\frac{(1+\varepsilon_s)\alpha_r d_r}{x_r^*} = \varepsilon_r d_r = \frac{(1+\varepsilon_s)\alpha_s d_s}{x_s^*} + \varepsilon_s d_s \tag{5.26}$$

Böylelikle geniş pozitif hata genellikle daha yüksek eşitlik değerine yönlenir. Diğer kaynakların zarar görmesi için. PF uygulama için $a_r d_r = a_s d_s$ olduğu zaman eğer kaynaklar aynı mutlak hataya sahipse $\varepsilon_r d_r = \varepsilon_s d_s$ kayan değerleri $1+\varepsilon_s$ ye orantılıdır.

Bununla birlikte VEGAS yayılma gecikmesi tahmininde hatanın varlığında kararlı olabilir. Hata 2 probleme sebep olabilir. İlki aşırı tahmin eşitlik kaynak değerini yükseltir. Bu fiyatları yukarı çeker ve hatta tampon geri bildirimlerinde kalıcı sıkışıklıklara yöneltir. İkincisi hata kaynaklarının araç fonksiyonlarını bozar. Yeni kaynakların tarafını tutarak iyi olmayan a eşitliklerine götürür. Bunu basit bir örnekle gösteririz örnek aynı zamanda teorem 4.1 in uygulamasını da kapsar.

Örnek: Kalıcı sıkışıklık

C pkts/ms kapasitesi ile bir bağlantı düşünün ve n kaynakları tarafından sonsuz tampon paylaşılmış olsun. Hepsi *d* ms ortak tur yayılma gecikmesi ve α pkts/ms ni düşünün. Bu da bütün kaynaklar yayılma gecikmesini açıkça bilirse her biri αd pkts yi eşitliğini kendi yolunda tutar.

Şimdi başarılı olarak kaynağın aktif olduğunu düşünün. *t* kaynağı, $t, t = 1,...,N$ t periyodunun başlangıcında kaynaklar $1,...,t-1$ eşitliğe ulaştıktan sonra aktif hale gelir. Daha sonra t kaynağının yaklaşık yayılma gecikmesi sorgu gecikmesi *p(t)* yi $1,...,t-1$ kaynaklarına bağlı olarak ihtiva eder. L periyodunda sadece 1 kaynağı aktiftir böylece 1 kaynağı yayılma gecikmesini doğrulukla tahmin eder. $d_1 = d_s$ ve $p(1) = \alpha d / c$ in sorgu gecikmesini üretir. Kaynak 2 tarafından yayılma gecikmesinin tahmininde bu bir hatadır.

örneğin $d_2 = d + p(1)$ $x_1^* + x_2^* = c$ olduğunda periyot 2 de 1 ve 2 kaynaklar *p(2)* nin eşitlik sorgu gecikmesini üretir ki teorem 4.1 in ispatı olarak eşitlik (5.25) i gerçekler.

$$\frac{\alpha d}{p(2)} + \frac{\alpha(d + p(1))}{p(2) - p(1)} = c$$

e sahip oluruz. Sonuç olarak eşitlik sorgu gecikmeleri başarılı periyotlarda

$$\frac{\alpha d}{p(t)} + \frac{\alpha(d + p(1))}{p(t) - p_1} + \frac{\alpha(d + p(2))}{p(t) - p(2)} + \cdots + \frac{\alpha(d + p(t-1))}{p(t) - p(t-1)} = c, \quad t = 2,\cdots,N \tag{5.27}$$

$$p(1) = \frac{\alpha d}{c} \tag{5.28}$$

Tarafından ters olarak hesaplanabilir. T periyodunda daha sonra eşitlik sorgu uzunluğu *cp(t)* pkts. dir. Eşitlik değeri $x_n(t)$ n kaynağı için t periyodunda $n = 1,...,t$ (5.25) nolu eşitlikten verilir.

$$x_n(t) = \frac{\alpha(d + p(n-1))}{p(t) - p(n-1)} \tag{5.29}$$

5.3.3. Sonuç

Kalıcı sıkışıklığı VEGAS ın orijinal benzetiminde görmedik. 3 faktörden dolayı olabilir. İlki ve en önemlisi ağda uygun olmayan tampon kapasitesi olduğunda VEGAS ın orijinal uygulaması RENO ya dönmüştür. İkinci olarak bizim benzetiminde düşünüldüğü gibi yönlendirme değişiminin olasılığını alamamıştır ama diğer taraftan ispat kaynak [18] deneyindeki problemde olduğu gibi yönlendirme değişimini ileri sürer. Son olarak bağlantının durumu seri olarak başlar. Pratikte bağlantıların devamı gelir ve gider böylece bütün kaynaklar yayılma gecikmesi + ortalama sorgu gecikmesini gösteren temel RTT değerini ölçebilir.

5.4. REM'le VEGAS

Kalıcı sıkışıklık, kaynaklara sıkışıklığı taşıyan vazgeçilemeyen geri bildirim sıkışıklık ölçümü olarak sorgu gecikmesine güvenen VEGAS'ın sonucudur. Bu bölüm [5] de tanıtılan REM (Rasgele logaritmik işaretleme) nin nasıl olduğunu REM in bu durumu düzeltmede kullanılabileceğini gösterir. Bizim hedefimiz VEGAS ın eşitlik değeri tahsisi en son bölümde tasvir edilen kalıcı sıkışıklık tehlikesi olmaksızın korumaktır. Bu log araç fonksiyonunu da korur ve VEGAS ın orantısal iyiliğini de korur. Bölüm 2.1 de anlatılan VEGAS ın bu 3 açıklamasını çağıralım. Eşitlik değer tahsisini korumak için 3. yaklaşımı kullanıyoruz ki VEGAS kaynağı eşitlik (5.12) de belirtildiği gibi Yol ücretinin Tur zamanı gecikmesi olmadığı durumlarda yayılma gecikmesi oranının yol ücretine orantılı olmasını sağlar. Bunun dışında REM algoritması kullanarak geri beslenmiştir ve hesaplanmıştır, aşağıda açıklanmıştır. AQM nin amacı tampon aşımından dolayı olası düşmeler ve işaretlemeler tarafından kayıp sinyalleri yerine koymak değildir. Ama amacı yol ücretini geri besleme olabilir.

REM de amacımızı gerçekleyen iki fikir vardır. İlki REM temiz tampon ve değeri seçmeye çabalar ve yüksek yararlanma ve düşük sorguya yöneltir. Küçük sorular ile minimum tur zamanı yayılma gecikmesi için en uygun yaklaşım olabilir. Bununla birlikte tur zamanı kaynağa fiyat bilgilerini daha fazla iletmez. REM in ikinci fikri belirlenen düşme veya işaretleme değerlerinden onların yol ücretlerini kaynağın tahmin etmesine izin verir. REM i şimdi özetleyebiliriz; kaynak [5] in tasarım mantığı, performans değeri ve parametre ayarlamaları için görün.

Her l hattı $p_t(t)$ yi günceller t periyodunda. Toplam giriş değeri $x^l(t)$ ve tampon işgali l hattında $b_l(t)$ ye dayanarak

$$p_t(t+1) = \left[p_t(t) + \gamma(\mu_l b_l(t) + x^l(t) - c_l)\right]^+ \tag{5.30}$$

$\gamma > 0$ ve $0 < \mu_l < 1.^2$ olduğunda y parametresi değerin dönüşümünü değerin geri bildirimi nu kontrol eder. Böylece $p_l(t)$ artırılmıştır ağarlıklı $b_l(t)$ geri bildirimi nun toplamı ve değerdeki hatalı seçimler $x^l(t) - c_l$, m tarafından ağırlaştırılmıştır,
Pozitiftir ve diğer durumlarda düşürülmüştür. Eşitlikte bu ağırlıklı toplam sıfırdır.(eşitlik ücreti $p^{*l} > 0$ olan sıkışıklık bağlantısında). $b_l^* = 0$ ve $x^{*l} = c_l$ uygulanılan nokta. Bundan

dolayı eğer $x^{*l} \neq c_l$ se sorgu uzunluğu b_l^* eşitlik içinde olmaz böylece $x^{*l} = c_l$ dir. Bu $b_l^* = 0$ olmasını sağlar ağırlıklı toplam 0 olduğu zaman. Bu özellik her yüksek yararlanma ve düşük kayıpta gecikmeye sebep olur.

KAYNAKLAR

[1]. International Zurich Seminar on Broadband Communications. 163-170. BRAKMO, L. S. AND PETERSON, L. L. 1995. TCP Vegas: end to end congestion avoidance on a global Internet.

[2]. BOUTREMANS, C. AND BOUDEC, J. Y L. 2000. A note on the fairness of tcp vegas. In Proceedings ofInternational Zurich Seminar on Broadband Communications. 163-170.

[3]. Mo, J., LA, R., ANANTHARAM, V., AND WALRAND, J. 1999. Analysis and comparison of TCP Reno andVegas. In Proceedings of IEEE Infocom

[4]. LOW, S. H. AND LAPSLEY, D. E. 1999. Optimization flow control, I: basic algorithm and convergence.IEEE/ACM Transactions on Networking 7, 6 (December), 861-874.

[5]. ATHURALIYA, S. AND Low, S. H. 2000a. Optimization flow control, II: Implementation. Submitted for publication, http: //netlab.caltech.edu. ATHURALIYA, S. AND Low, S. H. 2000b. Optimization flow control with Newton-like algorithm. Journal of Telecommunication Systems 15, 3/4, 345-358. BERTSEKAS, D. 1995. Nonlinear Programming.

[6]. BERTSEKAS, D. 1995. Nonlinear Programming thena. Athena Scientific.

[7]. KELLY, F. P. 1997. Charging and rate control for elastic traffic. European Transactions on Telecommunications 8,33-37.

[8]. KELLY, F. P., MAULLOO, A., AND Tan, D. 1998. Rate control for communication networks: Shadow prices,

[9]. BRAKMO, L. S. AND PETERSON, L. L. 1995. TCP Vegas: end to end congestion avoidance on a g9lobal Internet. IEEE Journal on Selected Areas in Communications 13, 8 (October), 1465-80 proportional fairness and stability. Journal of Operations Research Society 49, 3 (March), 237-252.

[10]. FLOYD, S. 1991. Connections with multiple congested gateways in packet-switched networks, Parti: one-way traffic. Computer Communications Review 21, 5 (October)

[11]. FLOYD, S. AND JACOBSON, V 1993. Random early detection gateways for congestion avoidance. IEEE/ACM Trans, on Networking 1, 4 (August), 397^113

[12]. LAKSHMAN, T. V AND MADHOW, U. 1997. The performance of TCP/IP for networks with high bandwidth-delay products and random loss. IEEE/ACM Transactions on Networking 5, 3 (June), 336-350

[13]. MATHIS, M., SEMKE, J., MAHDAVI, J., AND Ott, T. 1997. The macroscopic behavior of the TCP congestion avoidance algorithm. ACM Computer Communication Review 27, 3 (July)

[14]. BONALD, T. 1998. Comparison of TCP Reno and TCP Vegas via fluid approximation. In Workshop on the Modeling of TCP.

[15]. ATHURALIYA, S., Li, V. H., LOW, S. H., AND YIN, Q. 2001. REM: active queue management. IEEE Network. Extended version in Proceedings qflTCl 7, Salvador, Brazil, September 2001

[16]. FLOYD, S. 1994. TCP and Explicit Congestion Notification. ACM Computer Communication Review 24, 5(October)

[17]. RAMAKRISHNAN, K. K. AND FLOYD, S. 1999. A Proposal to add Explicit Congestion Notification (ECN) to IP. RFC 2481

[18]. PAXSON, V 1996. End-to-end routing behavior in the Internet. In Proceedings of SIGCOMM'96

BÖLÜM 6

KARARLI VEGAS

Mevcut TCP VEGAS algoritmasının ağ gecikmesi varlığında kararsız hale gelebileceğini gösteriyor ve bunu kararlı hale getirecek bir modifikasyon öneriyoruz. Kararlı VEGAS algoritması tümüyle kaynak-tabanlı kalmaya devam etmekte ve herhangi bir ağ desteğine gereksinim duymaksızın uygulanabilmektedir. Burada, ağ bazıları aktif dizi yönetimli olan, bazıları olmayan bir hatlar karışımından meydana geldiğinde kararlı VEGAS algoritması için adım adım bir derleme stratejisini ortaya koyuyoruz.

İlk olarak, VEGAS ve RENO arasındaki performans farkını karşılaştırmak için yapılmış kaynak [1], [2], [3] gibi kapsamlı deneylerin sonuçları. VEGAS' ın işleyişi ve uygun özellikleriyle ilgili olarak kaynak [4], [5] ve [6]'te de çalışılmıştır; ancak bu yazılar yalnızca bir boğum hattını ele almakta ve işleyişe ilişkin bu çalışmada ağ gecikmesi hesaba katılmamaktadır. Optimizasyon tabanlı modeller kaynak [7] ve [8]'de genel bir VEGAS ağının analizi için kullanılmaktadır. Özel olarak, kaynak [8] ve [9]'te herhangi bir TCP/AQM (Active Queue Management- Aktif kuyruk yönetimi) protokolünün toplam faydayı maksimize etmek için Internet üzerinde dağıtılmış bir birincil-ikili algoritmanın icrası olarak yorumlanabileceği gösterilmektedir. Aynı zamanda kullanıcı faydasının ise TCP algoritması ile tanımlandığı çoğunlukla ve açıkça gösterilmektedir. Kaynak [7], [10], [11], [12], [13]. Bu modeller çoğu zaman denge yapısı üzerine odaklanır ve ağ gecikmesini uygun bir şekilde ele almazlar. Bu iş dizisini tamamlamak için, burada bölüm 6.1 de tanımlanan, VEGAS'ın bir denge çevresindeki doğrusal kararlılığını analiz etmek için açık bir şekilde ileri ve geri tutarsız gecikmeler içeren çok-hatlarda çok-kaynaklı bir model kullanıyoruz. Önceki analitik çalışmayla karşılaştıracak olursak, gecikmenin VEGAS üzerindeki etkisini ve bunu nasıl kararlı hale getireceğimizi anlamak için küresel doğrusal olmayan işleyişi feda ediyoruz.

İkinci olarak, bu metin en son kaynak [14], [15], [16], [17] ve [18]'de geliştirilen doğrusal dağılımlı ve gecikmeli sistemin kararlılığı kuramından güç almaktadır. Özel olarak, bir TCP/AQM algoritması doğrusal kararlılık için keyfi ağ gecikmeleri ve kapasitelerini oluşturan [14]'de tasarlanmaktadır. Bu algoritma statik kaynak algoritmalarını kullanan kaynak [11] ve [19]'teki "ikilik" algoritmalar sınıfında yer alır ve kararlılığa taviz vermeksizin yüksek fayda ve hızlı tepkiyi sağlamak için ağ gecikmeleri ve kapasitelerinin karmaşık ölçeklemesini devreye sokar. Bununla birlikte, keyfi olarak ölçeklenebilen bu kararlılık formu özel bir kaynak kullanım fonksiyonunu ve böylece de oran tahsisinde özel bir uygunluğu dikte eder. Kaynak [14]'deki TCP/AQM, ağ gecikmeleri üzerinde bilinen bir sınırlama olması koşuluyla, yavaş bir sure ölçeği üzerindeki herhangi bir fayda fonksiyonu ya da uygunluğu takip etmek için kaynak algoritmasına daha yavaş bir sure ölçeği dâhil ederek kaynak [17]'dekine genişletilmektedir.

Bu iş dizisinin ana anlayışı kontrol altındaki geri besleme döngüsü üzerindeki kazancı sürdürmek için kaynak cevaplarını gidiş-dönüş süresiyle ve hat cevaplarını kapasiteleriyle azaltmaktır. Bu da ortaya koyar ki, VEGAS hat algoritması kaynak [14] ve [17]'de kullanılan kapasiteyle ilgili kesinlikle doğru bir ölçeklemeye sahiptir Kaynak [8]. Kapasiteyle ilgili bu

yerleşik ölçekleme, RENO ve varyantlarının tam tersine, potansiyel olarak VEGAS'ın yüksek bant genişliğine göre ölçeklenebilir olmasını sağlamaktadır. Bununla birlikte, VEGAS'ın kaynak algoritması gecikme söz konusu olduğunda kaynak [14] ve [17]'dekilerden farklı bir ölçeklemeye sahiptir. Bölüm 6.3 VEGAS'ın büyük gecikmelerde kararsız hale gelebileceğini gösteren uygun kararlılık koşulunu ortaya koyuyor. Bölüm 6.4 de bunu kararlı hale getirmek için küçük bir modifikasyon öneriyoruz. Bölüm 6.5 de ise, VEGAS kaynaklarının bir kuyruk-temizleme AQM algoritması kullanan ya da kullanmayan bir yönlendiriciler karışımı ile çalışmasına izin veren aşamalı bir derleme stratejisini tanımlıyoruz. En nihayet bölüm 6.6 da VEGAS'ın dinamiği ile kararlı versiyonunu karşılaştıran benzetim sonuçlarını sunuyoruz.

RENO ve varyantlarının tersine, VEGAS'ın özellikle yüksek hızlı ağlarla uyum sağladığı görülmektedir. RENO ve varyantlarının ağ kapasitesi arttıkça kararsız hale geldikleri görülmektedir kaynak [20], [21]. RENO aynı zamanda dengede kontrol için güvenilir kullanımı güç olan aşırı derecede küçük bir kayıp olasılığı oluşturmak zorundadır. Diğer taraftan, VEGAS kapasiteyi doğru olarak ölçeklemektedir. Dahası, denge kuyruk gecikmesi düşük kapasitede aşırı olabilirken kapasite arttıkça düşürülmektedir. Kuyruklar ve yeniden-yönlendirme nedeniyle yayılma gecikme tahminindeki kaynak [5], [8] hata gibi diğer sorunlar yüksek kapasitede arabellekler daha sık temizlendiğinden daha az sert olabilmektedir.

6.1. Ağ Modeli

Bir ağ sonlu *c = (cl, l G L)* kapasiteye sahip L sayıda hat seti kısıtlı kaynak olarak modellenir. Bunlar r ile indekslenen N sayıda kaynaktan oluşan bir set tarafından paylaşılır. Her r kaynağı L x N yönlendirme matrisi ile tanımlanan bir hat setini kullanır.

$$R_{lr} = \begin{cases} 1 & eger\, l\, linkinde\, r\, kaynakları\, kullanılırsa \\ 0 & dıger\, durumlarda \end{cases}$$

Her bir hatta karşılık gelen 1 değeri daha sonra "fiyat-price" olarak adlandıracağımız bir $p_l(t)$ yoğunluk ölçüsüdür. Aşağıda göreceğimiz gibi, $p_l(t)$ hat 1'deki ölçeklenmiş kuyruk gecikmesidir. Her bir r kaynağı paket/sn olarak bir $x_r(t)$ oranı oluşturur. Biz bu metinde esas olarak bir denge çevresinde doğrusal model ile ilgileniyoruz; bu yüzden r kaynağından hat 1'e doğru ileri yönlü gecikme dengesini $\tau_{\bar{l}r}$ ile ve hat 1'den r kaynağına geri yönlü gecikme

$$q_r(t) := \sum_l R_{lr} p_l(t - \tau_{\bar{l}r}) \tag{6.1}$$

ve hat 1'in toplam kaynak oranını gözlediğini varsayıyoruz.

$$y_l(t) := \sum_r R_{lr} x_r(t - \tau_{\bar{l}r}) \tag{6.2}$$

dengesini ise $\tau_{\bar{l}r}$ ile simgeliyoruz. t süresinde r kaynağının kendi yolunda toplam fiyatı gözlemlediğini Tr denge gidiş-dönüş süresini temsil etsin.

$$\tau_{\bar{l}r} + \tau_{\bar{l}r} = T_r, \qquad \forall l \in L$$

Olduğunu var sayıyoruz. O halde kaynak [8] TCP VEGAS'ı ilgili kuyruk yönetimi ile birlikte, aşağıdaki dinamik sistem olarak modeller:

$$\dot{p}_l(t) = \begin{cases} \frac{1}{c_l}(y_l(t) - c_l) \quad eger \;\; p_l(t) > 0 \\ \frac{1}{c_l}(y_l(t) - c_l)^+ eger \;\; p_l(t) = 0 \end{cases} \tag{6.3}$$

$$\dot{x}_r(t) = \frac{1}{T_r^2(t)s} gn\left(1 - \frac{x_r(t)q_r(t)}{\alpha_r d_r}\right) \tag{6.4}$$

burada, eğer z > 0 ise $(z)^+ = \max\{0, z\}\}, \text{sgn}(z) = 1,$, z < 0 ise -1 ve z = 0 ise 0'dır. α_r bir VEGAS protokol parametresi, d_r ise r kaynağının gidiş-dönüş yayılım gecikmesidir. $p_l(t)$ fiyatı hat 1'deki kuyruk gecikmesini ve $q_r(t)$ r kaynağının uçtan-uca kuyruk gecikmesini gösterir Kaynak [8]. Bölüm 6.3 de tanımlanan r kaynağının gidiş-dönüş süresi T_r denge değeri ile, $T_r(t) := d_r + q_r(t)$ olarak tanımlanır.

VEGAS algoritmasının bir yorumlanma biçimi, her r kaynağının kendi yolundaki kuyruklarda arabelleğe alınan paketlerin $\alpha_r d_r$ sayısını oluşturmak için kendi oranını ya da pencereyi ayarladığı şeklindedir. Hat algoritması eşitlik (6.3) arabellek süreci tarafından otomatik olarak yürütülür. Kaynak algoritması eşitlik (6.4) hatlarda ara belleğe alınan paketlerin $x_r(t)q_r(t)$ sayısının $\alpha_r d_r$'den küçük ya da büyük olmasına göre pencereyi her gidiş-dönüş süresinde 1 adet artırır ya da azaltır. Denge durumunda $x_r^* q_r^* = \alpha_r d_r$ ve özel denge oranlarında $x^* := (x_r^*, r = 1,...., N)$ hat kapasitesi sınırlamalarına konu olan $\sum_r U_r(x_r)$ toplam faydayı

$$U_r(x_r) = \alpha_r d_r \log x_r$$

fayda fonksiyonları ile maksimize eder kaynak [8] de olduğu gibi. Bu nedenle VEGAS tartılı oransallık uygunluğu hedefine ulaşır kaynak [10],[14] ve [17]'nin hat algoritması VEGAS hat algoritması eşitlik (6.3) ile benzerdir. Tek fark, burada c'nin dengedeki kuyruğu temizlemek için gereken gerçek hat kapasitesinden tam anlamıyla küçük olan sanal kapasiteyi göstermesidir. Burada $p_l(t)$ aynı girdi tarafından beslenen, ancak sanal bir kapasiteye yönlenen bir hattaki "sanal" kuyruk gecikmesi olarak yorumlanabilir. Kaynak [14] ve [17]'de gösterildiği gibi, $\dot{p}_l$'yı $1/c_l$ ölçeğinde küçültmek, sanal ya da gerçek, gecikmeye ağ kapasitesine bağlı olarak gerçek ölçeklemeyi sağlayacak olan şeydir. Kararlı VEGAS'ın hem sanal ve hem de gerçek kuyruk gecikmelerine genelleştirilebilecek şekilde hatlar karışımından oluşan bir ağ içinde kademeli olarak nasıl derlenebileceğini açıklayacağız.

6.2. VEGAS'ın Kararlılığı

VEGAS kaynak algoritması eşitlik (6.4) süreksizdir. Bu durum denge çevresinde yalpalanmaya neden olabilir. Orijinal VEGAS algoritması denge noktasını bir sete genişleterek yalpalanmanın önüne geçer. Hatlarda arabelleğe alınan paketlerin sayısı $x_r(t)q_r(t)$ $[\alpha_r d_r, \beta_r d_r]$ ile $\alpha_r < \beta_r$ $x_r(t)$ aralığında olduğu sürece kaynak oranı ya da pencere değişmeden kalır. Bununla birlikte, $\alpha_r < \beta_r$ kaynak [6] koşulu ile uygunluğun kontrolü zordur. Burada, kaynak [8]'de olduğu gibi $\alpha_r = \beta_r$ olarak düşünüyoruz.

Bu kesimde, VEGAS algoritmasının eşitlik (6.4) bir sürekli yaklaşımını sunuyor ve bu yaklaşıma dayanan yeterli bir doğrusal kararlılık koşulunu türetiyoruz. Bu koşul, gecikme arttığında bir VEGAS ağının kararsız hale gelebileceğini ortaya koymaktadır. Sonraki kesimde ise süreksizliğe bağlı yalpalanmanın önüne geçmek için gerçekte bu sürekli fonksiyonun (kararlı bir versiyonunun) eşitlik (6.4) yerine kullanılmasını önereceğiz. Yaklaşıklık Modeli:

$$\mathrm{sgn}(z) \cong \frac{2}{\pi}\tan^{-1}(\eta z)$$

olduğuna dikkat edelim. Limit $\eta \to \infty$ 'a giderken yaklaşıklık kesin haline gelir. Bu nedenle, yine $T_r(t) = d_r + q_r(t)$ iken eşitlik (6.4)'ün aşağıdaki yaklaşıklığını ele alalım:

$$\begin{aligned} \dot{x}_r(t) : &= f_r(x_r(t), q_r(t)) \\ &= \frac{2}{\pi}\frac{1}{T_r^2(t)}\tan^{-1}\eta\left(1 - \frac{x_r(t)q_r(t)}{\alpha_r d_r}\right) \end{aligned} \tag{6.5}$$

(x^*, p^*) denge noktasını düşünelim. Logaritmik fayda fonksiyonu tam anlamıyla konkav olduğundan x^* oranları benzersizdir. Yönlendirme matrisi R'nin full-rank olduğunu varsayalım, o halde denge fiyatları da $p^* = (p_l^*, l = 1,...,L)$ benzersiz olacaktır. Dahası, modelde yalnızca boğum hatlarının içirildiğini ve öyle ki, tüm l değerleri için $p_r^* > 0$olduğunu düşünelim. Denge durumunda, kaynak oranı x_r^* ve toplam fiyat q_r^*

$$x_r^* q_r^* = \alpha_r d_r$$

koşulunu sağlarlar. Denge noktası çevresindeki doğrusallaştırma

$$\begin{cases} x_r(t) = x_r^* + \delta x_r(t) \\ q_r(t) = q_r^* + \delta q_r(t) \end{cases}$$

O halde birinci derece için burada,

$$\dot{x}_r = \delta\dot{x}_r = \left.\frac{\partial f_r}{\partial x_r}\right|_* \delta x_r(t) + \left.\frac{\partial f_r}{\partial x_r}\right|_* \delta q_r(t)$$

ve $T_r = d_r + q_r^*$. Bu nedenle, Laplace domain'inde

$$\delta x_r(s) = -\frac{x_r^*}{q_r^s}\frac{a_r}{sT_r + a_r}\delta q_r(s) \tag{6.6}$$

$$a_r = \frac{2\eta}{\pi}\frac{1}{x_r^* T_r} \tag{6.7}$$

Hatlarda, (y_l^*, p_l^*) denge noktaları $y_r^* = c_l$ koşulunu sağlarlar. Hat algoritmasını eşitlik (6.3)

$$\begin{cases} y_l(t) = y_l^* + \delta y_l(t) \\ p_l(t) = p_l^* + \delta p_l(t) \end{cases}$$

dengesi çevresinde doğrusallaştırarak birinci derece için

$$\delta p_l = \left.\frac{\partial g_l}{\partial p_l}\right|_* \delta p_l(t) + \left.\frac{\partial g_l}{\partial y_l}\right|_* \delta y_l(t)$$
$$= \frac{1}{c_l}\delta y_l(t)$$

ve bunun Laplace dönüşümü olan

$$\delta p_l(s) = \frac{1}{c_l s}\delta y_l(s) \tag{6.8}$$

eşitliğini elde ederiz.
Özet olarak, VEGAS'ın doğrusallaştırılmış modeli eşitlik (6.6), (6.7) ve (6.8)'de tanımlanmaktadır. İşaretlemeyi basitleştirmek için metnin devamındaki genelleştirmede herhangi bir kayba neden olmaksızın tüm kaynakların aynı hedef kuyruk uzunluğuna sahip olduklarını, yani tüm 9r değerleri için $\alpha_r d_r = \alpha$ olduğunu var sayıyoruz aksi halde, a'yı takip eden kararlılık sonuçlarındaki $\alpha_r d_r$'yı minimum olacak şekilde alın.

6.2.1. Kararlılık

Kaynak [14]'i takip ederek, eşitlik (6.1)-(6.2) hata eşitliklerini matris formu içinde Laplace şeklinde

$$\delta y(s) = R(s)\delta x(s) \tag{6.9}$$
$$\delta q(s) = dieg\left\{e^{-sT_r}\right\}R^T(-s)\delta p(s) \tag{6.10}$$

olarak ifade edebiliriz. Burada,

$$R_{lr}(s) = \begin{cases} e^{-sT_r} & eger\ R_{lr} = 1 \\ 0 & diger\ durumlarda \end{cases}$$

Yönlendirme matrisi *R(0)=R* denge değerleri arasındaki statik ilişkiyi belirler, örneğin;

$$y^* = R(0)x^*, \quad q^* = R^T(0)p^* \tag{6.11}$$

Herhangi bir sonlu a değeri veri iken $\theta(a)$ a'nın (artan) bir fonksiyonu olarak

$$\theta \tan\theta = a \tag{6.12}$$

'nın (*0, π/2*)'sindeki benzersiz çözüm olsun.
VEGAS'ın kararlılığını ilk olarak maksimum pencere boyutu ile ve daha sonra da minimum kuyruk gecikmesi terimleriyle karakterize edeceğiz. Aşağıdaki sonuçlar eğer denge pencere boyutu yeterince küçük ise teorem 1, ya da aynı anlama gelmek üzere, denge kuyruk gecikmesi yeterince büyükse önerme 2 VEGAS'ın doğrusal kararlılığa sahip olduğunu söylemektedir.
Teorem 1: Tüm r değerleri için ve bazı k_0 değerleri için $k_0 T_r \geq \max x_r T_r$ olduğunu var sayalım. M, herhangi bir kaynak yolundaki hat sayısının üst sınırı olsun, $M \geq \max \sum_l R_{lr}$ eşitlik (6.3) ve eşitlik (6.5) ile tanımlanan VEGAS modeli, eğer $\hat{\eta} := 2\eta/\pi$ ve $\sin c\theta = \sin\theta/\theta$ iken

$$\max_r x_r T_r \sin c\theta\left(\frac{\eta}{x_r T_r}\right) > \frac{\alpha}{Mk_0^2}$$

ise, $(x_r^*, y_l^*, p_l^*, q_r^*)$ denge noktası çevresinde yerel olarak asimptotik kararlılığa sahiptir. Bölüm 6.2.2 de kanıtlanmıştır. $\theta(\cdot)$ sert bir şekilde arttığı ve sinc(-) sert bir şekilde azaldığı için Teorem 1'deki kararlılık koşulunun sol tarafının $x_r^* T_r$ pencere boyutunda hızlı bir şekilde arttığına dikkat edin. Bu sebeple kararlılık koşulu maksimum pencere boyutunda bir limite sahiptir. $x_r^* q_r^* = \alpha$ olduğunda bu koşul doğrudan kuyruk gecikmesi üzerinde bir limite dönüşür. Aşağıdaki önermenin sol tarafı $q_r^* T_r$ durumunda kuyruk gecikmesinde daha düşük bir sınır ima ederek hızlı bir artış göstermektedir.

Önerme 2: Bazı k_0 değerleri için tüm $k_0 T_r \geq \max x_r T_r$ olduğunu var sayalım. M, herhangi bir kaynağın yolu üzerindeki hat sayısının üst sınırını temsil etsin, $M \geq \max \sum_l R_{lr}$ $\hat{\eta} := 2\eta / \pi$ ve $\sin c\theta = \sin\theta / \theta$ iken, eğer

$$\min_r \frac{q_r^* / T_r}{\sin c\theta\left(\frac{\hat{\eta}}{\alpha} . \frac{q_r^*}{T_r}\right)} > M k_0^2$$

ise, eşitlik (6.3) ve (6.5)'te tanımlanan VEGAS modeli $(x_r^*, y_l^*, p_l^*, q_r^*)$ denge noktası etrafında yerel olarak asimtotik kararlılığa sahiptir.

Sonraki sonuç kaynak yolunda birden fazla hat olduğunda kararlılık koşulunun sağlanamadığını göstermektedir.

Önerme 3: Eğer bir kaynak birden fazla hatt sahipse, yani birden fazla l için $R_{lr} = 1$'i karşılayan bir r değeri var ise, kararlılık koşulu sağlanamaz. Kanıt: Teorem 1 ve Önerme 2'deki koşullar aynıdır, bu nedenle Önerme 2 ile çalışacağız. $\theta(\cdot) < \pi / 2$ olduğu için $k_0 \geq 1$ tanımı gereğince $\sin c\theta(.) > 2/\pi$. Bu nedenle, Önerme 2'deki kararlılık koşulu $\min_r \frac{q_r^*}{T_r} > \frac{2M}{\pi}$ sonucunu ima eder. Eğer $M \geq 2$ ise eşitliğin sağ tarafı 1'den büyüktür. Ama $T_r = d_r + q_r^*$ olduğundan eşitliğin sol tarafı 1'I geçemez.

Kararlılık koşulunun yalnızca çoklu hat durumunda yeterli olduğu üzerine odaklanıyoruz. Bununla birlikte, bu tek-hat tutarlı-kaynak durumunda hem zorunlu ve hem de yeterlidir. Şimdi, bu durum için kararlılık bölgesini ve protokol parametresi $\alpha = \alpha_r d_r$ 'nin etkisini açıklayacağız. Örnek 1: Tutarlı kaynaklı tek hat (c,d, N)N sayıda tutarlı kaynak tarafından paylaşılan c kapasiteye sahip ve gidiş-dönüş yayılım gecikmesi d olan bir tek hat düşünelim. Bu olay için, Teorem 1'in kanıtındaki tüm r kaynakları için $\alpha_r = \alpha_0, T_r = T_0,$ ve $w_r = w_0$ olacaktır. Bu bize kararlılık koşulunun (M=1 ve k0=1)

$$\frac{q_r^* / T_r}{\sin c\left(\frac{\hat{\eta}}{\alpha} \cdot \frac{q_r^*}{T_r}\right)} > 1 \qquad butun\ r\ ler\ icin \qquad (6.13)$$

hem gerekli ve hem de yeterli olduğunu ima eder. Denge miktarları q_r^* ve T_r'nin hedef kuyruk uzunluğu α'ya bağlı olduğuna dikkat edin. Protokol parametresi α'nın kararlılık üzerindeki etkisine dair bir bakış edinebilmek için daha basit bir koşula bakıyoruz.

Yukarıda da dikkat çekildiği gibi, $\theta(\cdot) < \frac{\pi}{2},$ olduğundan gerekli koşul

$$\frac{q_r^*}{T_r} > \frac{2}{\pi} \quad butun\ \ r\ ler\ icin$$

olacaktır. Simetri gereğince $T_r = d + q_r^*$ ve $q_r^* = \alpha / x_r^* = \alpha N / c$ olduğundan bu koşul

$$cd < \left(\frac{\pi}{2} - 1 \right) \alpha N \tag{6.14}$$

'a eşdeğerdir. Bu nedenle, VEGAS kararlılığı için gerekli bir koşul bant-genişliği gecikmesi sonucunun küçük olmasıdır. Buna ilaveten, daha büyük hedef kuyruk uzunluğu α ya da daha fazla N kaynak sayısı ile kararlılık bölgesinin daha geniş olmasıdır.

6.2.2. Teorem 1'in Kanıtı

Kanıt üç adımda gerçekleştirilir. İlk olarak, döngü kazanç matrisinin Nyquist yörüngelerinin N sayıda karmaşık *jω* fonksiyonunun konveks gövdesinde dâhil olduğunu göstermek için kaynak [15] ve [17] argümanını takip ediyoruz. İkinci adımda yeteri kadar büyük bir ω düzeyinde, bu fonksiyonlardan en azından bir tanesi -7T faz gecikmesine sahip olduğunda uygun koşullar altında bu fonksiyonların tümünün birim döngüye dâhil olduklarını ve bu nedenle karmaşık düzlemde -1'i kuşatamayacaklarını göstereceğiz. Üçüncü adımda ise bu koşulun teoremde yer alan koşul olduğunu göstereceğiz.

Adım 1: Kaynakta görülen dönüş oranı doğrusallaştırılmış eşitlik (6.6), (6.8), (6.9) ve (6.10) eşitlikleri kullanılarak

$$diag\left(a_r \frac{x_r^*}{q_r^*} \right) diag\left(\frac{e^{-sT_r}}{sT_r + a_r} \right) R^T(-s)\, diag\left(\frac{1}{c_l s} \right) R(s)$$

olarak tanımlanır. Kararlılık için, bu fonksiyonun özgün değerlerinin karmaşık düzlemde $s = j\omega$, ve $w \geq 0$ için -1 değerini çevrelemeyeceğini göstermek yeterlidir.

$$k = \frac{M a_r T_r}{q_r^*} = \frac{2\eta M}{\pi \alpha} \tag{6.16}$$

ve

$$\hat{R}(jw) = diag\left(\sqrt{\frac{1}{c_l}} \right) R(jw)\, diag\left(\sqrt{\frac{x_r^*}{M}} \right)$$

iken kendine özgü değerler seti

$$L(jw) = diag\left(\frac{k e^{-jwT_r}}{jwT_r(jwT_r + a_r)} \right) \hat{R}^T(-jw)\, \hat{R}(jw) \tag{6.15}$$

eşitliğinin değerlerindekiyle aynı biçimdedir. Neigen yörüngesini ve başlangıç noktasının konveks gövdesini temsil ederken kaynak [15]'ten, L(jω)'nin dalga bandı

$$\sigma(L(jw))$$
$$= \sigma\left(diag\left(\frac{ke^{-jwT_r}}{\Re w_r(\Re w_r + a_r)} \right) \hat{R}^T(-jw)\hat{R}(jw) \right)$$

$$\subseteq \rho\left(\hat{R}^T(-jw)\hat{R}(jw)\right) \cdot co\left(0 \bigcup \left\{ \frac{ke^{-jwT_r}}{\Re w_r(\Re w_r + a_r)}, r = 1,..., N \right\} \right)$$

koşulunu sağlar. Eşitlik (6.11) gereğince tüm mutlak satır toplamları 1'e eşit olduğundan $\hat{R}(jw)$ 'nun dalga yayılımı

$$\rho\left(\hat{R}^T(-jw)\hat{R}(jw)\right)$$
$$\leq \left\| diag\left(\frac{1}{M}\right)\hat{R}(-jw)diag\left(\frac{1}{c_l}\right)(jw)diag\left(x_r^*\right) \right\|_\infty$$
$$\leq \left\| diag\left(\frac{1}{M}\right)\hat{R}(-jw) \right\|_\infty \left\| diag\left(\frac{1}{y_l^*}\right)\hat{R}(-jw)diag\left(x_r^*\right) \right\|_\infty$$
$$= 1$$

koşulunu sağlar. Bu nedenle,

$$\sigma(L(jw)) \subseteq co\left(0 \bigcup \left\{ \frac{ke^{-jwT_r}}{\Re w_r(\Re w_r + a_r)}, r = 1,..., N \right\} \right)$$

Şimdi

$$\Lambda_r(jw) := \frac{ke^{-jwT_r}}{\Re w_r(\Re w_r + a_r)}$$

olsun. Şimdi, teorem koşulu altında hiçbir $\Lambda_r(jw)$düzeyinde konveks kombinasyonunun kritik -1 noktasını çevrelemeyeceğini gösteriyoruz.

Adım 2:

$a_0 = \min_r a_r$ ve $T_0 = \max_r T_r$ olsun; w_r $r = 0,1,..., N$, (*0, π/2*) aralığında

$$w_r T_r \tan a w_r T_r = a_r \quad r \geq 0 \tag{6.17}$$

koşulunu sağlayan değer olsun. Tüm r değerleri için açıkça $w_0 \leq w_r$ olacaktır. Burada $\Lambda_r(jw)$ özgün değer —tv faz gecikmesine sahipken w_r $r \geq 1$, kritik frekanstır. Bu nedenle, $w < w_0 \leq w_r$, için $\Lambda_r(jw)$ 'nun konveks kombinasyonu -1'I çevreleyemez; çünkü tüm r değerleri için faz$(\Lambda_r(jw) > -\pi)$. Şimdi $w \geq w_0$, için tüm $\Lambda_r(jw)$i değerlerinin birim döngü içinde yer aldığını ve bu nedenle bunların konveks kombinasyonunun -1 değerini çevreleyemeyeceğini göreceğiz. $k_0 T_r \geq T_o$, olduğundan, $w \geq w_0$ değeri için

$$
\begin{aligned}
\left|\Lambda_r(jw)\right| &= \frac{k}{wT_r\sqrt{w^2T_r^2+a_r^2}} \\
&\le \frac{kk_0^2}{w_0T_0\sqrt{w_0^2T_0^2+a_0^2}} \\
&\le \frac{kk_0^2}{a_0}\cdot\frac{1}{w_0T_0}\cdot\frac{\alpha_0}{\sqrt{w_0^2T_0^2+a_0^2}} \\
&= \frac{kk_0^2}{a_0}\cdot \sin c\theta(a_0)
\end{aligned}
$$

Burada son eşitlik (6.17)'yi ve (6.12)'deki *θ(.)* tanımını takip etmektedir.

$$
\frac{\sin c\theta(a_0)}{a_0} < \frac{1}{kk_0^2}
$$

$$
k = \frac{\hat{\eta}M}{\alpha} \quad ve \quad a_0 = \min_r a_r = \frac{\hat{\eta}}{\max_r x_r^*T_r}
$$

$$
\max_r x_r^*T_r \sin c\theta\left(\frac{\hat{\eta}}{\max_r x_r^*T_r}\right) < \frac{\alpha}{Mk_0^2}
$$

$\sin c\theta\left(\hat{\eta}(x_r^*T_r)^{-1}\right)$ $x_r^*T_r$ hızlı bir artış gösterdiğinden

$$
\begin{aligned}
&\left(\max_r x_r^*T_r\right)\sin c\theta\left(\frac{\hat{\eta}}{\max_r x_r^*T_r}\right) \\
&= \max_r\left\{x_r^*T_r \sin c\theta\left(\frac{\hat{\eta}}{x_r^*T_r}\right)\right\}
\end{aligned}
$$

böylece teoremdeki kararlılık koşulunu sağlamış oluruz.

6.3. Kararlı VEGAS

Bu kısımda, bir VEGAS kaynaklar ağını kararlı hale getirmek için her bir kaynak için bir PD (proportianal differential- oranlı diferansiyal) kontrolcüsü öneriyoruz. VEGAS algoritmasını eşitlik (6.5)

$$\dot{x}_r = \frac{w}{T_r^2(t)} \cdot \tan^{-1}(n_T(t)\Delta_r(t)) \tag{6.18}$$

ya da

$$\dot{x}_r = \begin{cases} \min\left[\frac{w}{T_r^2(t)}\left(e^{\eta_r(t)\Delta_r(t)}-1\right), \frac{\ln 2}{T_r(t)}x(t)\right], & eger\ \Delta_r(t) > 0 \\ \max\left[\frac{w}{T_r^2(t)}\left(1-e^{-\eta_r(t)\Delta_r(t)}\right), -\frac{\ln 2}{T_r(t)}x(t)\right], & diger\ durumlarda \end{cases} \tag{6.19}$$

şeklinde geliştiriyoruz.

Burada $T_r(t) = d_r + q_r(t), \Delta_r(t))1 - \frac{x_r(t)q_r(t)}{\alpha_r d_r} - \kappa_r(t)\dot{q}_r(t)$ ve

$$\kappa_r(t) = \frac{1}{a}.\frac{T_r(t)}{q_r(t)} \tag{6.20}$$

$$\eta_r(t) = \frac{\mu a}{w}.x_r(t)T_r(t) \tag{6.21}$$

Burada w parametresi her gidiş-dönüş süresi için pencere boyutundaki maksimum değişimi belirler. İlk VEGAS için, maksimum değişim her döngü dolaşımı için 1 pakettir. a>0 ve $\mu \in (0,1)$ parametreleri kararlılığı garanti altına alacak şekilde seçilecektir (aşağıya bakınız). Genel kazanç parametresi $\eta_r(t)$ mevcut pencere boyutu ile oranlıdır: pencere boyutu arttıkça yanıt daha hızlı olacaktır. Diferansiyel terimindeki $\kappa_r(t)$kazancı döngü zamanının r kaynağının uçtan-uca kuyruk gecikmesine bölümü ile orantılıdır ve $\dot{q}_r(t)$ 'nin bir normalleştirmesi olarak hizmet eder. Ek diferansiyel terimi $\kappa_r(t)\dot{q}_r(t)$ ise $q_r(t)$ 'nin gelecek değerini tahmin eder. Bu terim olmadan, eğer hatlarda arabelleğe alınan paketlerin sayısı $x_r(t)q_r(t))\alpha_r d_r$ ile karşılaştırmalı olarak küçük ise kaynak oranı $\kappa_r(t)$artacaktır. Bu terim olduğunda, $x_r(t)q_r(t)$ küçük bile olsa fiyatlar hızla büyüyor olsa bile, yani $\dot{q}_r(t)$ büyük olsa bile kaynak, oranını düşürebilecektir. Kaynak [22]'deki bağlantı algoritmasında aynı zamanda, AQM tasarımının uygun değer kontrol formülasyonundan destek alan bir diferansiyel teriminin de kullanıldığına dikkat ediyoruz.

Hem eşitlik (6.18) ve hem de eşitlik (6.19)'un orijinal VEGAS'la aynı denge noktasına sahip olduklarını ve her ikisinin de aynı birinci-derece eşitlikleri doğru sallaştırdığını görüyoruz:

$$\delta x_r = \left.\frac{\partial f_r}{\partial x_r}\right|_* \delta x_r(t) + \left.\frac{\partial f_r}{\partial q_r}\right|_* \delta q_r(t) + \left.\frac{\partial f_r}{\partial \dot{q}_r}\right|_* \delta \dot{q}_r(t)$$

burada

$$\left.\frac{\partial f_r}{\partial x_r}\right|_* = -\frac{\mu a}{T_r}$$

$$\left.\frac{\partial f_r}{\partial q_r}\right|_* = -\frac{\mu a x_r^*}{T_r q_r^*}$$

$$\left.\frac{\partial f_r}{\partial \dot{q}_r}\right|_* = -\frac{\mu x_r^*}{q_r^*}$$

ve Laplace dönüşümü ise

$$\delta x_r(s) = -\frac{\mu x_r^*}{q_r^*}\left(\frac{sT_r + a}{sT_r + \mu a}\right)\delta q_r(s) \tag{6.22}$$

şeklindedir. η_r ve κ_r'ı öyle seçmiştik ki, eşitlik (6.22)'deki öncü-gecikme düzenleyicisi tüm kaynaklar için genel bir sıfır a ve μa kutbuna sahiptir. Tersine, kaynak [17]'deki algoritma fayda fonksiyonlarının kısıtsız seçimine bağlı olarak, μ_r 'nin r'ye bağlı olmasına izin vermektedir. Bu nedenle kaynak [17]'den biraz farklı bir kararlılık kanıtına ihtiyacımız var.

Teorem 4: Tüm r değerleri için ve bazı k0. değerleri için $k_0 T_r \geq \max_r T_r$ olduğunu var sayalım. M, her bir kaynak yolu üzerindeki hat sayısının üst sınırı olsun, $M \geq \max_r \sum_l R_{lr}$. Verili her bir a >0 ve $\mu \in (0,1)$ değeri için eğer

$$\max_r x_r T_r < \frac{\alpha\phi}{\mu k_0 M}\sqrt{\frac{\phi^2 + \mu^2 (k_0 a)^2}{\phi^2 + (k_0 a)^2}} \tag{6.23}$$

ise ya da aynı anlama gelmek üzere

$$\phi = \tan^{-1}\frac{2\sqrt{\mu}}{1-\mu} \text{ve}$$

$\alpha = \alpha_r d_r$ ortak hedef kuyruk uzunluğuna eşit iken eğer

$$\min_r \frac{q_r^*}{T_r} > \frac{\mu k_0 M}{\phi}\sqrt{\frac{\phi^2 + \mu^2 (k_0 a)^2}{\phi^2 + (k_0 a)^2}} \tag{6.24}$$

ise eşitlik (6.3)'te ve eşitlik (6.18)—(6.21) arasında tanımlanan geliştirilmiş VEGAS modeli (x^*, y^*, p^*, q^*) denge noktası etrafında yerel olarak kararlılığa sahiptir.

Kanıt: Kanıt iki adımdan oluşmaktadır. İlk olarak, döngü kazanç matrisinin Nyquist yörüngelerinin N sayıda karmaşık jω fonksiyonunun konveks gövdesinde dâhil olduğunu göstermek için kaynak [15] ve [17] kanıtlarını takip ediyoruz. İkinci adımda yeteri kadar büyük bir ω düzeyinde, bu fonksiyonlardan en azından bir tanesi -7T faz gecikmesine sahip olduğunda uygun koşullar altında bu fonksiyonların tümünün birim döngüye dâhil olduklarını ve bu nedenle karmaşık düzlemde -1'i kuşatamayacaklarını göstereceğiz.

Adım 1: Doğrusallaştırılmış eşitlik (6.22), (6.8), (6.9) ve (6.10) kullanılarak kaynaklarda görülen dönüş oranı

$$diag\left(\frac{\mu x_r^*}{q_r^*}\right)diag\left(\frac{sT_r+a}{sT_r+\mu a}e^{-sT_r}\right)R^T(-s)diag\left(\frac{1}{c_l s}\right)R(s)$$

$$L(jw)=diag\left(\frac{\mu MT_r}{q_r^*}\right)\cdot$$

$$diag\left(\frac{e^{-jwT_r}}{jwT_r}\cdot\frac{jwT_r+a}{jwT_r+\mu a}\right)\hat{R}^T(-jw)\hat{R}(jw)$$

şeklinde yazılabilir. Burada

$$\hat{R}(jw)=diag\left(\sqrt{\frac{1}{c_l}}\right)R(jw)diag\left(\sqrt{\frac{x_r^*}{M}}\right) \tag{6.25}$$

eşitlik (6.25) ve (6.21)'i kullanan kanıt

$$\rho\left(\dot{R}^T(-jw)\dot{R}^T(jw)\right)\le 1$$

sonucunu verir. Kaynak [15]'teki durum o halde $L(jw)$ 'nın tüm kendine özgü değerlerinin konveks gövdeye sahip olacağını ima eder.

$$co\left\{0\bigcup\frac{MT_r}{q_r^*}\cdot\Lambda(jwT_r,\quad r=1,...,N,\right\} \tag{6.26}$$

burada

$$\Lambda(jwT_r):=\mu\cdot\frac{e^{jwT_r}}{jwT_r}\cdot\frac{e^{jwT_r}+a}{e^{jwT_r}+\mu a}$$

A(-) 'nin r'den bağımsız olduğuna dikkat edin. Genelleştirilmiş Nyquist kararlılık ölçütü nedeniyle kaynak [23] eğer eşitlik (6.26)'daki set -1 değerini çevrelemiyorsa sistem kararlıdır.

Adım 2: w_r, r kaynakları için faz $\angle\Lambda(jw_rT_r)$ 'nin —π'ye eşit olduğu kritik frekans olsun:

$$w_rT_r-\angle\frac{jwT_r+a}{jwT_r+\mu a}=\frac{\pi}{2}$$

Genelleşmede kayıp olmadan, tüm r değerleri için $T_1\ge T_r$ olduğunu var sayabiliriz. O halde tüm r değerleri için $w_rT_1=w_1T_r$ olduğundan yine tüm r'ler için $w_1\le w_r$ 'dir. Böylece, $w\le w_1$ iken eşitlik (6.26)'nın konveks gövdesi -1'I çevreleyemez. $w\ge w_1$ iken eşitlik (6.26)'daki set -1 değerini çevrelemez.. Şimdi bunun teoremdeki koşullarla ima edildiğini göstereceğiz. $w\ge w_1$, için $wT_r\ge w_1T_r\ge w_1T_r/k_0$ Büyüklüğün

$$\frac{mT_r}{q_r^*}\cdot\left|\Lambda(jwT_r)\right|<1 \tag{6.27}$$

ωT 'nin hızlı bir şekilde azalan bir fonksiyonu olduğuna dikkat edin.

$$\left|\Lambda\left(jwT_r\right)\right|=\frac{\mu}{wT_r}\sqrt{\frac{(wT_r)^2+a^2}{(wT_r)^2+\mu^2a^2}}$$

Bu sebeple, tüm r değerleri için

$$|\Lambda(jwT_r)| \le \left|\Lambda\left(jw_1\frac{T_1}{k_0}\right)\right|$$
$$= \frac{\mu k_0}{w_1 T_r}\sqrt{\frac{(w_1T_1)^2+(ak_0)^2}{(w_1T_1)^2+\mu^2(ak_0)^2}}$$
$$= \frac{\mu k_0}{\phi}\sqrt{\frac{\phi^2+(ak_0)^2}{\phi^2+\mu^2(ak_0)^2}}$$

φ teoremde tanımlı şeklinde iken son eşitsizlik tüm r değerleri için

$$w_r T_r \ge \frac{\pi}{2} - \tan^{-1}\frac{1-\mu}{2\sqrt{\mu}} = \phi$$

sonucunu ima eder.

Bu durumda teoremdeki eşitlik (6.24) koşul eşitlik (6.27)'nin sağlanmasını garanti eder. $x_r^* q_r^* = \alpha$ olduğundan eşitlik (6.23) ve (6.24) koşulları eşdeğerdedir. Bu nedenle ispat aşağıdaki denklemle tamamlanmaktadır.

$$\Lambda(jwT_r) = \mu\frac{e^{jwT_r}}{jwT_r}\cdot\frac{jwT_r - a}{jwT_r + \mu a}$$

burada $0 < \mu < 1$ ve $a > 0$. O halde, tüm r değerleri için $\omega > 0$,

$$\angle\Lambda(jwT_r) \ge -wT_r - \frac{\pi}{2} - \tan^{-1}\frac{1-\mu}{2\sqrt{\mu}}$$

O halde

$$h_r'(w) = \frac{(1-\mu)((wT_r)^2 - a\ \mu))a}{((wT_r)^2 - a^2)((wT_r)^2\mu^2a^2}$$

den

$$h_r(w) := \angle\frac{jwT_r - a}{jwT_r + \mu a}$$
$$= \tan^{-1}\left(\frac{wT_r}{a}\right) - \tan^{-1}\left(\frac{wT_r}{\mu a}\right)$$

9

$\mu \in (0,1)$ olduğundan, $h_r'(w) = 0$'ın çözümü $w_r^* = \frac{a\sqrt{u}}{T_r}$,'nın faz $h_r(w)$'ı minimize eden çözümü kontrol edilebilir. Bu nedenlerden bağımsız olarak,

$$\angle h_r(w) \ge \tan^{-1}\sqrt{\mu} - \tan^{-1}\frac{1}{\sqrt{\mu}} := \psi$$

Buna ek olarak,

$$\tan\psi = \frac{\sqrt{u} - \frac{1}{\sqrt{u}}}{2} = -\frac{1-\mu}{2\sqrt{\mu}}$$

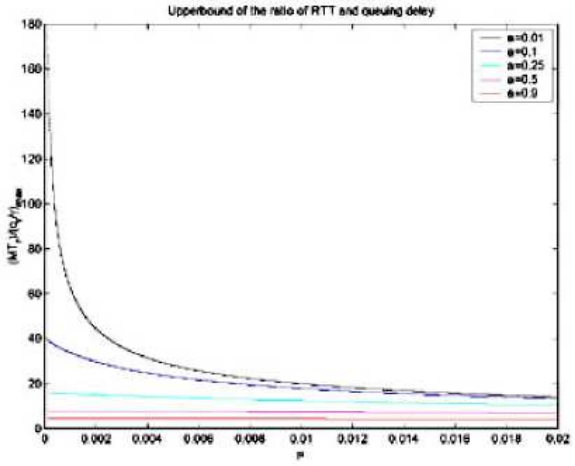
$$\angle\Lambda(jwT_r) = wT_r - \frac{\pi}{2} + \angle h_r(w)$$
$$\geq -w_r T_r - \frac{\pi}{2} - \tan^{-1}\frac{1-\mu}{2\sqrt{\mu}}$$

ve böylece denklem takip eder.

Teorem 4'ün çıkarımları üzerinde duralım. Tutarlı döngü tamamlanma süresinde, yani k = 1 iken eşitlik (6.24) kararlılığı

$$M \max_r \frac{T_r}{q_r^*} < \frac{\theta}{\mu}\sqrt{\frac{\phi^2 + \mu^2 a^2}{\phi^2 + a^2}} \tag{6.28}$$

haline gelir. M bir kaynak yolundaki boğum hatlarının sayısının bir sınırını gösterir. Tipik olarak 10'dan az olmaktadır. |f- döngü gidiş-dönüşündeki kuyruk gecikmesinin toplam döngü süresine oranıdır; her iki nicelik bir VEGAS kaynağında kullanılabilir durumdadır. Mevcut ağ için, bu oran uzun gecikme rotaları için 5'ten az küçük olarak görünmektedir. Bu yüzden, tasarım parametreleri a ve μ 'nın kararlılık koşulunun sağ tarafının 100'den fazla olmasını garanti edecek şekilde seçimi güvenli görünmektedir. Şekil 6.1'e göre, bu durum küçük a ve μ değerleri seçilmesini gerektirmektedir (söz gelimi, a = 0.01 ve μ = 0.001).



Şekil 6.1. $\frac{MT_r}{q_r^*}$ 'nin üst limiti

Eşitlik (6.20) ve eşitlik (6.21)'deki tanımı hatırlarsak; $kappa_r(t) = (T_r(t)/q_r(t))/a$ de $\eta_r(t) = \mu a(x_r(t)T_r(t)/w)$ Küçük a büyük bir *kappar(t),* değerini ima etmektedir. Bunun anlamı kararlı VEGAS'ın fiyat değişimine $\dot{q}_r(t)$ daha hızlı tepki gösterdiğidir. Küçük μa ise küçük bir η, değerini ima etmekte, bu da denge çevresindeki eşitlik (6.18)'in eğiminin küçük olduğunu göstermekte ve kazanç genelinde daha yumuşak bir eğim olduğunu ortaya koymaktadır. Tutarsız döngü-zamanı olayı için, yani $k_0 > 1$durumu için, kararlılığı garanti altına almak için tutarlılık durumundan daha küçük bir a değeri gerekmektedir.

Örnek 2: Tutarlı kaynaklı tek hat *(c, d, N)*

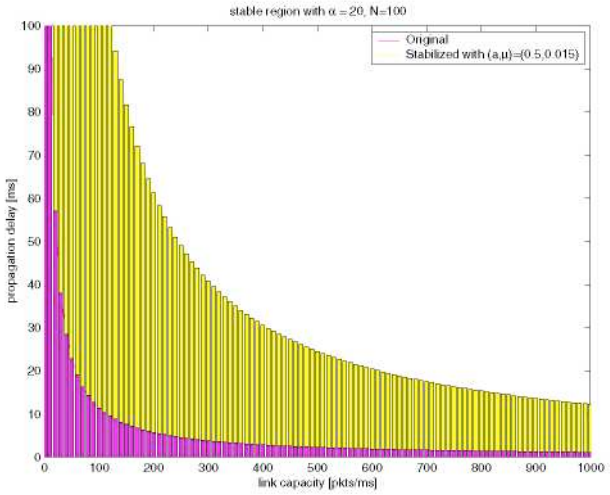
Orijinal VEGAS kararlılığı ile doğrudan bir karşılaştırma yapabilmek için Örnek 1'deki kurgunun aynısını düşünüyoruz: N sayıda tutarlı kaynak tarafından paylaşılan c kapasiteye sahip ve gidiş-dönüş yayılım gecikmesi d olan bir tek hat. Bu olay için yeterli koşul,

$$\dot{q}_r / T_r > \frac{\mu}{\phi}\sqrt{\frac{\phi^2 + a^2}{\phi^2 + \mu^2 a^2}} \quad butun\ r\ ler\ icin$$

$T_r = d + \dot{q}_r$ ve $\dot{q}_r = \alpha / x_r^* = \alpha N / c$ simetrisi gereğince bu koşul (M = 1 ve k0 = 1) ile Teorem 4'ün

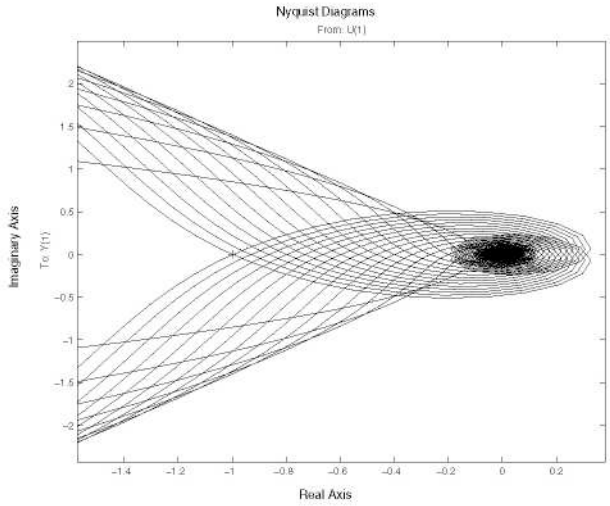
$$cd < \left(\frac{\phi}{\mu}\sqrt{\frac{\phi^2 + a^2}{\phi^2 + \mu^2 a^2}} - 1 \right) \alpha N \tag{6.29}$$

şeklinde sadeleştirilmesiyle eş değerdir. Bu nedenle, orijinal VEGAS' ta olduğu gibi, bu aynı zamanda daha büyük kuyruk uzunluğu α ya da daha fazla N kaynak sayısı ile daha büyük bir kararlılık bölgesine de sahiptir. Buna ek olarak, orijinal VEGAS' ın kararlılığı için (29)'un sağ tarafının (14)'ün sağ tarafından büyük olabilmesi gibi α ve N veri iken kararlı VEGAS küçük bir (a > 0, μ € (0,1)) değeri seçebilir. Bu durum Şekil 6.2' de gösterilmektedir. Şekilde, eşitlik (6.14) ve (6.29)'daki kararlılık bölgeleri VEGAS ve kararlı VEGAS için sırasıyla (a,μ) = (0.5,0.015)ve α = 20 paket, N = 100 kaynak için diyagrama dökülmüştür.



Şekil 6.2. Örnek 1 ve 2'nin kararlılık bölgeleri: tutarlı kaynaklar tarafından paylaşılan tek hat.

Kararlılık koşulu yalnız başına yeterlidir. Gerçekten, a ve μ için daha az tutucu değerler kullanılabilir. Örneğin, $MT_r q_r^* = 100$ için $f(MT_r q_r^*) \cdot \Lambda(jwT_r)$ 'nin Nyuquist dağılımı Şekil 6.3'te, Örnek 2'deki a = 0.1 ve μ G [0.001,0.015] senaryosu için diyagrama dökülmüştür. Bu a ve μ değerleri Teorem 4'teki kararlılık koşulunu sağlamasalar bile, Nyquist dağılımda da görüldüğü gibi, ağ gerçekten de kararlıdır.



Şekil 6.3. Nyquist Kararlılık a = 0.1, μ = [0.001 : 0.015]

$$\Lambda(jwT_r) \quad \frac{MT_r}{q_r^*} = 100 \;\; icin.$$

6.4. Uygulama ve Derleme

TCP VEGAS'ın en çekici özelliği yüksek hızlı büyük gecikmeli ağlar için uygunluğudur. Bu rejimde pencere boyutu büyüktür ve TCP RENO ya da varyantları denge düzeyinde aşırı derecede küçük bir kayıp olasılığı oluşturmak zorundadırlar. Bu kadar küçük bir olasılığa güvenmek büyük sorunları da beraberinde getirmektedir.

Diğer taraftan, VEGAS'ın her ikisi de gecikmenin bir yoğunluk ölçüsü olarak kullanımından ileri gelen iki avantajı daha vardır. İlk olarak, VEGAS'ın kesinlikli hat algoritması ağ kapasitesiyle ilgili yerleşik bir ölçeklemeye sahiptir. Bu kararlı kaynak algoritmasıyla bir aradadır ve böylece potansiyel olarak daha büyük bant genişliği gecikme sonucunu ölçekleyebilir. İkincisi, bir kaynağın her gecikme ölçüsü ikili-değerli kayıp ya da işaretlemenin ortaya koyduğundan daha ince bir yoğunluk tahmini ortaya koyar. Kapasite büyüdükçe, α parametresini ölçekleyerek yoğunluk işaretinin (gecikmenin) gücünü sağlamak için kaynakta ölçekleme yapmak daha kolaylaşmaktadır.

Gecikmenin düşük bant genişliğinde dengeye ulaşmak için VEGAS'ta aşırı ölçüde olabilmesi sorunu geniş bant rejiminde çok daha az şiddetlidir. Buna ek olarak, yayılım gecikmesi tahminindeki hatayla ve süreğen yoğunlukla ilgili sorun kaynak [5], [8] da yüksek kapasiteyle birlikte arabelleklerin daha sık boşaltılmasıyla kolaylaşmaktadır. Gecikmenin yoğunluk kontrolünde kullanımına ilişkin başka konular bulunmakla birlikte, ECN çok geniş kapsamlı derlenmedikçe bu sorunların kontrol için aşırı derece küçük kayıp olasılığının kullanılmasına güvenilmek zorunda kalınması şeklindeki temel güçlükten daha önemli olmadıkları görülmektedir.

Şimdi kararlı VEGAS için yeni AQM ve ECN'nin kademeli derlenmesiyle çalışmak için uygulanabilir ve tutarlı bir strateji tanımlıyoruz. VEGAS daki hat algoritması kuyruk gecikmesini şu şekilde hesaplamaktadır:

$$\dot{p}_l(t) = \frac{1}{c_l}(y_l(t) - c_l) \tag{6.30}$$

VEGAS'ın ağ kapasitesiyle yerleşik ölçeklenebilirliğini veren, q'ya bölünmedir. Kaynak [8]'de de incelendiği gibi, VEGAS yoğunluk fiyatlarını otomatik olarak hesaplamak için

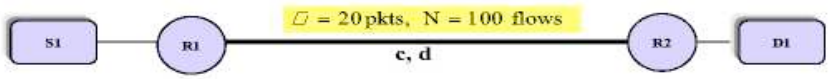
sıfırdan-faklı bir kuyruk gecikmesi oluşturmak pahasına, arabellek sürecini kullanır. Bunlar VEGAS'ın kesin bir şekil 6.3 de çözdüğü fayda en yüsek düzeye çıkarma problemi için kullanılan Lagrange çarpanlarıdır. Kaynak [14] ve [17]'nin ölçeklenebilir şemadaki hat algoritması, $\dot{p}_l(t)$ fiyatını hesaplamak için gerçek hat kapasitesi q yerine biraz daha küçük olan (mesela, q'nun %95'i) sanal bir hat kapasitesini kullanıyor olması haricinde eşitlik (6.30)'daki ile aynı ifadeyi kullanır. Sanal bir kuyruk kullanmanın avantajı fiyatlar sıfırdan-farklı değerlere yakınlaşırken gerçek kuyruğun dengede temizlenecek olmasıdır. Kuyruklar şimdi boş olduklarından kuyruk gecikmesi artık yoğunluk işaretçisi olarak hizmet etmeyecektir. ECN işaretlemesi fiyatları açıkça geri beslemek için kullanılmalıdır.

Şimdi her iki tipten hatta sahip olan bir ağ hayal edelim; bu hatlardan biri kuyrukları temizlemek için ne ECN'yi kullanır ne de AQM'i işletirken diğeri bnu kullanıyor olsun. İlk tip bir kuyruk oluşturmakta ama işaretlememekte, ancak ikincisi kuyruk gecikmesine sahip değilken kaynakları işaretler akımı göndermektedir. Bir kaynak iki tip geri besleme sinyalini gözlemektedir. Birinci tip *l* hatlarından gelen toplam kuyruk gecikmesi ve ikinci tip hatlardan REM tahmini sonrasında gelen toplam fiyatlar. Bu iki işaret yalnızca birbiriyle etkileşmemekte, bunların toplamları kaynak yolundaki toplam fiyatı da açıkça ortaya çıkarmaktadır. Bu nedenle, her iki işareti gözleyerek ve bunları toplayarak kaynak kontrol için gerekli bilgiyi otomatik olarak temin etmekte ve bunu yaparken yol üzerindeki hatların tipi ve sayısı konusunda bir bilgiye sahip olmamaktadır. ECN yoluyla AQM'ye daha ve daha fazla hat dönüştürüldükçe, kaynak algoritmasının yükseltilme ihtiyacı da artmaktadır. Tek etki kuyruk gecikmesinin düzenli olarak azalmasıdır.

6.5. Benzetim Sonuçları

VEGAS ve kararlı VEGAS algoritmalarının farklı sayılarda kaynaklar, tutarlı ve tutarsız kaynaklar, tekli ve çoklu boğum hatları ile ve dinamik senaryolar içindeki davranışlarını incelemek için geniş kapsamlı *ns-2* simülasyonları oluşturduk kaynak. Yer sınırlaması nedeniyle, burada yalnızca tek boğumlu hatla ilgili önceki kesimlerde ele aldığımız teoriyi doğrulayan ve kararlı VEGAS' ın yüksek hızlı büyük gecikmeli pencere boyutunun paket olarak büyük olduğu ağlarda uygunluğunu gösteren en basit sonuçları açıklayacağız. Diğer benzetim sonuçları bu metnin dergi versiyonunda ele alınmaktadır.

Örnek 1 ve 2'deki N sayıda tutarlı kaynak tarafından paylaşılan tek boğum hattı üzerinde duran senaryoları benzettik. Simülasyonlar iki yönlendirici üzerinden alışıldık N sayıda kaynağın N sayıda hedefe bağlandığı uçbirim kullanmaktadır. Kaynaklar ve bunların yönlendiricisi arasındaki erişim hatları sıfır gecikmeli boğumsuz ve iki yönlendirici arasındaki hat ise yalnızca tek boğumlu ve 1 KB sabit paket boyutlu c Gbps kapasiteye sahiptir. Yönlendiriciler arası gecikme d/2 ms.'dir. Droptail ve kuyruk kapasitesiyle ilgili FIFO prensibine göre (ilk giren-ilk çıkar) çalışan yönlendiriciler 40K pakete ayarlıdır ve paket kaybı olasılığı ihmal edilebilir düzeydedir. Paket gönderimine tüm kaynaklar eş zamanlı olarak başlarlar.α = 20 paket ve N = 100 olarak sabitleyip c ve d'yi farklılaştırıyoruz. Şekil 6.4'te de görüleceği gibi, Şekil 6.2 bölgesinde üç farklı (c, d)'den oluşan üç farklı benzetim seti sunuyoruz.



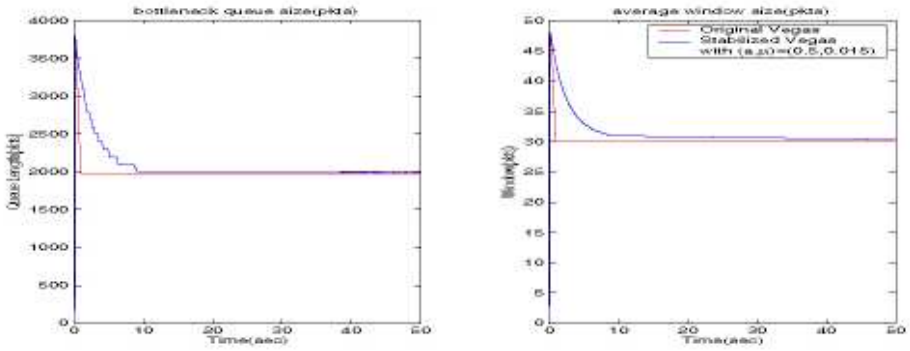
	capacity c [Gbps] (pkts/ms)	propagation delay d [ms]	expected queue length □N [pkts]	expected window size □ □ $\frac{cd}{N}$ [pkts]
(a)	0.8 (100)	10	2000	30
(b)	8.0 (1000)	10	2000	120
(c)	0.8 (100)	100	2000	120

Şekil. 6.4. Tutarlı kaynak ve tek boğumlu ağ topolojisi : (α, N) = (20 pkts, 100 flows). Kararlı VEGAS için, (a, μ) = (0.5, 0.015).

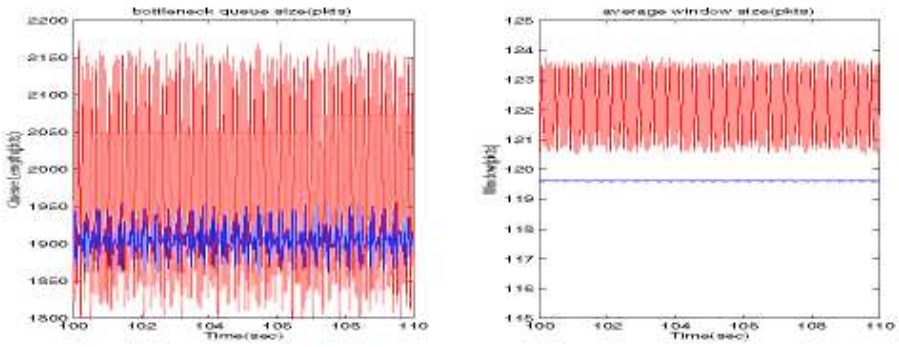
Simülasyon (a) her iki kararlılık bölgesinin kesişimin deki küçük kapasite ve gecikme içindir. Simülasyon (b) kapasiteyi 10 kat artırır, benzetim (c) ise (a)'da kullanılan gecikmeyi 10 kat artırır. Hem (b) ve hem de (c) orijinal VEGAS kararlılık bölgesinin dışındadır, ancak halen daha kararlı VEGAS 'ın kararlılık bölgesi içindedir. Hem orijinal VEGAS ve hem de kararlı VEGAS için her kaynağın hedef kuyruk uzunluğunu α = 20 paket ve N = 100 "flows" olarak oluşturuyoruz.

Şekil 6.4'ün son iki kolonu kaynak [8]'den hesaplanan denge kuyruk uzunluğu ve denge pencere boyutunu göstermektedir.

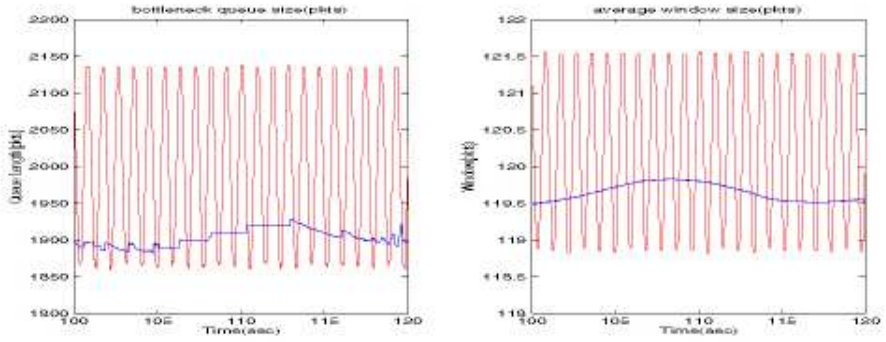
Simülasyon sonuçları Şekil 6.5'te gösterilmektedir. Her olay için ilk dağılım boğum hattında ara belleğe alınan toplam kuyruk uzunluğunu göstermektedir. İkinci dağılımlar N kaynak üzerinden alınan ortalama pencere boyutudur. Beklendiği gibi, orijinal VEGAS (b) ve (c) olaylarında kararsızlık gösterirken kararlı VEGAS kararlı kalmaktadır.



(a) c = 800 Mbps (100 pkts/ms) ve d = 10 ms



(b) c = 8 Gbps (1000 pkts/ms) ve d = 10 ms

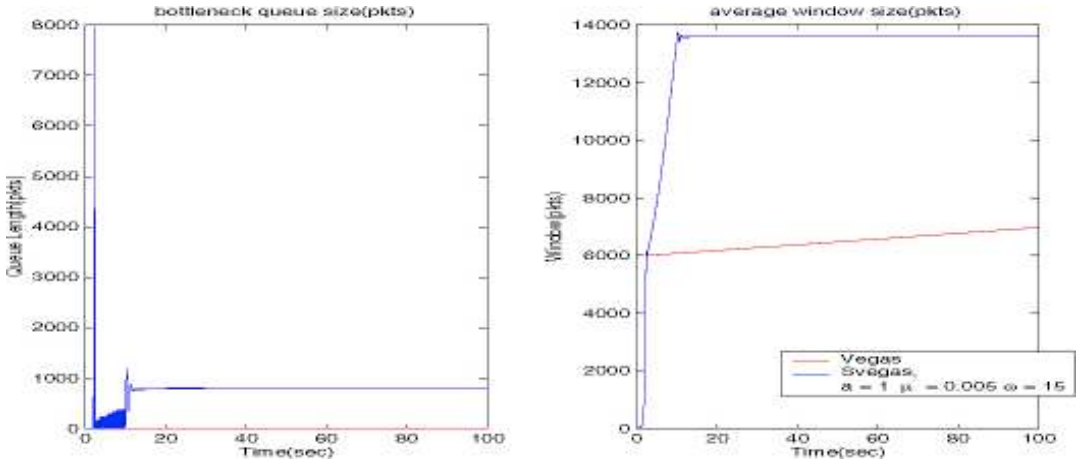


(c) c = 800 Mbps (100 pkts/ms) ve d = 100 ms

Şekil 6.5. Denge yakınında kuyruk uzunluğu ve ortalama pencere boyutu (α = 20 pkts, N = 100).

Kararlı VEGAS'ın pencere boyutunun büyük olabildiği yüksek hızlı büyük gecikmeli ağlardaki performansını görmek için çok küçük akım sayıları kullanıyoruz; N=3, (c,d)=(3.2 Gbps, 100ms.). Aynı zamanda, α = 400 paket olarak ayarlıyoruz ki, böylece yayılım g9ecikmesinin yaklaşık %3'üne kuyruk gecikmesi olarak izin verilmektedir. Burada [8]'den hesaplanan denge pencere boyutu yaklaşık olarak kaynak başına W* = 13,700 pakettir. Şekil 6,6'da görüldüğü gibi, kararlı VEGAS 10 ms'den daha kısa sürede neredeyse denge pencere boyutuna ulaşmaktadır, buna karşın orijinal VEGAS'ın doğrusal artış hızı bunun çok

gerisinde kalmaktadır. Kararlı VEGAS bunun yanında aynı zamanda iyi bir düzenlilik durumuna sahiptir.



Şekil 6.6.yüksek hızlı ağ için hızlı yaklaşma c = 3.2 Gbps, d =100 ms ve (α, N)=(400 pkts, 3).

6.6. Sonuç

Bu bölümde, tutarsız ileri ve geri yönlü gecikmelere sahip genel birçok-hatlı çok-kaynaklı kurgu içinde VEGAS kararlılığının detaylı bir analizini sunduk. VEGAS'ın gecikme varlığında kararsız olabileceğini ortaya koyan bir kararlılık koşulu türettik. VEGAS'ı büyük ağ gecikmeleri halinde kararlı hale getirecek küçük bir modifikasyon önerdik.

VEGAS ağ kapasitesiyle yerleşik ölçeklenebilme özelliği nedeniyle özellikle yüksek hızlı ağlarda caziptir. Yüksek bant genişliği rejiminde VEGAS' ın sürekli yoğunlukla ilişkili potansiyel sorunu azaltılabilmektedir. Bunun yanı sıra, RENO' nun yapmak zorunda olduğu gibi aşırı derece küçük kayıp olasılığına dayanan kontrol zorunluluğundan kaynaklanan temel nitelikteki güçlükten kaçınmaktadır. Bu avantajlarına rağmen, gecikme tabanlı yoğunluk kontrolüyle ilgili çözümlenmesi gereken, özellikle kademeli derleme ile ilgili sorunlar da mevcuttur. Biz burada bunun bir yönünü, yani ağ ECN-tabanlı bir AQM'e taşındığında VEGAS kaynağı nasıl sorunsuz çalışacağını tanımladık.

KAYNAKLAR

[1]. Lawrence S. Brakmo and Larry L. Peterson. TCP Vegas: end-to-end congestion avoidance on a global Internet. IEEE Journal on Selected Areas in Communications, 13(8):1465-80, October 1995. http://cs.princeton.edu/nsg/papers/j sac-vegas.ps.

[2]. J. S. Ahn, P. B. Danzig, Z. Liu, and L. Yan. Evaluation of TCP Vegas: emulation and experiment. In Proceedings of SIGCOMM'95, 1995.

[3]. U. Hengartner, J. Bolliger, and T. Gross. TCP Vegas revisited. In Proceedings of IEEE Infocom, March 2000.

[4]. Thomas Bonald. Comparison of TCP Reno and TCP Vegas via fluid approximation. In Workshop on the Modeling of TCP, December 1998. http://www.dmi.ens.fr/\%7Emistral/ tcpworkshop. html.

[5]. J. Mo, R. La, V. Anantharam, and J. Walrand. Analysis and comparison of TCP Reno and Vegas. In Proceedings of IEEE Infocom, March 1999.

[6]. C. Boutremans and J. Y. Le Boudec. A note on the fairness of TCPVegas. In Proceedings of International Zurich Seminar on Broadband Communications, pages 163-170, February 2000.

[7]. Jeonghoon Mo and Jean Walrand. Fair end-to-end window-based congestion control. IEEE/ACM Transactions on Networking, 8(5):556-567, October 2000.

[8]. Steven H. Low, Larry Peterson, and Limin Wang. Understanding Vegas: a duality model. J. of ACM, 49(2):207-235, March 2002.

[9]. Steven H. Low. A duality model of TCP and queue management algorithms. In Proceedings of ITC Specialist Seminar on IP Traffic Measurement, Modeling and Management (updated version), September 18-20 2000. http://netlab.caltech.edu.

[10]. Frank P. Kelly, Aman Maulloo, and David Tan. Rate control for communication networks: Shadow prices, proportional fairness and stability. Journal of Operations Research Society, 49(3):237-252, March1998.

[11]. Steven H. Low and David E. Lapsley. Optimization flow control, I: basic algorithm and convergence. IEEE/ACM Transactions on Networking, 7(6):861-874, December 1999. http://netlab.caltech.edu.

[12]. L. Massoulie and J. Roberts. Bandwidth sharing: objectives and algorithms. In Infocom'99, March 1999. http://www.dmi.ens.

[13]. Srisankar Kunniyur and R. Srikant. End-to-end congestion control schemes: utility functions, random losses and ECN marks. In Proceedings of IEEE Infocom, March 2000. http: //www. ieee- infocom. org/2 0 00/papers/4 01 .ps.

[14]. Fernando Paganini, John C. Doyle, and Steven H. Low. Scalable laws for stable network congestion control. In Proceedings of Conference on Decision and Control, December 2001. http : //www. ee. ucla.edu/"paganini.

[15]. Glenn Vinnicombe. On the stability of end-to-end congestion control for the Internet Technical report, Cambridge University, CUED/F-INFENG/TR.398, December 2000.

[16]. Glenn Vinnicombe. Robust congestion control for the Internet. Submitted for publication, 2002.

[17]. Fernando Paganini, Zhikui Wang, Steven H. Low, and John C. Doyle. A new TCP/AQM for stability and performance in fast networks.

[18]. S. Kunniyur and R. Srikant. A time-scale decomposition approach to adaptive ECN marking. IEEE Transactions on Automatic Control, June 2002.

[19]. Sanjeewa Athuraliya and Steven H. Low. Optimization flow control with Newton-like algorithm. Journal of Telecommunication Systems, 15(3/4):345-358, 2000.

[20]. Chris Hollot, Vishal Misra, Don Towsley, and Wei-Bo Gong. A control t heoretic analysis of RED. In Proceedings of IEEE Infocom, April 2001. http://www-net.cs.umass.edu/papers/papers.html

[21]. S. H. Low, F. Paganini, J. Wang, S. A. Adlakha, and J. C. Doyle. Dynamics of TCP/RED and a scalable control. In Proc. of IEEE Infocom, June 2002. http://netlab.caltech.edu.

[22]. Ki Baek Kim and Steven H. Low. Analysis and design of AQM based on receding horizon control in stabilizing TCP. Caltech Technical Report, caltechCSTR:2002.009, March, 2002. Submitted for publication, 2002.

[23]. C. A. Desoer and Y. T. Yang. On the generalized Nyquist stabilitycriterion. IEEE Trans. on Automatic Control, 25:187-196, 1980.

BÖLÜM 7

TCP VEGAS ARACILIĞI İLE MODELLEME

TCP VEGAS ın önceki analitik modelleri kayıpsız ağlar için geliştirilmiştir. Bu bölüm TCP VEGAS ın basit ve uygun analitik modelini geliştirmiştir. Paket kayıplarıyla huzur içinde büyür. Benzer modeller TCP RENO için daha önce geliştirilmiştir. Bunla birlikte RENO nun tamamlayıcılarından daha farklı olarak VEGAS için çeşitli yaklaşımların farklı olarak davranmaya ihtiyacı vardır. Amaçlanan model değişik bir mekanizma sunar ki yavaş başlama, sıkışıklık giderme ve sıkışıklık düzeltme sırasında VEGAS ı çalıştırır. Bu sonuçlar basit TCP VEGAS ın geçerli modelini ihtiva eder bununla eşitlik temelli UDP akımları, belirlemek için basit formüller, ölçülmüş paket kayıp değerlerinden, ister ağ tamponları aşırı iletilmiş olsun isterse TCP VEGAS akışı hedeflenen değere ulaşmış olsun gibi başka akımların değer kontrolleri için kullanılabilir.

Geçmişte araştırmalar tek TCP akımının çeşitli sayıdaki analitik modelini Tur zamanın fonksiyonu ve paket kayıp değeri olarak arz etmiştir. Bu modeller bu ağ parametrelerinde Örneğin UDP akışları kaynak [1], [2] deki gibi TCP performansının duyarlılığını anlamamızı sağlamıştır ve diğer tip internet akışlarını kontrol etmek için çeşitli ilerlemelerin sağlanmasında kullanılmıştır. Bu bütün modeller TCP nin değişik geniş yerleştirmelerini adresler, TCP RENO olarak adlandırılan örneğin kaynak [3], [4], [5], [6], [7] deki gibi.
TCP nin başka bir değişkeni TCP VEGAS olarak arz edilir. VEGAS birçok yeni tekniği görevlendirir ki birlikte belirli gelişmeler sonuçlanabilir. Bunun la aynı zamanda paket kayıpları azalır. Bu gelişmelerden bir kısmı geçmişte uygulanmıştır. Bazı durumlarda alternatif mekanizmalar kullanmak TCP nin farklı formlarında örneğin TCP VEGAS, TCP yeni RENO gibi sadece pencere boyutunu bir kez küçültür çoklu paketle aynı pencereden düştüğü zaman. TCP RENO pencere boyutunu gönderilen her 3 lü çiftleme ACK si için küçültür. Bazı diğer yenilikler hala iyi anlaşılamamıştır ve geniş olarak yayılmamıştır. Örneğin VEGAS sıkışıklığı giderme algoritması bazı anahtar avantajlara sahiptir. Paket kayıplarından sakınmada olduğu gibi aynı zamanda bağlantılara karşı eğimi düşürerek gecikmeleri daha uzun üretmeyle yapar. Bu modellerin bütünü TCP nin bir değişkeni tarafından geliştirilmiş geniş çapta kullanılanıdır ve TCP RENO olarak adlandırılır.

TCP nin diğer bir elemanı TCP VEGAS olarak ifade edilebilir. VEGAS yeni birkaç tekniği çalıştırır ki birlikte uygun ilerleme sağlanır daha az paketleri oluşur. Bu ilerlemelerden bazıları alternatif mekanizmalar kullanılarak önceden bazı durumlarda uygulanmıştır. Örneğin VEGAS gibi TCP yeni RENO aynı pencereden çiftli paketler düşürüldüğünde sadece pencere boyutunu bir kere düşürür. TCP RENO her üçlü çift ACK (bilgilendirme) alındığında pencere boyutunu düşürür. Bazı diğer yaklaşımlar hala tam olarak anlaşılamamıştır ve geniş bir şekilde kullanılmamaktadır. Örneğin VEGAS sıkışıklık giderme algoritması Paket kayıplarını önleme yönünde bazı anahtar avantajlara sahiptir. TCP VEGAS ın performansı başka akış tipleri ile etkileşimi olan karışık ağ ortamlarında tam olarak anlaşılamamıştır. Önceleri benzetimle kullanılmıştır veya VEGAS ın analitik modellerinin davranışları kayıpların olmadığı ortamda TCP VEGAS ın analitik modeli paket kayıplarının durumuna göre takip

eder, başka akımlar ile ağ paylaşımı tarafından bizim bilgimizde dâhil olmayan daha önceden amaçlanmış protokol performansını ve mekanizmayı anlamamızda önemli bir araç olabilecek.

TCP VEGAS ın çeşitli gösterimleri RENO daki karşılıklarından biraz farklı olarak davranmaya ihtiyacı vardır. Bu sıkışıklık belirlemeyi ve giderme algoritmalarını içerir ve yeni bağlantılar düzeltme mekanizmaları içerir. Protokollerin bu özelliklerini yakalamak için statik eşit zaman girdilerine akışları böleriz ve rasgele çeşitli girişler aracılığı ile kapalı form çözüm üretiriz. Hem çiftli ACK ler hem de zaman aşımlarının formlarında kayıp işaretlemeleri maksimum pencere boyutunun etkisi ile modellenmiştir kaynak [8]

Bu model zamanda VEGAS ın mekanizmasını yeni bir set olarak birleştirerek dereceli geliştirilmiştir. Bu VEGAS tarafından görevlendirilen farklı mekanizmaların arkasındaki sezgiyi analitik olarak karakterize ederek test etme imkânı bize sağlar. Aynı zamanda ölçülmüş paket kayıpları olasılığından TCP VEGAS akımı hedeflenen düşük eşik değerini sağlayıp sağlamadığını belirlemek için kapalı form ifadeyi elde ederiz.

7.1. Temel

Bu bölüm açıkça TCP VEGAS ın yeniliklerini göstermektedir. İlk önemli yaklaşım TCP RENO dan açıkça ayrılan VEGAS sıkışıklık giderme mekanizmasıdır. TCP RENO ağ da sıkışıklık olduğunu gösteren sinyali paketlerin kaybı olarak kullanır. Gerçekte RENO bağlantının mümkün olan bant genişliğini bulabilmek için kayıplar yaratmaya ihtiyacı vardır. VEGAS ın amacı sıkışıklığı önceden belirlemektir. Ve daha sonra paket kayıplarının olabilme olasılığını önlemeye çalışarak azaltmaktır.

Ağ sıkışıklığı belirlemek için her tur zamanı RTT, TCP VEGAS hali hazırdaki pencere boyutu (W) kullanır. En yakın RTT ve minimum RTT kısa sürede belirlenir hesaplamak için

$$diff = \left(\frac{W}{baseRTT} - \frac{W}{RTT} \right) baseRTT = W \frac{RTT - baseRTT}{RTT} \tag{7.1}$$

(RTT – temel RTT) toplam yol sorgu gecikmesi olduğunda ve W/RTT hali hazırdakinin yaklaşığı olduğunda bu iki değerin ürünü ağda geri bilgilendirilen bu akışın paketlerin sayısının yaklaşığıdır. VEGAS sıkışıklık giderme algoritmasının amacı bu sayıyı 2 eşik değeri tarafından belirlenen alfa ve beta sabit bir aralıkta tutmaktır. Böylece bir kere bütün RTT yavaş başlama madunda değilken TCP VEGAS pencere boyutunu aşağıdaki gibi ayarlar

$$W = \begin{cases} w+1 & , diff < \alpha \\ w & , \alpha \le diff \le \beta \\ w-1 & , diff > \beta \end{cases} \tag{7.2}$$

Alternatif olarak fark temel RTT kaynak [9] ile α ve β eşik değerlerinin paketlerin standart birimlerinde belirlendiği durumda böylece bu sonuçlar bağlantının eşit olmayan davranışlarında farklı temel RTT ile bölünebilir kaynak [10]. Paketlerin birimlerinde ki fark ve eşik değerlerine sahip olduğunun farkında olduğumuz bütün VEGAS uygulamaları ve benzetimleri bu yazının hatırlatıcısı olduğu iddia edilir. Eşik değerlerinin her versiyonu sıkışıklık istemeksizin elde edilebilir bant genişliğini yararlı hale getirebilmek için değer göndererek ayarlama yapar.

VEGAS ın diğer bir özelliği RENO dan daha tutucu olarak yavaş başlama hareketi ile geliştirilebilir olmasıdır. Özellikle VEGAS eğer fark eşik değerini aşmışsa (veya kayıp oluşmuşsa)bütün diğer RTT ler için farkı kontrol eder ve yavaş başlama ihtiva eder. Bunun dışında pencere büyüklüğü çiftlemiştir. Basitlik için bu yazının kalanında $\gamma = (\alpha + \beta)/2$ ü ileri süreceğiz. Bu algoritma VEGAS ın Aktif olan sıkışıklık belirlemeleri ve kayıp giderme mekanizmaları diğer bir durumudur. Pencereyi çiftlemek bütün diğer RTT lerde aynı zamanda iyi bir temel RTT nin ölçümünü belirlemeyi sağlar.

TCP VEGAS da son 4 yeni mekanizma sıkışıklık düzetme mekanizmasıdır. İlki 2 paketin pencere boyutu başlangıç sırasında ve zaman aşımından sonra kullanılmıştır. İkincisi VEGAS her bir paket gönderildiğinde ve çift bilgilendirme geldiğinde zamanı kaydeder eğer çok önceden gönderilmişse uygun tane zamanlayıcı değeri belirlenir. Gönderici eski bilgilendirme paketini yeniden iletir. RENO daki gibi 3 lü çiftli ACK her zaman paket yeniden gönderimlerinde sonuç verir ama uygun tane zamanlayıcı daha önceden kayıpları belirler. 1 veya 2 çiftli ACK den hemen sonra yeniden gönderimleri paketlemeye yöneltir. Eğer yeniden gönderim oluşuyorsa bir sonraki 2 normal ACK nin her birinde aynı zamanda bilgilendirilmemiş en eski paketin yeniden gönderimini tetikler eğer uygun tane zamanlayıcı süresi dolmuşsa. Not edin ki uygun tane zamanlayıcının süresinin dolmasından dolayı paket yeniden gönderimi belirli ACK ler almaya devam eder. 3.su paket yeniden gönderimi çiftli ACK tarafından tetiklendiğinde sıkışıklık pencere boyutu sadece hali hazırdaki RTT den son pencere boyutu azalması daha az olduğunda azalır. Yeniden gönderimden sonra çiftli olmayan ACK tarafından tetiklenir, pencere boyutu küçülmez. Not edin ki çoklu kayıplar tek pencerede meydana geldiğinde VEGAS sıkışıklık pencere boyutunu bu ilk kayıplar için düşürür. 4. sü kayıplardan dolayı pencere küçültülür, çiftli ACK tarafından belirlenir VEGAS pencere boyutunu RENO da ki gibi %50 yerine % 25 küçültür.

Eğer kayıp bölümü yeterince fazla ise uygun tane zamanlayıcı tetiklenmesi için kontrol eder, hiçbir ACK gönderilmez. Kayıplar RENO tipi baya tane zaman aşımı tarafından belirlenir. Bu yazının hatırlatıcısında belirtilmediği sürece zaman aşımı terimi bayağı tane zaman aşımı olarak gösterilir.

7.1.2. İlgili Çalışmalar

Birçok analitik modelde tek TCP RENO nun durumu için ölçülmüş kayıp değeri ve ortalama RTT nin fonksiyonu olarak transfer edilir. Mathis et al. [6] da zaman aşımlarını ihmal ederek TCP RENO nun sıkışıklık gidermesini analiz etmiştir. Pad-hye et al. Kaynak [5] de zaman aşımlarını ihtiva eden daha tamamlayıcı bir çalışma sağlamıştır onların sonuçlarını kullanarak ve başlangıç- yavaş başlamayı ihtiva ederek analizde Cardwell et al. [3] TCP RENO transferinin keyfi boyutunun yaklaşık model ortaya çıkarmıştır. Kaynak [5] deki model Goyal et al. in kaynak [7] tarafından yeniden ziyaret edilmiştir. Ve yeniden incelenen versiyonu sunulmuştur. Farklı bir gelişme Misra et al. [4] tarafından yapılmıştır. TCP RENO nun kararlı davranışı akıcı analiz kullanılarak modellenmiştir. Bizim modelleme gelişmemiz kaynak [5] dekine benzerdir. Burada her tur temelinde akışı analiz ederiz. Başka taraftan bizim gelişmemiz VEGAS ın çok farklı davranışlarının analizinden farklıdır kaynak [9], [11], [12]. Bu model aşamalı olarak VEGAS mekanizmasının yeni bir seti ile ilişkilendirilerek geliştirilmiştir.

Değişken	Açıklama
α, β	Paketlerde ölçülmüş VEGAS ın eşik değerleri
γ	Yavaş başlamadan çıkmak için eşik değeri (α + β)/2
P	Kayıt bölümler arasında iletilmiş ortalama paketlerin sayısının tersi
$baseRTT$	Akış sırasında belirlenmiş Minimum tur zamanı
RTT	Keyfi tur zamanı
R	Transfer için ortalama tur zamanı
W	Zamanda keyfi bir noktada pencere boyutu
$W_{\max}$	Alıcı tarafından bildirilen maksimum pencere boyutu
W_0	Kararlı geri bildirim durumunda ortalama pencere boyutu
t_0	İlk TO nun TO serisinde Ortalama süresi

Tablo 7.1: Model notasyonu.

7.2. MODEL

Model notasyonu tablo 7.1 de özetlenmiştir. Model giriş parametreleri *R* ve *p*. *Temel RTT*, $W_{\max}$ t_0 α, β kaynak [3], [5], [7].

Daha önceki başarılı TCP RENO modeli boyunca TCP VEGAS davranışlarını daireler şeklinde modelledik. Data pencerelerinin her dönüşte iletildiği gibi ve dönüş süresi RTT ye eşit olduğu ve pencere boyutundan bağımsız varsayılır. İddia ediyoruz ki farklı turlarda oluşan paket boyutları bağımsızdır ama paket kaybolduğunda geriye kalan bütün paketlerde kaybolur.

Bir ileriki iddia temel RTT kısmen akış boyunca kararlı ise ihtiyaç duyulur ki rasgele seçilmiş girişler boyunca akış boyuna eşittir. Bölüm 4 deki deney gösterir ki bu iddia pratik olan ağ durumunda gerçekleşir. Eğer bu devam etmezse modeller akışın her kısmına uygulanabilir ve farklı temel RTT değerleri elde edilir

Aşağıda devam eden akışlar için ilk olarak TCP VEGAS ı düşünürüz ki paket kayıpsız olmasına sebep olur, zaman aşımı olmayan akışlar tarafından takip edilir, tek zaman aşımı olayı ardışık paket iletimleri için ile takip eder ve sonunda bu deneyimi taşır. Her durunda VEGAS ın genişletilmiş bir setini modelleriz ve bunlar için kapalı form ifade hesaplarız

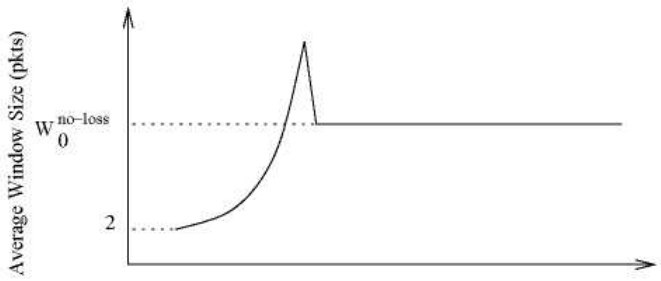
7.2.1 Paket Kayıpsız Model-1

Tahmin edilen TCP VEGAS ın pencere boyutunun dönüşümü şekil 7.1 deki gibi paket kayıpsız olduğunda akış yavaş başlama ile pencere ikiye eşit olmakla beraber başlar. Ve pencere boyutu fark her diğer RTT ler farkı γ yi aştığında veya $W_{\max}$ a pencere boyutu ulaştığında çiftlenir (genel durumda) . Bundan sonra akış sıkışıklık gidermeye kalır.

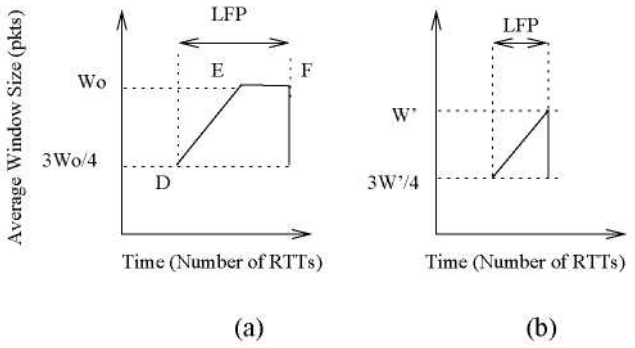
Bir nokta düşünün ki yavaş başlama periyodunda sonraki keyfi nokta olsun. $W_0^{n0-loss}$ noktadaki tahmin edilen pencere boyutunu temsin etsin. W gerçek pencere boyutunu temsil etsin ve RTT en son ölçülen tur zamanı olsun daha sonra zamanda bu noktadaki farkın değeri 1. eşitlikle verilsin.

İddia ediyoruz ki 2 neden den dolayı ortalama fark değeri yaklaşık beta olsun. İlki A-B VEGAS ın iyiliğini büyütür ve $\gamma = \beta$ uygulanır, başlangıç yavaş başlama periyodunda pencere boyutunun çiftlenmesi β DİFF i aştığında ortadan kaybolur. Daha fazla olarak ağdaki uygun sıkışıklık olmamsından kaynaklanan RTT fazla kararsız olmaz ve bu bir kere farkı β ya küçültür. Bu göreli sabit olarak kalmasını sağlar. Ağın çapraz trafiğinin geniş çeşitliliği ile TCP VEGAS ın genişletilmiş benzetimi sırasında belirlendiği gibi. İkinci olarak kayıpsız durumda RTT temel RTT civarında kararsız davranacaktır ve böylece sıkışıklık giderme algoritması mümkün olduğunca yüksek sorgulanmış paketlerin sayısını tutacaktır. Eşitlik 1 in her iki tarafında ki ihtimali alarak, RTT düşük değerli olduğu iddia edilir ve pencere boyutundan bağımsızdır, $W_0^{no-loss} = E/W$ için çözümdür ve maksimum pencere boyutu $W_{\max}$ ı hesaplar, (3) eşitliğini elde ederiz büyük transfer hesaplaması sırasında

$$W_0^{no-loss} = \min\left(\beta \times \frac{R}{R - baseRTT}, W_{\max}\right) \tag{7.3}$$



Şekil 7.1:Umulan pencere boyutu hesaplaması: kayıpsız.



Şekil 7.2: Umulan pencere boyutunun serbest kayıp olduğu zaman hesaplanması

Böylelikle yavaş başlama durumu sırasında ihmal edilebilir. Buda ortalamada TCP VEGAS $W_0^{no-loss}$ paketini her turda iletir. $W_0^{no-loss} < W_{\max}$ olduğunda eşitlik (7.4) ortaya çıkar

$$\Lambda_{no-loss} = \frac{\beta}{R - baseRTT} \tag{7.4}$$

Kayıp ihmal edilebilir olduğu zaman (bütün VEGAS ortamlarında) TCP VEGAS böylelikle yukarıdaki eşitlik tarafında yaklaştırılır. Bu eşitlik gösterir ki VEGAS bunları düşürmek için kullanılan ölçüm, a sıkışıklığını belirleyen veya elde edilebilir bant genişliğini belirleyen $R - baseRTT$ dir. Tahmin edilen sorgu gecikmesi aynı şişe boynu paylaşımı akımı için yaklaşık olarak aynı olması umulur, uygun temel RTT her akış için iddia edilir. Böylece denklem 4 de gösterildiği gibi kayıp ihmal edilebildiğinde TCP VEGAS büyük yayılma gecikmeleri ile akışa karşı maili yoktur RENO da olduğu gibi. Bu 21 ve 4 deki sonuçlar ile örtüşür ve ilerde belirtilecek olan 4.4.1 deki gibi

7.2.2 Zaman Aşımsız Model- 2

TCP kaynakları gibi RENO ile TCP VEGAS akımı şişe boynu bağlantısını paylaştığında veya kontrol edilemeyen çapraz trafikle kayıplar tecrübe edilebilir. Bu bölümde göstereceğiz ki bu tip kayıplar olur ama bütün kayıp parçaları çiftlenmiş ACK lar tarafından belirlenir (1 ile 3 arasında herhangi bir sayı). Kayıp bölümleri paketlerin serisi olduğu yerde tek turda kaybolur. Bu iddianın verilmesi, kayıp bölümler oluştuğunda VEGAS pencere boyutunu ¼ küçültmekle turda ilk kaybı belirlemiş gibi hareket edecektir. Daha iler ki paketler aynı turda kaybolacaktır. Çünkü daha sonraki pencere boyutunda hiçbir azalma olmayacaktır. Pencere boyutu düştüğünde VEGAS sıkışıklık gidermeye devam eder. Denklem 2 ye dayanarak pencere boyunu ayarlar.

Kayıp bölümleri ile kayıpsız periyotlara aralık diyeceğiz (LFPs). Ilk yavaş başlama periyodunu dikkate almadan büyük transfer sırasında ihmal edilebilir bir etkiye sahiptir. Akış uygun kesin LFPs serisini içerir. 2 durum düşünürüz: ilki p nin yeterince küçük değerleri için akış uygun geri bildirim durumuna ulaşır ki kayıpsız akışı karakterize eder, şekil 7.2a da gösterildiği gibi gerekli bütün LFP de. İkincisi büyük p için akış bu duruma asla ulaşmaz. Şekil 7.2b de gösterildiği gibi. Not edin ki p b deki durumdan a daki durama düştüğünde umulan maksimum pencere boyutu LFPs ler için W0 boyunca uygun geri bildirime ulaşamaz. Buda analizdeki basitlik için rasgele şekil 7.2a umulan pencere değişimini gösterir. Bunun dışında rasgele LFP nin umulan pencere boyunun değişimini gösterdiğini iddia ederiz. Bölüm 3.2.1 ve 3.2.3 bu durmun her biri için göreli olarak rasgele LFP nin değerini hesaplar. Daha fazla olarak akışın kesin analizi LFPs nin her tipinin katışımıdır. Daha ileri çalımlarda göz

atılabilir. Böylece model doğrulanması bu yazının daha sonraki bölümünde gösterir ki bu yaklaşık model gerçekçidir. Bölüm 3.2.2 model girişlerinden Ortalamada ister LFPs uygun geri bildirim durumuna erişsin bunu belirlemek için basit bir formül ortaya koyar. Bu formül modelde kullanılır ve eşitlik temelli değer kontrolünde de kullanılabilir, bölüm 3.2.1 den veya bölüm 3.2.3 deki formülden belirlenebilir.

7.2.2.1 Kararlı Geri Bildirimler Erişilebilir

Tahmin edilen pencere boyutu geri bildirilebilir durum sırasında (W_0) kayıpsız durumdakine benzer olarak çıkarılabilir. Bununla birlikte ağda ki sıkışıklığın derecesi kararsız olduğunda VEGAS ağda ki geri bildirimi ayarlar α ve β arasında ve böylece umulan fark değeri $(\alpha+\beta)/2$ye atanır. Beta yerine eşitlik (7.5) gider.

$$W_0 = \min\left(\frac{\alpha+\beta}{2}\times\frac{R}{R-baseRTT}, W_{\max}\right) \tag{7.5}$$

LFP nin devamını çıkarmak için ortalama paket sayısını LFP sırasında iletilen ortalama paket sayısını hesaplamak için DLFP ve LFP nin tahmin edilen süresinde ihtiyacımız vardır. 22 deki gibi benzer yapıda delilleri kullanarak LFP nin devamı bu iki önermenin oranıdır. P_{LFP} iki kayıp aralığındaki iletilen tahmin edilen paketlerin sayısı olarak ifade edilebilir (Örneğin *1/p*). Artı ilk gönderilen paketlerin ilk kayıp zamanı arasında iletilen paketlerin sayısı ve 22 de kayıpların gönderici tarafında belirlendiği zaman kaynak [5]:

$$P_{LFP} = \frac{1}{p} + W_0 - 1 \tag{7.6}$$

D_{LFP}yi hesaplamak için şekil 7.2a daki notasyonu kullanabiliriz. $D_{LFP} = D_{DE} + D_{EF}$ olsun. D den E ye durumu sırasında VEGAS ideal olarak her bir tur tarafından pencere boyutunu artırır. Ortalamada $W_0/4$ turu için

$$D_{DE} = \frac{W_0}{4}\times R \tag{7.7}$$

E den F ye durumu sırasında VEGAS her tur da W_0 paketlerin ortalamasını iletir. Böylece düşük değişiklik ideası bu durum sırasında pencere boyutu umulan tur sayısı bu durum içinde umulan iletilmiş paketlerin sayısına eşittir.

$$D_{EF} = \frac{P_{LFP} - P_{DE}}{W_0}\times R \tag{7.8}$$

olduğunda

$$P_{DE} = \sum_{\dot{I}=\frac{3W_0}{4}}^{W_0-1} \dot{I} = \frac{7{W_0}^2}{32} - \frac{W_0}{8} \tag{7.9}$$

eşitlik (7.6) ve (7.9) u kullanarak ve basitleştirerek eşitlik (7.10) elde edilir.

$$D_{LFP} = \left(\frac{1-p}{pW_0} + \frac{W_0}{32} + \frac{9}{8}\right)R \tag{7.10}$$

Sonuç olarak eşitlik (7.6) ve (7.10) birbirine bölünerek eşitlik (7.11) bulunur.

$$\Lambda_{no.TO}^{stable} = \frac{\frac{1-p}{p} + w_0}{\left(\frac{1-p}{pw_0} + \frac{w_0}{32} + \frac{9}{8}\right)R} \tag{7.11}$$

W_0 eşitlik (7.5) de verildiğinde not ederiz ki bu analiz TCP VEGAS için basit bir formüle gider. Bu durumda eşitlik (7.5) in devamında yerine koyarsak gösterir ki paket kayıpları oluştuğunda (örneğin ağda akılın diğer tiplerinden dolayı) daha uzun ortalana RTT ile bağlantılara karşı bazı eğilimler vardır. Bu eğilim basit bir karakteristiğe sahip değildir ama bölüm 4.4.1 de genel bir bakış atılacaktır.

7.2.2.2. Elde edilebilir kararlı geri bildiğim için durumlar

Eşitlik (7.11) sadece kayıp bölümler olduğunda medya gelir W W_0 a ulaştıktan sonra. Bu da $P_{DE} \le (1/p + W_0 - 1)$ eşitlik (7.9) kullanılarak:

$$p = \frac{32}{7W_0^2 - 36W_0 + 32} \tag{7.12}$$

eğer (7.12) eşitliği (7.5) ile beraber kullanılırsa ilerde sunulacak analizi kullanmış oluruz.

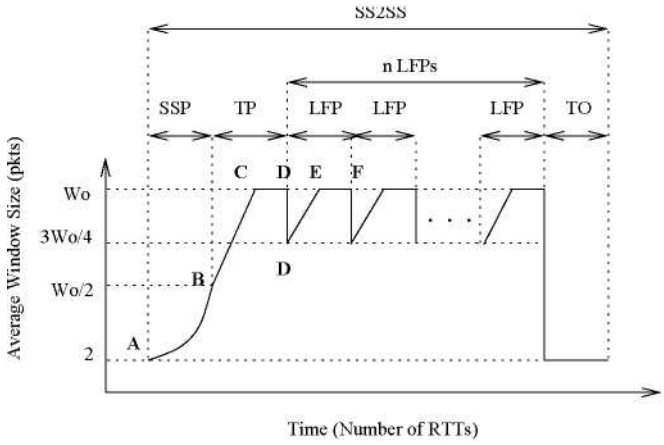
7.2.2.3. Ulaşılamaz uygun geri bildirim

Kayıp bölüm oluştuğunda ortalamadan uygun geri bildirimden önce ulaşılırsa VEGAS ın davranışı şekil 7.2b de resmedildiği gibi RENO ya benzerdir. Sıkışıklık giderme mekanizması RENO ya döner böylelikle uygun farklılıklar vardır sıkışıklık düzeltme mekanizmasında W' hesaplamak için not edin LFP sırasında VEGAS ideal olarak pencere boyutunu her bir turda bir kere büyültür. (P'_{LFP}) is $1/p + W' - 1$

$$\sum_{i=\frac{3W'}{4}}^{W'} i = \frac{1}{p} + W' - 1 = P'_{LFP} \Rightarrow W' = \frac{2 + 2\sqrt{\frac{56}{p} - 55}}{7} \tag{7.13}$$

iletilen paketlerin umulan sayısı $1/p + W' - 1$ dir. Böyle eşitlik (7.13) oluşur LFP de umulan turların sayısı $W'/4$ dür.

$$\Lambda_{no.TO}^{stable} = \frac{P'_{LFP}}{D'_{LFP}} = \frac{\frac{1-p}{p} + W'}{\frac{W'}{4}R} = \frac{4\sqrt{\frac{56}{p} - 55} + \frac{14}{p} - 10}{(1 + \sqrt{\frac{56}{p} - 55})R} \tag{7.14}$$



Şekil 7.3. SS2SS devir örneği.

Yukarıdaki eşitlik gösterir ki eğer kararlı geri bildirime asla ulaşamamışsa Ortalama yayılma gecikmesi ile ters orantılıdır TCP ren odaki gibi. Genel olarak TCP VEGAS ın bağımlılığı böylece (R) de RENO daki gibi açık değildir. İki uç noktası vardır. Birincisi *P=0* daki gibi ve VEGAS böylece R ye bağlı değildir. İkincisi p yeterince geniş olduğu durumda uygun geri bildirime asla ulaşılamaz ve VEGAS burada R ye ters orantılıdır. Bu iki uç nokta için p arttığında bunun bağımlılığı R de daha güçlü olur ter orantılı bağımlılığa ulaşana kadar.

7.2.3 Bir Tek Zaman Aşımlı Model-3

TCP VEGAS ın yeni yaklaşımı zaman aşımlarının modeli olarak uygulanacaktır. Bu durumda kayıp bölümler çiftli ACK ler ve zaman aşımları tarafından belirlenecektir. Zaman aşımları kayım bölümlerinden sonra oluşmuşsa yeterli çiftli ACK kayıp paketlerin yeniden gönderimini tetiklemek için göndericiye yeterli değildir.

İşlenmemiş tane zamanlayıcı paket için süresi geçtiğinde VEGAS t periyodu için bekleme konumunu alır ve daha sonra yavaş başlamaya gider ve pencereyi iki olarak ayarlar. Her RTT de iki kez yumuşatılmış RTT ortalaması artı 4 kez RTT değişkeni olarak *To* hesaplanır.

Burada iddia ediyoruz ki bütün zaman aşımı serileri tek bir zaman aşımını içerir. Biz ilk olarak VEGAS kararlı geri bildirim durumunda olduğunda bütün kayıp bölümlerin oluştuğu durumu analiz ettik. Bu senaryo altında akışın davranışı birbirine yakın istatistiksel olan uygun aralıklara Şekil 7.3 de pencere boyutu değişiminin umulduğu duruma bölünebilir. Bu aralıkların her birine yavaş başlamadan yavaş başlamaya periyodu deriz (SS2SSS). Rasgele SS2SS periyodunu burada çıkarmak için tahmin edilen iletilmiş paketlerin sayısını ve periyodun umulan süresini hesaplarız. Bunu yapmak için SS2SS i aşağıdaki periyotlara böleriz (şekil 7.3): (1) yavaş başlama periyodu (SSP) ikide pencere boyutunun başladığı gibi ve bütün diğer turları çiftlediği gibi daha sonra yavaş başlama eşik değerine ulaşır. Eşitlik (7.2) iletim periyodu (TP) kararlı geri bildirim durumuna ulaşana kadar bir tur tarafından pencere boyutu artırıldığı sırada ve akım kararlı geri bildirim durumunda kayıp bölümler oluşana kadar. Eşitlik (7.3) eğer TP TO (time out) ile bitmezse birbirini takip eden kayıpsız periyotlar (LFPS) serisi ilk olarak n–1 LFPS ile takip eder çiftli ACK ler tarafından belirlenen kayıp bölümler ile bitirilir ve n. LFP "kaypsız devir" TO tarafından belirlenen kayıp bölümlerle biter. Eşitlik (7.4) tek zaman aşımı. Buda

$$\Lambda_{SS2SS} = \frac{P_{SSP} + P_{TP} + nP_{LFP} + P_{TO}}{D_{SSP} + D_{TP} + nD_{LFP} + D_{TO}} \tag{7.15}$$

P_X X ve D_X ortalama periyodun süresi periyodunda ortalama iletilen paketlerin sayısı olarak ele alınır. Bu terimler SS2SS periyodunun bütün bileşenleri için bundan sonraki 3 bölümde tek tek çıkarılacaktır.

7.2.3.1 Yavaş Başlama ve İletim fazı (TP)

SSP(yavaş başlama periyodu) ve TP(iletim periyodu) şekil 7.3 de *a* noktaları ile *b* noktaları ve *b* ile *d* arasında ayrı ayrı olarak gösterilmiştir. A dan D ye periyodu birbirini takip eden kayıp bölümleri ile başlar ve biter
Böylece

$$P_{AD} = P_{SSP} + P_{TP} = \frac{1}{P} + W_0 - 1 \tag{7.16}$$

Pencere boyutu iki olduğunda yavaş başlama başladığında ve yavaş başlamanın eşik değerine ulaşılana kadar her diğer RTT de çiftlenir

$$\begin{aligned} P_{SSP} &= 2(2 + 4 + \cdots + \frac{W_0}{4}) = 2\sum_{i=0}^{\log\frac{W_0}{4}} 2^i = 2^{\log W_0} - 4 \\ veya \quad D_{SSP} &= 2(\log\frac{W_0}{4})R = 2(\log W_0 - 2)R \end{aligned} \tag{7.17}$$

başlangıç pencere boyutu Wo/2 nin dışındayken rasgele TP nin analizi LFP nin analizine benzerdir. Umulan. Böylece b ve c için şekil 7.3 de

$$P_{BC} = \sum_{i=\frac{W_0}{2}}^{W_0-1} i = \frac{3W_0^2}{8} - \frac{W_0}{4} \quad veya \quad D_{BC} = \frac{W_0}{2}R \tag{7.18}$$

durumun (c den d ye) umulan süresi P_{CD}/W_0dür. $P_{CD} = P_{AD} = P_{SSP} - P_{BC}$ olduğunda eşitlik (7.16-7.18) i kullanarak

$$\begin{aligned} D_{TP} = D_{BC} + D_{CD} &= \frac{W_0}{2}R + \frac{P_{AD} - P_{SSP} - P_{BC}}{W_0}R = \\ &= \left(\frac{1-p}{pW_0} + \frac{W_0}{8} + \frac{5}{4} + \frac{4 - 2^{\log W_0}}{W_0}\right)R \end{aligned} \tag{7.19}$$

7.2.3.2 Kayıpsız Periyot Serileri (LFPS)

Eşitlik (7.6) ve (7.10) LFP de gönderilen paketlerin umulan sayısını, umulan süresini ayrı ayrı olarak verir. *n* için bir formül çıkarmak için umulan LFPS lerin ardışık sayısı SS2SS periyodunda, şekil 7.3 den not ederiz ki kayıp bölümlerin kırılması TO (Time out), P_{TO} tarafından belirlenir. $p_{TO} = 1 - (n+1)$ olarak verilmiştir. *n* için bunu çözerek

$$n = \frac{1 - p_{TO}}{p_{TO}} \tag{7.20}$$

P_{TO} olasılığı TCP RENO için çıkarılmıştır. Kayıp bölümlerden sonra 3 çiftli ACK lerden az olma olasılığı analiz edilerek çıkarılmıştır. VEGAS kilit farkı vardır ki paketler 3 çiftli ACK den daha az yetinebilir. Zaman aşımı meydana gelme olasılığını düşürebilir. 14 de

RENO ya oranla TCP VEGAS ın performansı yüksek kazancı katkıda bulunmadığı iddia ediliyor. Bununla birlikte benzetimlerimizin sonuçları gösteriyor ki kayıplarımızın dışında çiftli ACK ler tarafından belirlenmiş büyük çoğunluk bir ve ya iki ACK den sonra belirlenir kaynak [12] ve kayıp değerleri bir çiftli ACK nin %5 den daha büyüktür sadece tek bit çiftli ACK birçok durum için kaybı belirlemeye yeterlidir. P_{TO} nun basit analizi aşağıdaki sonuca sevk etmiştir. İddia eder ki çiftli ACK tarafından bütün kayıplar gelen ilk çiftli ACK tarafından belirlenir. *A(w,k)* w dışında k paketlerinin bilgilendirildiğin olasılığı olarak ele alınsın. *C(w,k)* pencere boyutu ile turda kayıp bölümler olduğu verilsin. W nun turundan alınmış k paketlerinin olasılığı olsun böylece

$$A(w,k)=\frac{(1-p)^k p}{1-(1-p)^w}\quad C(w,k)=\begin{cases}(1-p)^k p & ,k<w\\ (1-p)^w & ,k=w\end{cases}$$

kayıp bölümlere sahip olan *W* paketlerinin turu verilsin. Çiftli ACK siz olmaya yönelten senaryo ve böylece turda zaman aşımı 1: bütün pencere kayıp tur veya 2: *w* siz *i* paketleri alıcıya ulaşır, alıcı i ACK leri gönderir. Gönderici i yeni paketleri gönderir ve bütün *i* paketleri kaybolur. W_0 ın tekrar çağrılması kayıp bölümler oluştuğunda umulan pencere boyutundadır.

$$\begin{aligned}p_{TO}(W_0)&=\min\left(1, A(W_0,0)+\sum_{i=1}^{W_0-1}A(W_0,i)C(i,0)\right)=\\ &=\min\left(1,\frac{p+p(1-p)(1-(1-p)^{W_0-1})}{(1-(1-p)^{W_0}}\right)\end{aligned}\tag{7.21}$$

7.2.3.3. Zaman Aşımı

TO (time-out) sırasında hiçbir paket iletilmez böylece

$$p_{TO}=0\qquad,\qquad D_{TO}=T_0\tag{7.22}$$

Eşitlik (7.6), (7.10), (7.16), (7.17), (7.19) ve (7.22) eşitlik (7.15) in yerine koyduğumuzda aşağıdaki yaklaşık TCP VEGASın akışının durumunu sadece bir TO oluştuğu durumda elde ederiz.

$$\Lambda_{SS2SS}=\frac{(n+1)\left(\frac{1-p}{p}+W_0\right)}{N_{SS2SS}R+T_0}\tag{7.23}$$

$$N_{SS2TO}=2\log W_0+(n+1)\frac{1-p}{pW_0}+(1+\frac{n}{4})\frac{W_0}{8}+\tag{7.24}$$

$$+\frac{9n}{8}-\frac{11}{4}+\frac{4-2^{\log W_0}}{W_0}\tag{7.25}$$

olduğunda

Bu ifade VEGAS ın daha önceki formülünden daha karışıktır. Zaman aşımsızlığı iddia eder. Ama r nin bağımlılığı bölüm 3.2.3 de açıklanmıştır. N nin değeri eşitlik (7.20), (7.21) ve (7.5) nolu eşitliklerle hesaplanabilir.

7.2.3.4 Kararlı Geri Bildirim

Yukarıdaki analizlerden eşitlik (7.12) ile birlikte tamamen düzeltmek için TO dan akış mümkündür. Ortalamada, kararlı geribildirim durumuna daha sonraki kayıp bölge oluşmadan önce ulaşılır. Buda $p_{AC} \le (1/p + W_0 - 1)$ Veya

$$p_{AC} \le \frac{8}{3W_0^{\,2} - 10W_0 + 16\log W_0 + 8} \tag{7.26}$$

dır.

Eşitlik (7.12) veya (7.26) bağımlılıklarının olmadığı zamanki durumlarda VEGAS ortalamada kesin kayıp bölgesi arasında kararlı geri bildirim durumuna ulaşamaz. Pencere değişimi bu durumda şekil 7.3 dekine benzer. TP ve LFPS keskin tepe noktalarına sahip olur. İfadeyi daha basitleştirmekle aşağıdaki eşitlik bu durumlarda VEGASın tahmin edildiği gibi kullanılır.

$$\Lambda_{SS2SS}^{not-stabil} = \frac{P'_{SSP} + P'_{TP} + nP'_{LFP} + P_{TO}}{D'_{SSP} + D'_{TP} + nD'_{LFP} + D_{TO}} \tag{7.27}$$

P'_{LFP} ve D'_{LFP} çıkarılan *SSP* ve *TP* için analizler bölüm 3.3.1 deki gibi uygundur. $W_0/2$ yerine pencere boyutu $W'/2$ iletimin iki faz arasında olduğunda yer alır. Asıl yaklaşım bu analizi tutmak için umulan pencere boyutu *TP* nin sonunda kayıp oluşuğunda bölüm 3.2.3 ile aynıdır LFP için örnek W_0.

7.2.4 Tam Model

TO (zaman aşımı, time-out) oluştuğunda sonraki TO lar ilkiyle arka araka oluşabilir. Burada iletilmiş tahmin edilen paket sayısını TO serileri sırasında ve umulan bu tip serilerin süresi P_{TO} ve D_{TO} terimi için daha doğru yaklaşımlar sağlamak adına formül 15 de çıkarabiliriz. TO tarafından takip edilen İki paketin çevrimi ile 3 durum belirleyebiliriz. A) ister paket kayıp olsun ister olasılık $\rho_0 = (1-p)^2$ olsun B) sadece ikinci paket kayıp olma olasılı $\rho_1 = (1-p)$ c) her bir paket kayıp olma olasılığı ile $\rho_2 = p$ (Verilen ilk kayıpla ikinci kayıp 1 olasılığı ile meydana gelir). Not edin ki olasılıklar toplamı 1 dir. 1 durumunda TO serilerinin sonunda sinyaller c yeni bir TO ya sebep olur. B durumunda yeni TO ya sebep olur eğer sadece bir fazladan paket tek ACK nin alıcı tarafından geri gönderilen cevabı olarak iletildiğinde, aynı zamanda kayıptır. Burada bizim analizimizi basitleştirmek için iddia ederiz ki bu her zaman olur. Örneğin b durumunda her zaman daha sonraki TO lara sebep olur. Böylece TO turları acilen takip eder. Olasılık daha sonraki TO lar oluştuğunda $\rho_1 + \rho_2$ dir. Rasgele TO serisinde M ardışık TO ların sayısı olsun. Daha sonra ilk TO verildiğinde

$$P[M = k] = (P_1 + P_2)^2 \times P_0 = (2p - p^2)^{k-1}(1-p)^2 \tag{7.28}$$

Ve umulan ardışık TO ların sayısı EM

$$E[M] = \sum_{k=1}^{\infty} kP[M = k] = \frac{1}{(1-p)^2} \tag{7.29}$$

dir

Pencere boyutu iki olduğunda her TO dan sonra $P_{TO-series} = 2(E/M)$ dir. Son TO dan sonra turdaki iletilen iki paketi hariç tutarız. Bir sonraki SSP (eşitlik (7.17) yi görünüz) ye tur dahil edildiğinde. Son olarak umulan iletilmiş paketlerin sayısı rasgele TO serileri sırasında

$$P_{TO-series} = \frac{2p(2-p)}{(1-p)^2} \tag{7.30}$$

İlk TO ların süresi T_O dır ve her yeni TO için süre ikiye katlanır. Süre 64 to a ulaşılana kadar. Daha sonraki TO larda süre sabit kalır. TO serilerin süresi aşağıdaki gibidir.

$$D_k = \begin{cases} (2^k - 1)T_O & ,k > 6 \\ (63 + 64(k-6)) & ,k \geq 6 \end{cases}$$

Umulan rasgele TO serilerinin süresi

$$\begin{aligned} D_{TO-series} &= \sum_{k=1}^{\infty} D_k P[M = k] = \\ &= \left(\frac{64}{(1-p)^2} - 321 + (1-p)^2 \sum_{k=1}^{6} (2^k - 64k + 320)(2p - p^2))^{k-1} \right) T_0 \end{aligned} \tag{7.31}$$

Tarafından verilmiştir. Yukarıdaki eşitlikten parantez içindeki değeri veririz d(p) gibi örneğin $D_{TO-series} = d(p)$Eşitlik (7.15) P_{TO} ve D_{TO} yı daha uygun bir yaklaşım olan $P_{TO-series}$ ve $D_{TO-series}$ yerine tek TO da koyarız. SS2SS periyodunun geriye kalan analiz açıkça daha önceki bölümde tasvir edildiği gibidir. Böyle 15 denkleminde

$$\Lambda_{loss} = \frac{(n+1)\left(\frac{1-p}{p} + W_0\right) + \frac{2p(2-p)}{(1-p)^2}}{N_{SS2TO}R + d(p)T_O} \tag{7.32}$$

yi elde ederiz.

n değişkeni eşitlik (7.20) ve (7.21) denkleminden hesaplanabilir. *SS2TO* eşitlik (7.25) de verilmiştir, W_0 eşitlik (7.5) de verilmiştir. Eşitlik (7.4) ve (7.32) denklemlerinin seti TCP VEGAS ın toplam Hem kayıpları hem de kayıpsız senaryoyu içeren modelini meydana çıkarır. Eşitlik (7.12) ve (7.26) durumlarında verilen bağımlılıklar sağlanır.

KAYNAKLAR

[1]. S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. In SIGCOMM 00, Stockholm, Sweden, August 2000.

[2]. M. Vojnovic and J.-Y L. Boudec. On the long-run behavior of equation-based rate control. In SIGCOMM 02, Pittsburgh, PA, August 2002.

[3]. N. Cardwell, S. Savage, and T. Anderson. Modeling TCP latency. In INFOCOM 00, Tel Aviv, March 2000.

[4]. V. Misra, W. Gong, and D. Towsley. Stochastic differential equation modeling and analysis of TCP-windowsize behavior. In Performance, Istanbul, Turkey, October 1999.

[5]. J. Padhye, V. Firoiu, D. Towsley, and J. Krusoe. Modeling TCP throughput: A simple model and its empirical validation. In SIGCOMM 98, Vancouver, CA, September1998.

[6]. M. Mathis, J. Semke, and J. Mahdavi. The macroscopic behavior of the TCP congestion avoidance algorithm. Computer Communications Review, 27(3), 1997.

[7]. M. Goyal, R. Guerin, and R. Rajan. Predicting TCP throughput from non-invasive network sampling. In INFOCOM 02, New York, NY, June 2002.

[8]. Ns-2 simulator, http://www.isi.edu/nsnam/ns

[9]. L. S. Brakmo, S. W. O'Malley, and L. L. Peterson. TCP Vegas: New techniques for congestion detection and avoidance. In SIGCOMM 94, London, UK, Sept. 1994

[10]. S. Low. A duality model of TCP and queue management algorithms. In ITC Specialist Seminar on IP Traffic Measurement, Modeling and Management 00, Monterey, CA, September 2000.

[11]. L. S. Brakmo and L. L. Peterson. TCP Vegas: End to end congestion avoidance on a global internet. IEEE Journal on Selected Areas in Communications, 13(8), 1995.

[12]. U. Hengartner, J. Bolliger, and T. Gross. TCP Vegas revisited. In INFOCOM 00, Tel Aviv, March 2000.

SONUÇ

Bu çalışmada TCP RENO üzerine geliştirilen VEGAS ın sıkışıklık sorununa getirdiği yaklaşım ele alınmıştır. Farklı protokoller sıkışıklık ölçüleri için farklı ölçüler kullanır. Örneğin RENO kayıp olasılığını kullanır ve VEGAS gecikmeyi sorgular.

VEGAS ağ kapasitesiyle yerleşik ölçeklenebilme özelliği nedeniyle özellikle yüksek hızlı ağlarda caziptir. TCP RENO nun sıkışıklık tespiti ve kontrol mekanizmaları parçaların kaybını bir işaret olarak kullanmaktadır. Bu parça kayıpları şebekede tıkanıklık olduğunu göstermektedir. Bu yüzden TCP RENO nun kayıplar olmadan önce tıkanıklığın başlangıç aşamalarını tespit edecek bir mekanizması yoktur. VEGAS yüksek bant genişliği rejiminde potansiyel sıkışıklık sorunu azaltılabilmektedir. Bunun yanı sıra, RENO' nun yapmak zorunda olduğu gibi aşırı derece küçük kayıp olasılığına dayanan kontrol zorunluluğundan kaynaklanan temel nitelikteki güçlükten kaçınmaktadır. Bu avantajlarına rağmen, gecikme tabanlı yoğunluk kontrolüyle ilgili çözümlenmesi gereken, özellikle kademeli derleme ile ilgili sorunlar da mevcuttur. REM temiz tampon ve değeri seçmeye çalışır ve yüksek yararlanma ve düşük sorguya yöneltir. Küçük sorular ile en az tur zamanı yayılma gecikmesi için en uygun yaklaşım olabilir.

VEGAS ın RENO algoritmasından farkı ağ kapasitesinin ne kadar olduğunu öğrenmek için sıkışıklığa teşvik etmesidir. VEGAS kaynağı sıkışıklık saldırısını, gerçekleşen ve beklediği arasındaki farkı göstererek bekler. VEGAS yol boyunca yönlendiricide daha az sayıda paket tamponlanmasını sağlamak için kaynağın gönderim boyutunu artırma stratejisidir.

TCP VEGAS ın sıkışıklık tespit mekanizması aktiftir, yani çıktı oranındaki değişiklikleri gözlemleyerek tıkanıklıktaki başlangıcı tespit etmeye çalışır. TCP VEGAS bu çıktı ölçümlerinden tıkanıklık penceresi ayarlama politikasını çıkarır, bu da bağlantı kayıplar vermeden önce gönderme oranını azaltabilmeyi sağlar.

Duality modelin temel fikri genel sıkışıklık problemlerini çözmek için ağ üzerinde kaynak ve hatlar tarafından taşınan dağıtılmış algoritmayı sıkışıklık kontrolü olarak yorumlamasıdır. Duality model VEGAS algoritmasına yeni bir yorum katmıştır. Kaynak yolundaki gecikmenin değerini sorgulamak için kendi gecikmesini belli bir oranda tutar. VEGAS bu gecikme oluşumunu minimum tam tur zamanı olarak kabul eder.

TCP VEGAS çeşitli değişik tekniklerin bir birleşimidir. Her bir teknik kendi başına bir tartışma konusudur. Daha önce yapılan tartışma ve çalışmalar ya yalnız belli bir mekanizma üzerinde yoğunlaşmış ya da TCP VEGAS ın bütün olarak davranışını değerlendirmeye çalışmıştır. Ancak asıl soru TCP VEGAS içerisindeki hangi tekniğin performans kazanımlarından sorumlu olduğudur.