TCP PERFORMANSININ VERİ TRANSFERİ UYGULAMALARI İÇİN GELİŞTİRİLMESİ

İhsan GÜNEŞ[1] Ali Yavuz ÇAKIR[2] Cüneyt AKINLAR[3]

[1,2,3]Bilgisayar Mühendisliği Bölümü
Mühendislik-Mimarlık Fakültesi, İki Eylül Kampüsü
Anadolu Üniversitesi, 26470, Eskişehir
[1]e-posta: ihsang@anadolu.edu.tr [2]e-posta: aliyavuz@anadolu.edu.tr
[3]e-posta: cakinlar@anadolu.edu.tr

Anahtar sözcükler: TCP, TCP Newreno, Flow Control, Congestion Control

ABSTRACT

Transmission Control Protocol (TCP), which offers a reliable, in-order packet delivery service, is at the heart of many Internet protocols including FTP, HTTP. It is a known phenemenon that in-order nature of TCP causes what is known as a head-of-line blocking problem. That is, when the first packet within a TCP sender window is lost, the window will not move forward until the packet is successfully transmitted and recovered at the receiver. In this paper we propose changing the service interface of TCP to reliable, unordered packet delivery and leave the packet reordering to the application layer. Thus the sender views the active window to consist of multiple independent logical channels and pumps data in from any empty channel. The receiver delivers the packets to the application in the order they arrive. It is then the the application's responsibility to order the packets. We present simulation results of this new TCP protocol using the popular "ns" simulator and show that the performance of data transfer applications improve around 30% compared to NewrenoTCP.

1. GİRİŞ

TCP [RFC 793] İnternet üzerinde iki host arasında güvenilir ve sıralı veri iletimini sağlayan en yaygın protokoldür [1]. Son yıllarda İnternet ağı önemli derecede büyüdüğü ve hızla büyümeye devam ettiği için TCP'nin İnternet üzerindeki performansının geliştirilmesi ile ilgili çalışmalar çok büyük önem kazanmıştır. TCP protokolü üzerinde yapılan iyileştirmeler sonucunda bir çok TCP versiyonu geliştirilmiştir. Bunlardan en yaygın olanları Tahoe, Reno, Newreno ve SACK TCP'dir [2]. Her yeni versiyonla TCP'nin sağlamış olduğu güvenilir ve sıralı veri iletimininin performansı artırılmaya çalışılmıştır. İnternet üzerinde FTP, HTTP, TELNET gibi TCP'yi kullanarak veri iletimi yapan bir çok uygulama vardır. Bu uygulamalardan bazıları veri iletimini hızlandırmak için bazı yöntemler kullanmaktadırlar. Örnek olarak bir FTP uygulaması olan FlashGet programı bir dosyayı İnternet üzerinden indirirken o dosyayı parçalara ayırır ve iki host arasında birden fazla TCP bağlantısı kurarak parçaların her birini farklı bağlantı üzerinden alır [3]. Her bir bağlantı üzerinden yine sıralı ve güvenilir veri iletimi yapılmaktadır. Yükleme işlemi bittiğinde veri parçaları sıralı olarak birleştirilerek veri bütünlüğü sağlanmaktadır. FlashGet'de veri iletiminin hızlandırılmasına karşın birden fazla TCP bağlantısı kurulması hostlar ve ağ üzerindeki yükün artmasına sebep olmaktadır.

Veri iletimini hızlandırmak için TCP gibi bir iletim protokolü olan SCTP (Stream Control Transport Protocol) geliştirilmiştir. SCTP, TCP'den farklı olarak veri iletimini hızlandırmak için bir bağlantı üzerinden birden fazla mantıksal veri akışı (Multistream) kullanır. Her bir veri akışı kendi içinde sıralı iletim yapmakta olup diğer veri akışlarından bağımsızdır [4].

Bu çalışmada, diğer protokollerden farklı olarak TCP üzerinde yapılan değişklikler sonucunda TCP'nin güvenilir fakat sıralı olmayan veri iletimi yapması sağlanmıştır. TCP'nin katı bir şekilde sıralı iletim yapmasından meydana gelen gecikmeler engellenerek veri iletimi hızlandırılmıştır. Sıralama işlemi uygulama katmanına bırakılarak TCP üzerindeki yük azaltılmıştır.

2. TCP PROTOKOLÜ NASIL ÇALIŞIR ?

TCP güvenilir olmayan IP (Internet Protocol) servisi üzerinde güvenilir ve sıralı veri iletimi sağlar. TCP güvenilir iletimi sağlarken kümülatif ACK (Acknowledge - Alındı Bilgisi) bilgisi ve bir tane tekrar gönderim zamanlayıcısı (Timer) kullanır. Aldığı veriyi onaylar ve kaybolan paketi tekrar gönderir [5]. TCP kaybolan veriyi zaman aşımı süresi dolmadan

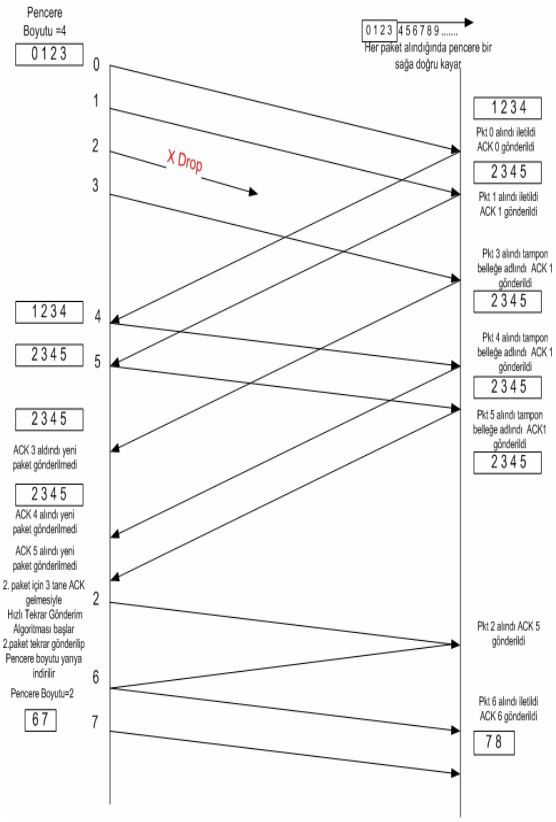
önce göndermek için Hızlı Tekrar Gönderim (Fast Retransmit) algoritmasını kullanır. Bu algoritma üç tane aynı ACK bilgisi alınması halinde ACK bilgisinin göstermiş olduğu paketi tekrar gönderir [6]. TCP, alıcının, kaybolan paketleri veya gelen aynı paketleri belirlemesini sağlamak için sıra numarası (sequence number) kullanır. Birden fazla paketi aynı anda göndermek için ardışık düzen (Pipeline) yöntemini kullanarak veri iletim hattının verimliliğini artırır. Gönderilecek paket sayısını, TCP akış denetimi (flow control) ve tıkanıklık denetimi (congestion control) algoritmalarını kullanarak belirler. Akış denetimi algoritması alıcının kabul edebileceği paket sayısını, tıkanıklık denetimi algoritması ise, göndericinin, ağın kullanılabilir bant genişliğine göre, iletebileceği paket sayısını belirler. Gönderilecek paket sayısı da bu iki değerin küçük olanı olarak belirlenir [7]. Bu gönderilecek paket sayısı pencere boyutu (window size) olarak ifade edilir.

3. TCP PROTOKOLÜ ÜZERİNDE YAPILAN DEĞİŞİKLİKLER

Çalışmamızda yaptığımız bazı değişikliklerden sonra TCP'nin güvenilir fakat sıralı olmayan veri iletimi yapması sağlanmıştır. TCP de aynı şekilde paket sıra numarası ve ACK bilgisi kullanarak güvenilirlik sağlamakta ancak alıcı tarafta paketlerin üst katmana sıralı olarak iletilip iletilmediğinin kontrolü yapılmamaktadır. Alıcı, paketlerin sıra numarasına bakmaksızın aldığı paketleri üst katmana iletip göndericiye ACK bilgisini göndermektedir. Paketlerin sıralanması işlemi tamamıyla uygulama katmanına bırakılmıştır. Bunun sonucunda TCP'de sıralama işlemi için meydana gelen gecikmeler azaltılarak veri iletimi hızlandırılmıştır. Uygulamada, TCP'nin güvenilir veri aktarım prokolünde veri akışını ve veri güvenilirliğini sağlayan Kayan Pencere Protokolü (Sliding Window Protocol) ile tekrar gönderimleri tetikleyen Hızlı Tekrar Gönderim (Fast Retransmit) algoritması değiştirilmiştir.

3.1 Kayan Pencere Protokolü

TCP'deki Kayan Pencere Protokol'ünün iki temel görevi; veri iletiminin güvenilirliği ve akış kontrolünü sağlamaktır. Kayan Pencere Protokol'ünde, her bir pakete sıra numarası verilir ve her bir paket alıcıya ulaştığında göndericiye ACK bilgisi gönderilir. Eğer gönderici kendisine gelen ACK bilgilerine bakarak bir paketin kaybolduğunu anlarsa o paketi tekrar gönderir. TCP, bir paketin kaybolduğunu, o paket için tutulan zamanlayıcının zaman aşımına uğraması veya o paket için üç tane aynı ACK bilgisinin gelmesi sonucunda anlar. Bu iki durumda da kaybolan paket tekrar yollanarak güvenilirlik sağlanır. TCP, Kayan Pencere Protokolü ile akış kontrolü sağlayarak veri bant genişliğini daha verimli kullanır. Alıcı, gönderdiği her ACK bilgisi içinde kendi kabul edebileceği byte sayısını (window size) alıcıya iletir. Burada pencere boyutu, göndericinin, pencere içerisindeki ilk paket için ACK alınmasını beklemeden gönderilebileceği paket sayısını gösterir. Bunun sonucunda, TCP'de, göndericinin, alıcının kabul edemeyeceğinden fazla byte veya paket göndermesi engellenmiş olur [5].

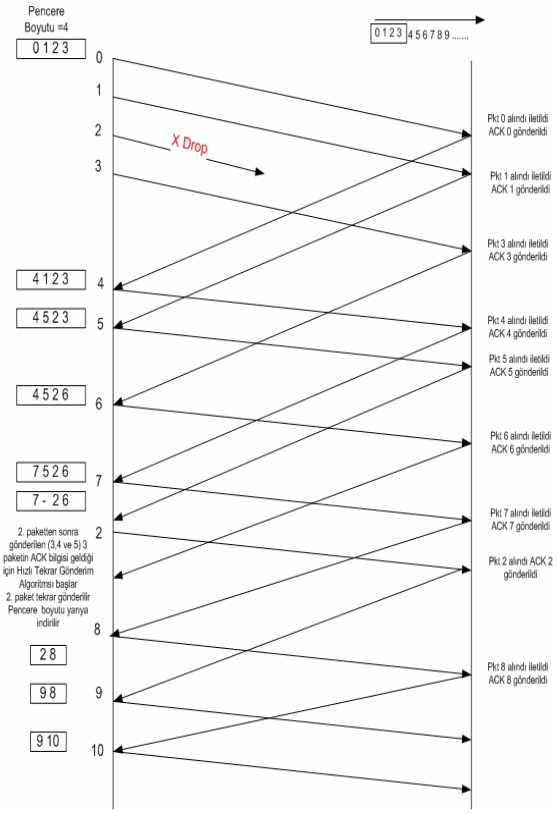


Şekil-1. Newreno TCP'nin Güvenilir Veri iletimi

Şekil-1'de Newreno TCP'nin güvenilir veri transfer protokolünün nasıl çalıştığı gösterilmiştir. Pencere boyutu 4 olarak belirlenmiş olup bir paket kaybolduğunda, Newreno TCP'nin kaybolan paketi tekrar göndermek için nasıl bir yöntem uyguladığı gösterilmiştir. Şekilde görüldüğü gibi her sıralı paket alındığında üst katmana iletilip bu paket için ACK bilgisi göndericiye iletilmektedir. Alınan her paketten sonra pencere bir adım sağa doğru kaymaktadır. Şekil-1'de 2 sıra numaralı paketin kaybolmuş olduğu varsayılmıştır. Bu nedenle 3, 4 ve 5 sıra numaralı paketler alındığı halde 2 sıra numaralı paket alınmadığı için pencere sağa doğru kaymayacak ve 2 numaralı paket tekrar gönderilene kadar yeni paket gönderimi yapılmayacaktır. Bu bekleme süresinde veri iletim hattının boş kalması, bant genişliğinin verimli kullanılmasını engellemektedir. Bu bekleme TCP'nin katı bir şekilde sıralı paket iletimi yapmasından kaynaklanmaktadır. 2 numaralı paket gönderilip ACK bilgisi geldikten sonra, pencere sağa doğru kayarak yeni paket gönderilmektedir. Alıcı, 2 numaralı paketin gelmediğini belirtmek amacıyla 2 numaralı paketten sonra gelen paketler için de ACK 1 bilgisini gönderir Kaybolan 2 numaralı paket için üç tane ACK bilgisi geldiğinde TCP'nin Hızlı Tekrar Gönderim

Algoritması çalışmakta ve pencere sayısı yarıya indirilerek 2 numaralı paket tekrar iletilmektedir. TCP üç tane aynı ACK bilgisinin alınmasından sonra pencere sayısını yarıya indirerek veri iletimine devam etmektedir [8].

3.2 Kayan Pencere Protokol'ünün Değiştirilmesi



Şekil-2. Değiştirilmiş Newreno TCP'nin Güvenilir Veri iletimi

Bu çalışmada, iki uç arasındaki bağlantının birbirinden bağımsız mantıksal kanallara ayrıldığı düşünülmektedir. Bu kanal sayısı dinamik olup pencere sayısına eşittir. Her bir kanal üzerinden paketler birbirinden bağımsız olarak gönderilmektedir. Bir kanaldan gönderilen paket için ACK bilgisi geldiği anda, o kanal üzerinden yeni paket diğer kanalların durumuna bakılmaksızın hemen gönderilmektedir. Değiştirilmiş olan Newreno TCP, güvenilir ve sıralı olmayan iletim yapmaktadır. Şekil-2'de görüldüğü gibi paketleri sıralama işlemi yapılmadığı için hiçbir bekleme söz konusu değildir. Alıcı, paketleri alır almaz paketin sıra numarasına bakmazsızın uygulama katmanına göndermektedir. Paketlerin her birinin sıra numarası olduğu için sıralama işlemi uygulama katmanı tarafından kolaylıkla yapılabilir. Şekil-2'de, Şekil-1'de olduğu gibi 2 numaralı paketin kaybolduğu varsayılarak, aynı durumda, geliştirilen yaklaşımda nasıl bir yöntem izlendiği gösterilmiştir. 2 numaralı paketten sonra gelen üç paket için ACK bilgisi alınması halinde, Hızlı Tekrar Gönderim Algoritması başlar ve 2 numaralı paket tekrar yollanır. Şekilde-2'de görüldüğü gibi 3 ,4 ve 5 sıra numaralı paketler için ACK bilgisi geldiğinde, 2 numaralı paketin kaybolmuş olduğu kabul edilir ve bu paket tekrar gönderilir. Pencere sayısı, dolayısıyla kanal sayısı yarıya indirilmiştir.

3.3 Hızlı Tekrar Gönderim Algoritması (Fast Retransmit)

Çalışmanın ikinci kısmında, TCP'nin Hızlı Tekrar Gönderim Algoritması üzerinde değişiklik yapılmıştır. TCP'nin ilk versiyonlarında, bir paketin tekrar gönderilmesi için o paket için tutulan zamanlayıcının zaman aşımına uğraması gerekiyordu. Bu sürenin çok uzun olması, kayıp paketin tekrar gönderilmden önce uzun beklemelere neden olmaktaydı. Bu bekleme süresini azaltmak için Hızlı Tekrar Gönderim Algoritması geliştirilmiştir. Hızlı Tekrar Gönderim Algoritması, zaman aşımı süresi dolmadan kayıp paketin tekrar gönderilmesini sağlar. TCP'de alıcı taraf, sırayı bozan bir paket aldığı zaman hemen en son sırada aldığı paket için bir ACK bilgisi gönderir. Bu ACK bilgisi, sırayı bozan bir paket alındığını ve hangi sıra numaralı paketin beklenildiğini gösterir. Ancak TCP, ilk önce almış olduğu aynı ACK bilgisinin kaybolan bir paketten mi, yoksa paketleri sıralama işleminden mi kaynaklandığını bilemez [6]. Üç tane aynı paket için ACK bilgisinin alınması durumunda, TCP, bu paketin kaybolmuş olduğunu kabul edip, bu paket için tutulan zaman aşımı süresini beklemeden paketi tekrar gönderir.

Geliştirilen Yaklaşımda, paketlerin sıralı olup olmadığı kontrol edilmediği için aynı ACK bilgisinin gönderilmesi söz konusu değildir. Göndericide ACK bilgisi gelmemiş en küçük sıra numaralı paket için zamanlayıcı tutulur. Eğer kendisinden sonra gelen 3 paketin alındı bilgisi geldiyse bu paket kaybolmuş olarak kabul edilir ve zaman aşımı süresi beklenmeksizin tekrar gönderilir.

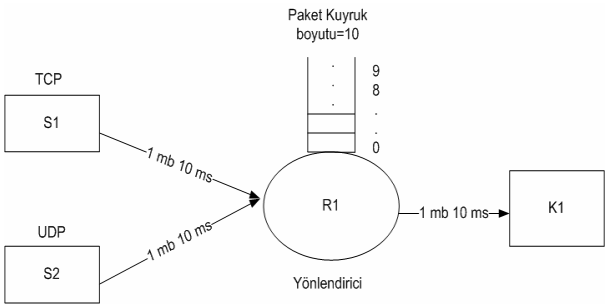
4. SİMÜLASYONLAR

Bu bölümde, Newreno TCP ve Değiştirilmiş Newreno TCP değişik simülasyon senaryoları üzerinde çalıştırılarak karşılaştırılmıştır. Simulasyon senaryoları, NS-2 (Network Simulator) programı üzerinde TCL scripti yazılarak oluşturulmuş ve çalıştırılmıştır. Simulasyonlarda veri trafiği göndericiden alıcıya doğru olmak üzere tek yönlüdür [9].

4.1 Simulasyon Senaryosu 1

Şekil-3'de görüldüğü gibi simülasyon senaryosu, iki gönderici (S1,S2), bir yönlendirici (R1) ve bir alıcıdan (K1) oluşmaktadır. Simülasyonda, S1'den K1'e FTP bağlantısı, S2'den K1'e UDP bağlantısı kurulmuştur. FTP (File Transfer Protocol) uygulaması, Newreno ve değiştirilmiş Newreno üzerinde ayrı ayrı çalıştırılarak

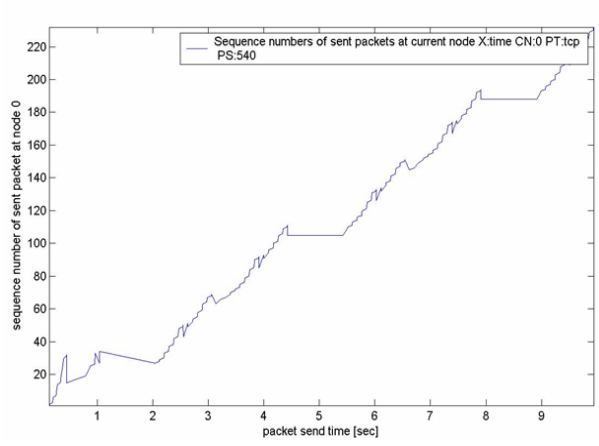
elde edilen sonuçlar karşılaştırılmıştır. UDP üzerinde, belirtilen oranda paket trafiği oluşturan CBR (Constant Bit Rate) uygulaması çalıştırılmıştır.



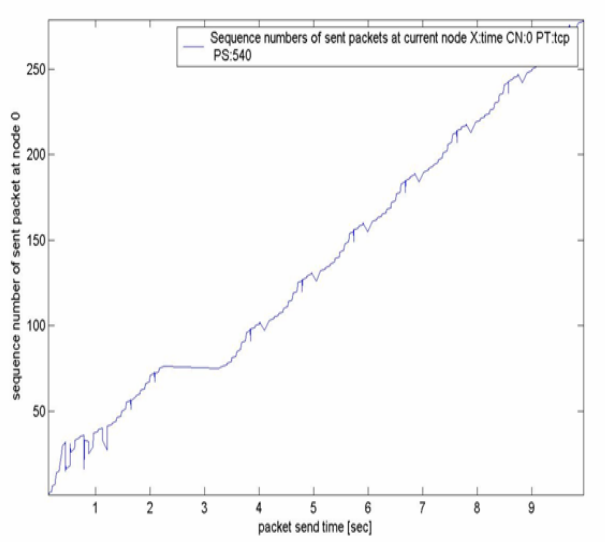
Şekil-3. Simulasyon Topolojisi 1

Şekil-3'te görüldüğü gibi R1 ve K1 arasında bir darboğaz oluşturulup TCP ve UDP paketlerinin rastgele olarak düşürülmesi sağlanmıştır. R1 yönlendiricisi üzerindeki kuyruk boyutu, 10 paket olarak belirlenmiş olup bu boyutun aşılması durumunda paketler kuyruktan atılacaklardır.

Yukarıdaki senaryoya göre belli bir zaman aralığında gönderilen toplam paket sayısı karşılaştırılmıştır. Şekil-4 ve Şekil-5 karşılaştırıldığında, aynı süre içinde, geliştirilen yaklaşımda %20-%30 arasında daha fazla paket gönderildiği görülmektedir. Şekil-4 ve Şekil-5'te X ekseni zamanı, Y ekseni ise gönderilen paketlerin sıra numarasını göstermektedir



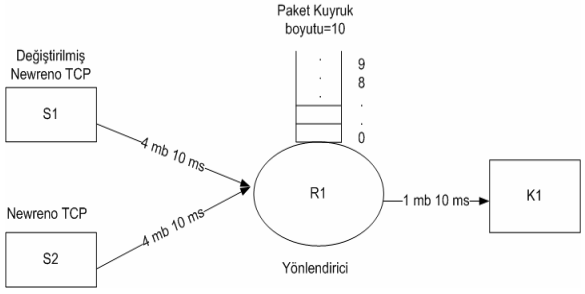
Şekil-4. NewReno TCP'de Gönderilen Paketlerin Sıra Numarası



Şekil-5. Değiştirilmiş NewReno TCP'de Gönderilen Paketlerin Sıra Numarası

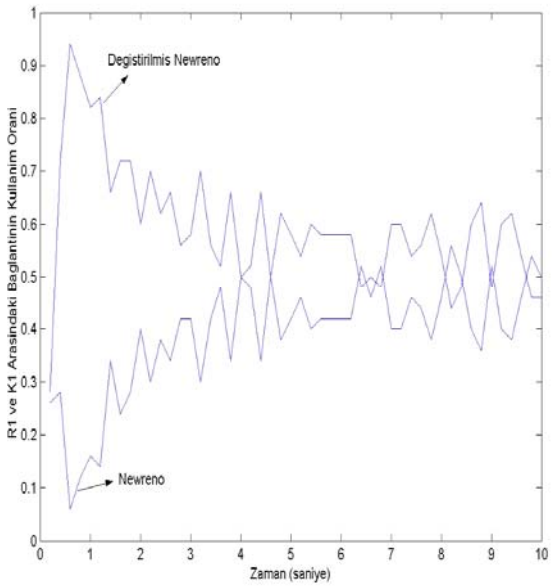
4.2 Simülasyon Senaryosu 2

Şekil-6'da Newreno TCP ile değiştirilmiş Newreno TCP, aynı senaryoda çalıştırılarak performansları karşılaştırılmıştır. Her iki TCP bağlantısı üzerinde de FTP uygulaması çalıştırılmıştır. R1 ve K1 arasında darboğaz oluşturulup paketlerin rastgele düşmesi sağlanmıştır. Senaryo 1'de olduğu gibi R1 üzerindeki kuyruk sayısı 10 olarak ayarlanmıştır.



Şekil-6 Simulasyon Topolojisi 2

Şekil-7'de R1 ile K1 arasındaki bağlantının zamana göre verimli kullanım oranı karşılaştırılmıştır. X ekseni zamanı Y ekseni de bağlantının kullanımını göstermektedir. Buna göre çalışmamızın R1 ile K1 arasındaki bağlantıyı daha verimli kullandığı açıkca görülmektedir.



Şekil-7 Verimlilik Değerlerinin Karşılaştırılması

5. SONUÇ

Bu makalede, TCP'nin güvenilir veri iletim protokolü üzerinde değişiklik yapılarak hızlandırılması sağlanmıştır. Geliştirilmiş olan Newreno TCP çeşitli simülasyon topolojilerinde kullanılarak sonuçlar karşılaştırılmıştır. Simülasyon sonucunda elde edilen grafiklerden görüldüğü gibi yapılan değişikliklerin %30 arasında performansı arttırdığı görülmektedir.

KAYNAKLAR

[1] J. Postel, Transmission Control Protocol, RFC 793, September 1981

[2] K. Fall, S. Floyd, Simulation - Based Comparisons of Tahoe, Reno, and SACK TCP, ACM computer Communication Review, 26(3), pp.5-21, 1996.

[3] Flashget Web Site, http://www.amazesoft.com/

[4] S. Ladha, P. D. Amer, Improving Multiple File Transfer Using SCTP Multistreaming.

[5] J. F. Kurose, K. W. Ross, Computer Networking, Second Edition, 2003.

[6] W. R. Stevens, TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms, RFC 2001, January 1997.

[7] W. R. Stevens, TCP/IP Illustrated, Volume 1: The Protocols, Reading, Massachusetts: Addison-Wesley, 1994.

[8] M. Allman, V. Paxson, W. Stevens, TCP Congestion Control, RFC 2581, April 1999.

[9] K. Fall, K. Varadhan, The ns Manual (Formerly ns Notes and Documentation), December 13, 2003.