AĞ PROTOKOLLERİNDE GÖRÜLEN BAZI PROBLEMLER VE ÇÖZÜM ÖNERİLERİ

Turgay KALAYCI

***Özet*-Bilgisayarla iletişim sistemlerinde ortaya çıkan problemler ve bu problemlerin bazılarını çözen metodlar vardır. Örneğin, iletim esnasında bitlerin nasıl bozulduğunu biliriz. Bu tip hataları bulmak için, veri iletimi protokolleri ; parity bit,frame checksum ve CRC gibi birkaç çeşit teknik kullanırlar. Seçilen kesin teknik tüm protokol takımının planına bağımlıdır. Bazı protokoller hata bulmaktan daha fazlasını yaparlar. Bunlar problemleri önlemek veya onarmak için çaba sarfederler. Özellikle taşıma protokolleri en karmaşık olanların bazılarını çözmek için çeşitli araçlar kullanırlar. Bu makale ortaya çıkan problemlerin bir kümesini inceler ve bunları çözmek için kullanılan protokol tekniklerini tartışır.**

***Abstract*- There were some examples of problems that arise in communication systems and the ways protocols solve some of those problems. For example, we know how bits corrupted during transmission . To detect such errors , data link protocols use a variety of techniques including: a parity bit , frame checksum, or a cyclic redundancy check (CRC). The exact technique chosen depends on the design of the entire protocol suite.**

Some protocols do more then detect errors. They make an effort to repair or circumvent problems. In particular , transport protocols use a variety of tools to handle some of the most complicated communication problems. This paper reviews a set of problems that can arise and discusses techniques protocols use to solve them.

I. GİRİŞ

Asıl iletişim donanımı bitleri bir uçtan diğerine ileten düzeneklerden oluşur. Ancak bu haliyle donanımı iletişim için kullanmak , 1 ve 0'larla program yapmaya benzer. Buda karmaşık ve zaman alıcı bir iştir. Programcıların desteği ile ağa bağlı bilgisayarlar uygulamalar için yüksek seviyeli , elverişli bir arayüz yazılımı kullanırlar. Bu yazılım birçok düşük seviyeli iletişim ayrıntılarını ve problemlerini otomatik olarak bertaraf ederek , kolayca iletişimi mümkün kılmaya olanak sağlar. Bu sebepten uygulama programlarından önemli bir kısmı iletişimde ağ yazılımına itimat ederler ve böylece direk olarak ağ donanımı ile ilgi değildirler.

Bir iletişim içinde bulunan bütün gruplar mesajla dönüştürürken kullanılan kural kümesini kabu etmelidir. (yani , kullanılan dil ve mesa gönderimindeki kurallar). Diplomatlar böyle bi sözleşmeye "protokol" derler. Bu terim bilgisaya iletişiminde de kullanılır: mesaj formatını ve her mesa için istenen uygun eylemi belirleyen kurallar kümesin "ağ protokolü" veya "bilgisayar iletişim protokolü denilir. Bu kuralları çalıştıran yazılıma da "protoko yazılımı" denir. [1]

T.KalaycıİstanbulÜniversitesi Bilgisayar Bilimleri Uyg.ve Araş.Mer34452 , Beyazıt/İSTANBUL

II. PROTOKOLLERİN KULLANDIĞI TEKNİKLER

II.1.Sırası Bozuk Paketleri Sıralama

Bağlantı yollarını değiştirebilen bağlantısız bir a sistemi , paketleri bozuk sıralı olarak dağıtabilir. Niçi böyle olduğunu anlamak için , bir dizi paketler gönderildiğini varsayalım ve unutmayın ki , a herhangi bir zamanda boş olan en kısa yolu seçer. Eğe i.ci paket gönderildikten hemen sonra en kısa yo devreye girmişse , i+1.ci paket bu kısa yol gönderildiği için i.ci paket ulaşmadan i+1.ci pake ulaşır.

Sırası bozuk olan paketleri taşıma protokolü "sıralama işlemine tabi tutar. Gönderen taraf her pakete bir sır no ekler. Alan taraf sıralı olarak ulaşan son paketin sır nosunu ve sırası bozuk ulaşan paketlerin bir listesin saklar. Bir paket geldiğinde , alan taraf bu pakete hang işlemin yapılacağını belirlemek için sıra nosun denetler. Eğer paket sırasında gelmişse protoko yazılımı bu paketi bir üst katmana gönderir , ve sıras bozuk olanların tutulduğu listeye de bakarak buradan gönderilmesi gerekenlerin olup olmadığını kontro eder. Eğer gelen paketin sırası bozuksa , protoko yazılımı bu paketi sırası bozukların listesine ekler. [2]

II.2.Çift Gönderilen Paketleri Elemek İçin Sıralama

Kötü fonksiyonlu donanım , paketlerin çift gönderilmesine neden olabilir. Çift gönderm WAN'larda sık görülsede LAN'larda da görülebilir

Örneğin, LAN'daki CSMA/CD kullanan bir bağlantının aksaması sonucu gönderenin bir çatışma olduğunu fark etmesine karşın alıcı bir geçerli ileti aldığını sanır. Sonuçta ; gönderici çatışma sonrası geri gelerek tekrar bir kopya daha gönderir ve böylece alıcı tarafta iki kopya olacaktır.

Sıralama bu çift gönderme problemini çözer. Alıcı tarafın yazılımı ulaşan paketin sıra nosuna bakarak çift olup olmadığını denetler. Eğer sıra no daha önceden dağıtılmışsa veya sıra no listede bekleyen paketlerden biri ile çakışıyorsa , yazılım gelen yeni kopyayı atar.

II.3 Kaybolan Paketleri Yeniden Gönderme

Paket kaybı bilgisayar ağlarının önemli bir problemidir. Çünkü ; iletişim hataları bitleri bozar ve çerçeveyi geçersiz yapar. Alıcı böylesi problemleri fark ettiğinde bu çerçeveyi atar. Güvenli bir transferi garanti etmek için (yani paket kaybı olmaksızın) protokoller "geri gönderilen kesin alındı bilgisi" kullanırlar. Bir çerçeve tam olarak ulaştığında , alıcı protokol yazılımı , bu çerçeveyi başarılı olarak aldığını beyan eden küçük bir mesajı geri gönderir. Bu mesaj "doğrulama bilgisi" yani "ACK" (Acknowledgement) dir. Gönderici gönderdiği her paketin doğru olarak ulaştığının garantisini vermekle yükümlüdür. Gönderici bir paket gönderdiğinde , protokol yazılımı bir zaman başlatır. Eğer doğrulama bilgisi zaman bitmeden ulaşırsa, yazılım zaman sayacını durdurur. Eğer doğrulama bilgisi ulaşmadan önce zaman dolarsa , yazılım paketin başka bir kopyasını gönderir ve zamanı tekrar başlatır. İkinci kopyayı gönderme olayına "tekrar gönderilme" denir ve bu kopyaya "tekrar gönderim" adı verilir. [3,4]

Eğer donanım bozukluğu ağı kalıcı olarak kesmişse veya alıcı bilgisayar çökmüşse tekrar gönderim başarılı olamaz. Bu sebepten , protokoller tekrar gönderim sayılarına bir maksimum sınır koyarlar. Bu sınır aşıldığında protokol tekrar gönderim işlemini durdurur ve iletişimin mümkün olmadığını deklare eder.

Dikkat etmek gerekir ki, tekrar gönderim çift paketlere neden olabilir. Çünkü, gönderici kaybolan paketi ve uzun zaman geciken paketi birbirinden ayıramadığından , sonunda bu paketi tekrar göndermeye karar verebilir. Sonunda paketin iki kopyası dağıtılabilir. Bunun için tekrar gönderimleri kullanan protokoller güvenliği sağlamak için çift paket problemlerini de halletmelidirler.

II.4 Aşırı Gecikmelerin Neden Olduğu Tekrarlamadan Sakınma

Paket anahtarlama sisteminde gecikmenin bir kaynağı da sakla ve gönder (store and forward) tekniğinden kaynaklanmaktadır. Paket anahtarına bir paket geldiğinde , bu paket kuyruğa konulur. Eğer paketler onu gönderen anahtarın hızından daha hızlı gelmişlerse , paketlerin beklediği kuyruk büyüyecek ve gecikme aşırı olabilecektir. Aşırı gecikmeler "tekrarlama hatalarına" sebep olabilirler. "Tekrarlama"nın anlamı : bir eski gecikmiş paket daha sonraki iletişimi etkiler. Örneğin , aşağıdaki olaylar dizisini düşünelim:

- İki bilgisayar saat 13'de iletişim için anlaşsınlar.
- Bilgisayardan biri diğerine 10 paketlik bir dizi göndersin.
- Bir donanım problemi 3. paketin gecikmesine neden olsun.
- Bu donanım probleminden kurtulmak için yollar değişsin.
- Gönderen bilgisayarın protokol yazılımı 3. paketi tekrar gönderir ve kalan paketler hatasız olarak gönderilsin
- Saat 13:05'de iki bilgisayar tekrar iletişim için anlaşsın
- İkinci paket geldikten sonra , daha önceki bağlantıdan 3. paketin geciken kopyası gelsin.
- İkinci bağlantıdan 3. paket gelsin.

Maalesef , protokol dikkatli planlanmadıkça , önceki bağlantıdan gelen bir paket sonraki bağlantılar tarafından alınabilir ve gelen doğru paket sanki çift gelmiş gibi atılır.

Tekrarlama kontrol paketlerinde de oluşabilir. Örneğin, protokoller bir bağlantıyı bitirmek için çoğunlukla özel bir kontrol paketi gönderirler. Eğer bitirme talebinin bir kopyası önceki bağlantıdan gelmişse , bu vakitsizce bir oturumun protokol yazılımı tarafından bitirilmesine neden olabilir.

Tekrarlamayı önlemek için , protokoller her oturumu özel bir ID ile işaretler ve her pakette bu özel ID'yi ister. Protokol yazılımı geçersiz ID'yle gelen paketleri atar. Tekrardan sakınmak için bir ID makul bir süre (saatler) geçmedikçe tekrar kullanılmamalıdır.

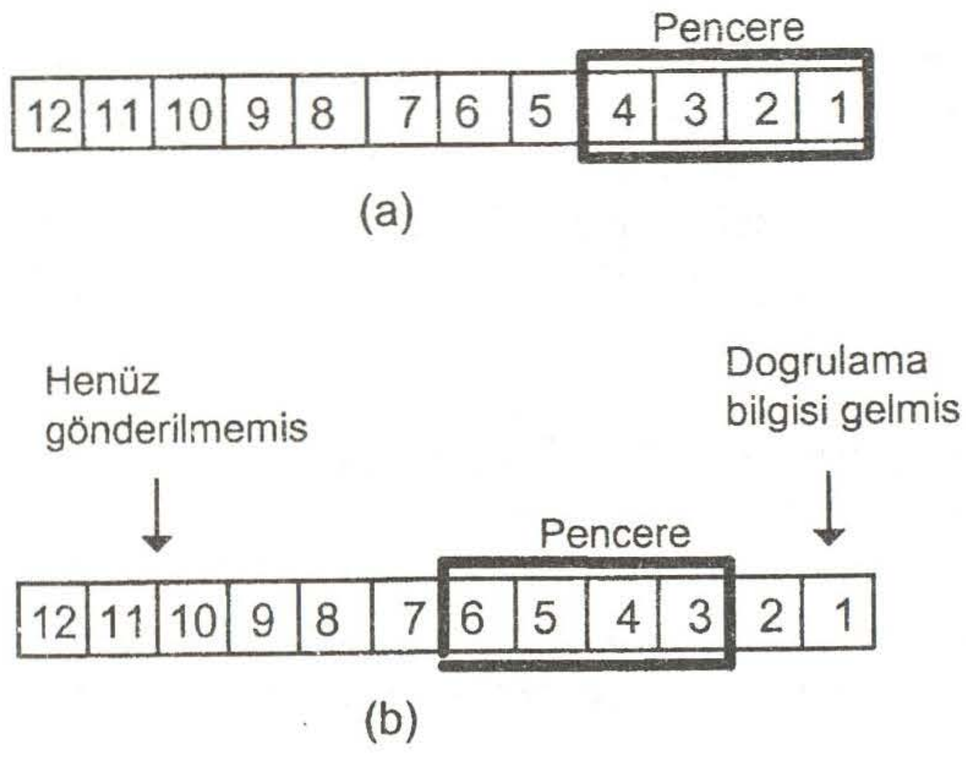
II.5 Veri Aşırılığını Önlemek İçin Akış Denetimi

Bilgisayarların hepsi aynı hızda çalışmazlar. Bir bilgisayar ağ üzerinden karşı bilgisayarın alabileceği veriden daha hızlı veri gönderirse , o zaman veri aşırılığı görülür. Bunun sonucu olarak da veri kaybolur.

Veri aşırılığını çözmek için bir çok teknik vardır. Toplu olarak , bu tekniklere "akış denetimi" yöntemleri denilir. Akış denetiminin en basit şekli "bekle ve git" sistemidir. Bu sistemde gönderici her paketi gönderdikten sonra bekler. Alıcı başka bir paketi almak için hazır olduğunda doğrulamanın bir biçimi olarak

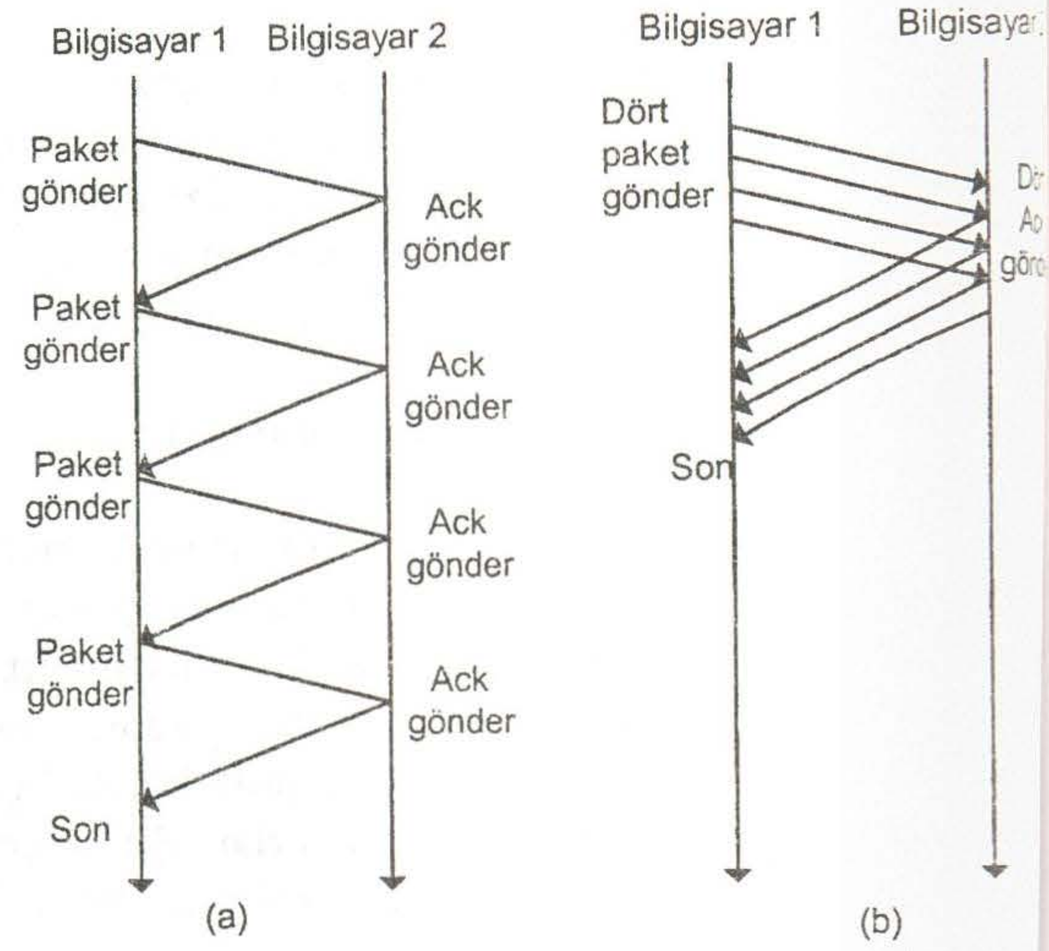
kontrol mesajı gönderir. Bekle ve git protokolleri veri aşırılığını önlemesine rağmen , ağ bant genişliğinin son derece yetersiz kullanımına neden olabilir. Niçin böyle olduğunu anlamak için , bir paketin 1000 bayt , ağın hız kapasitesi 2 Mbps ve gecikmenin 50 ms olduğu bir ağ düşünelim. Ağ donanımı bir bilgisayardan diğerine 2 Mbps gönderebilir (biz buna ağın hızı diyoruz). Ancak , bir paket gönderdikten sonra gönderici diğer paketi göndermeden önce 100 msec beklemelidir (yani 50 ms paketin alıcıya ulaşması için ve 50 ms doğrulama mesajının geri gitmesi için). Bu nedenle , bekle ve git kullanılarak gönderilen verinin maksimum hızı her 100 ms'da bir pakettir. Bit hızı olarak ifade edilirse, bekle ve dur'un başardığı maksimum hız 80.000 bps. Bu da donanım kapasitesinin %4'dür.

Yüksek ağ hızlarını (throughput) elde etmek için , protokoller "kayan pencere" adı verilen bir akış kontrol tekniği kullanırlar. Gönderen ve alan bir doğrulama (ack) ulaşmadan önce gönderilebilen maksimum veri miktarından ibaret olan sabit bir pencere ölçüsü , kullanmak için programlanılır. Örneğin, gönderen ve alan 4 paketlik bir pencere ölçüsünde anlaşsınlar. Gönderen ilk önce birinci pencereyi doldurmak için verileri alır ve kopyaları gönderir. Güvenlik açısından gönderen tekrar gönderim gerekir diye elinde gönderdiklerinin yedeğini tutar. Alan taraf gelen tüm pencereyi alabilen buffer alanını hazır tutmalıdır. Bir paket sırasında geldiğinde alıcı bu paketi kabul etme işlemlerine gönderir ve gönderene bir doğrulama mesajı atar.



Şekil 1. Dört paketlik bir pencere giden verinin üzerinde kayar. (a)'da iletişimin başlangıcında pencerenin durumu gösterilmektedir, (b) 2 paket doğrulandıktan sonraki durum ve (c) 8 paket doğrulanmıştır. Gönderen , pencere içindeki bütün paketleri iletebilir.

Kayan pencere tekniği ağ hızını oldukça artırabili Niçin olduğunu anlamak için , bekle ve git tekniği v kayan pencere tekniğinde iletişim sırasını inceleyelim Şekil 2, 4 paket iletimi için bir karşılaştırm içermektedir.



Şekil 2. (a) bekle ve git akış kontrolü kullanılarak 4 paketin iletiş sırası (b) 4-paket kayan pencere tekniği. Zaman aşağı doğru ok gösterilmiştir. Ve her bir ok bir bilgisayardan diğerine giden mes göstermektedir.

Doğrulama göndericiye ulaştığında , doğrulanm paketin kopyası silinir ve bir sonraki paketi gönder Şekil 1. bu işleme neden kayan pencere dendiğ açıklamaktadır. [4]

Şekil 2.a bekle ve git protokolü için iletimlerin sırası göstermektedir. Bir paket gönderildikten sonra protokol diğer paket gönderilmeden önce b doğrulama (ack) bekler. Eğer ağ içinde bir tek paket bir yerden bir yere gitmesi için gereken gecikme N is toplam zaman 8N'dir.

Şekil 2.b kayan pencere kullanıldığında iletiş sırasını göstermektedir. Bu protokol pencere içindeki tüm paketleri beklemeksizin gönderir. Gerçeğe uysu diye , şekilde peş peşe paket transferleri arasında küçü bir gecikme gösterilmiştir. Bu gecikm gösterildiğinden daha da küçük olabilmesine rağmen iletişim asla ani değildir. Kısa bir zamana (çoğunluk birkaç mikro saniye) donanım tarafından pake iletimini tamamlamak için gerek vardır. Bu sebepten paketi göndermek için gereken toplam zama 2N+ε'dur. Burada ε küçük bir gecikmeyi ifade eder.

Kayan pencerenin önemini anlamak için , bir ço paketi ilgilendiren kapsamı genişletilmiş bir iletişi düşünelim. Böyle durumlarda , iletim için gerek toplam zaman o kadar büyüktür ki ε dikka alınmayabilir. Kayan pencerenin faydalarını anlam

için, ağ hızı yüksek olan bir ağ ve fazla gecikme düşünelim (yani uydu kanalı). Böyle ağlar için kayan pencere protokolü 1'den daha büyük bir faktör ile performansı artırabilir. Gerçekten potansiyel iyileştirme :

$$T_w = T_g * W$$

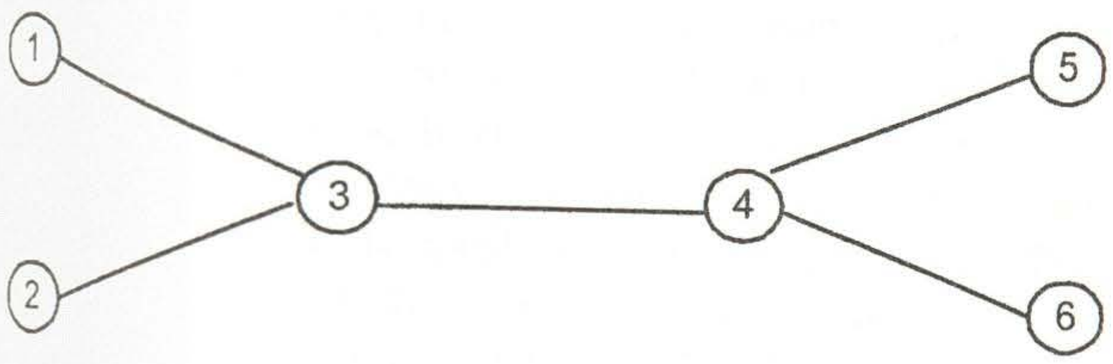
Burada T_w kayan pencere protokolü ile elde edilen ağ hızıdır , T_g ise bekle ve git protokolü ile elde edilen ağ hızıdır ve W pencere ölçüsüdür. Bu eşitlik , şekil 2b'de açıklanan kayan pencere protokolünün neden şekil 2a'da açıklanan bekle ve git protokolünün 4 katı olduğunu açıklamaktadır. Elbette, ağ hızı sadece pencere ölçüsü artırılarak keyfen büyütülemez. Kullanılan ağın bant genişliği bir sınır belirler. Donanımın taşıyabileceğinden daha hızlı bitleri gönderemezsiniz. Bu sebepten , yukarıdaki eşitliktekrar şu şekilde yazılır :

$$T_w = \min (B , T_g * W)$$

Burada B o andaki kullandığınız ağ parçasının bant genişliğidir. [5]

II.6 Ağ Tıkanıklığını Önlemek İçin Yöntemler

Paket anahtarlı sistemlerde asıl problem tıkanmadır. Nedenini anlamak için, şekil 3'deki grafikle gösterilen bir ağ düşünelim.



Şekil 3. Grafik 8 paket anahtarlı bir ağı gösterir. Böyle ağlar tıkanıklılık yaşayabilir.

Varsayalım ki, kullanılan ağ içindeki her bağlantının bant genişliği 1.5 Mbps olsun ve bir taraftan diğerine trafik akışının orta link'ten nasıl geçeceğini düşünelim. Örneğin, düğüm 1'e bağlanan bilgisayar düğüm 5'e bağlanan bilgisayara bir dizi paket gönderdiğini farzedelim. Paketler 1 ve 3 düğümleri arasında 1.5 Mbps ile giderler ve 3 ile 4 düğümleri arasında 1.5 Mbps ile giderler. [6]

Eğer yalnız bir bilgisayar paketleri gönderiyorsa , ağ iyi çalışır. Ancak, orta link'te herhangi bir ek trafik tıkanmaya neden olacaktır. Örneğin , düğüm 2'ye bağlı olan bir bilgisayar düğüm 6'daki bir yere paketleri göndermeye başlarsa , bu paketlerde orta link üzerinden geçeceklerdir. Düğüm 1 ve düğüm 2 'nin her ikisi birden paket göndermesiyle düğüm 3'e iki defa veri ulaşır , düğüm 3 düğüm 4'e elinden geldiği kadar hızlı bir şekilde verileri gönderir. Düğüm 3'deki paket anahtarı düğüm 1 ve düğüm 2'den gelen paketleri , gönderilene kadar bir kuyruğa atar. Gönderebildiğinden daha fazla paket geldiğinde ise , kuyruk büyür ve etkin gecikme artar. Bu duruma "tıkanma" denir.

Eğer tıkanma süreklilik gösteriyorsa , paket anahtarı belleği tüketecek ve paketleri atmaya başlayacaktır. Kayıp paketleri kurtarmak için tekrar gönderme olmasına rağmen , bu işlem zaman alır. Üstelik bu durum devam ederse , bütün ağ kullanılamaz olabilir. Bu duruma "tıkanma çökmesi" denir. Protokoller ağı izleyerek tıkanma çökmesinden korunmak için teşebbüste bulunurlar ve tıkanma başlar başlamaz hızlı olarak harekete geçerler. İki yaklaşım vardır :

- Tıkanma göründüğünde gönderenleri bilgilendirecek şekilde paket anahtarları düzenlenir.
- Tıkanıklılık olmasına karşın kayıp paket kullanılması

İlk yaklaşım iki halden birinde uygulanır: tıkanma görüldüğünde paketlerin kaynağına paket anahtarları bir özel mesaj gönderir veya tıkanıklılıktan dolayı bir gecikme yaşanırsa paket başlıklarının içindeki bir bit değiştirilir. Eğer bu bit değiştirilirse paketleri alan bilgisayar doğrulamanın içinde orijinal göndericiyi uyaran bilgiler koyar. [7]

Tıkanıklılık olmasına karşın kayıp paket kullanma modern ağlarda olağan bir şeydir. Çünkü, modern ağ donanımı iyi çalışır , paket kaybının çoğu tıkanmadan kaynaklanır, donanım bozulmasından değil. Bu sebepten gönderen ağın tıkanmış olduğunu ve hepsinin kayıp gördüğü varsayımını yaparak zamanın çoğu için geçerli bir varsayım yapmış olur. Eğer gönderen mola ve tekrar gönderme stratejisi kullanırsa paket kaybı kolayca ölçülebilir. Her tekrar gönderme olayı tıkanmanın olduğunu gösterir.

Tıkanmaya karşı bir tavır olarak, paketler gönderiliyorken hızını azaltmak düşünülebilir. Bazı protokoller , hangi sıklıkla paketlerin üretildiğini izleyen ve sonra tıkanma görüldüğünde geçici bir süre paket hızlarını azaltan bir hız denetimi yöntemi kullanırlar. Kayan pencere protokolleri geçici bir süre pencere ölçüsünü azaltarak aynı etkiyi başarabilirler.

III. SONUÇ

Belirli problemleri çözmek için gereken teknikler çok iyi bilinmesine karşın, protokol planlaması iki sebepten dolayı önemlidir. Birincisi , iletişimi etkin kılmak için detaylar dikkatlice seçilmelidir. Küçük planlama

hataları yanlış işlemle , gereksiz paketler veya gecikmelerle sonuçlanabilir. Örneğin, sıra noları çoğu kez paket başlığının içinde sabit bir alanda saklanır. Bu alan sıra noları sık sık tekrar kullanılmaması için yeterince büyük olmalıdır. Fakat yeterince küçük olması da bant genişliğini gereksiz yere harcamaktan imtina edilmiş olur. Eğer bu protokol en iyi olmayan mesaj ölçüsü seçerse benzer şekilde yüksek seviyeli bir protokol sürekli ekyüke neden olabilir (örneğin, paketin maksimum uzunluğundan bir bayt daha fazla). İkincisi , protokol yöntemleri umulmayan biçimde uyuşum içinde çalışabilirler. Örneğin akış kontrolü ve tıkanma kontrolü yöntemleri arasındaki etkileşimi düşünün. Kayan pencere tekniği mütecavizce içinde bulunduğu ağın bant genişliğinin çoğunu kullanarak ağ hızını arttırır. Tıkanma kontrolü yöntemi yukardaki cümlenin tersine ağın çökmesini önlemek için paketlerin geçişini azaltır.

Kayan pencere ve tıkanma kontrolü arasındaki dengeyi kontrol etmek zor olabilir ve her ikisini iyi şekilde planlamak güçtür. Bant genişliği kullanma konusunda oldukça mütecaviz olan bir protokol çalıştığı ağı tıkayabilir, tutucu olan (yani paketlerin yavaş yavaş geçmesini isteyen) bir protokol gereğinden daha az ağ hızını oluşturabilir. Daha da önemlisi , planlamalar gerektiğinde bant genişliği harcama konusunda mütecaviz gerektiğinde de tutucu olmalıdır.

KAYNAKLAR

[1] Comer , E.Douglas, "Computer Networks And Internets" , Prentice –Hall Inc , 1999

[2] Wellsh, Matt ., Basu , Anindya., Eicken, Thorsten von ,Department of Computer Sciences,Ithaca, NY 14853 http://www.cs.cornell.edu/Info/projects/U-Net

[3] A.Edward, G.Watson, J.Lumley, D. Banks, C. Calamvokis and C.Dalton. "User-space protocols deliver high performance to applications on a low-cost Gb/s LAN". Proc. Of SIG-COMM-94,p 14-23, Aug.1994

[4] M.Blumrich, C. Dubnicki, E.W. Felten and K.Li. "Virtual-Memory-Mapped NetworkInterfaces". IEEE Micro, Feb. 1995, p21-28

[5] S.Komandur and D.Mosse, "SPAM: A data forwarding model for multipoint to multipoint connection support in ATM networks". Proceeding of the sixth international conference on computer communications and networks (ICCCN '97) , pp. 383-388, Septemper 1997.

[6] E. M. Spiegel, "Multipoint connection support in PNNI 2.0". ATM Forum Perspectives, IEEE Network, Nov/Dec 1997.

[7] Sastri Kota, "Multimedia Satellite Networks : Issues and Challenges", Proc. SPIE International Symposium on Voice , Video and Data Communications, Boston , Nov 1-5, 1998.