BOLUM 4

BAGLANTI YONETIMI

HTTP ozellikleri HTTP mesajlarini oldukca iyi acikliyor,ama HTTP baglantilari hakkinda cok konusulmadi,kritik tesisatlarda HTTP mesaj akisi saglanir.Eger HTTP uygulamasi yazan programcisiysaniz,HTTP baglantilarini iciyle disiyla nasil kullanicaginizi bilmelisiniz.

HTTP baglanti yonetimi birazcik kara buyu gibidir,literaturdekiler gibi cirakliktan yavasca deneyimleyerek ogrenebilirsiniz.Bende hicbir sey ogrenmedim sadece anlaticiyim(The narrator).Bu bolumde ogreniceklerimiz:

1-HTTP,TCP baglantilarini nasil kullaniyor.

2-TCP baglantilarindaki gecikme,engel,tikanikligi

3-HTTP optimizasyonlari(parallel,yasam sureleri,borulama baglantilari)

4-Baglanti yonetimi icin yapilacaklar ve yapilmayacaklar

TCP BAGLANTISI

Su an tam anlamiyla dunyadaki tum HTTP iletisimleri TCP/IP araciligiyla tasiniyor,populer katmanli paket anahtarlamali bir network protokoludur,dunya uzerinde bilgisayarlarla ve network cihazlariyla konusur.Bir Istemci uygulamasi server uygulamasiyla TCP/Ip baglantisi acabilir,ve dunyanin herhangi bir yerinden calisabilir.Bir kere baglanti saglandi mi.Istemci ve server arasinda degisen mesajlar asla kaybolmaz,hasar gormez,yada alici disina cikmaz.

Joe's Hardware dukkaninin en guclu araclari bolumune ugrayalim:

http://www.joes-hardware.com:80/power-tools.html

Verilen URL,Tarayiciniz asagidaki adimlari uygulayacak,adim 1-3 de,serverin Ip adresi ve portu URL'den cekiliyor.TCP baglantisi adim 4'te yapiliyor.Adim 5 te istek mesaji yollaniyor.Cevap mesaji adim 6'da okunuyor baglanti adim 7'de kapaniyor.

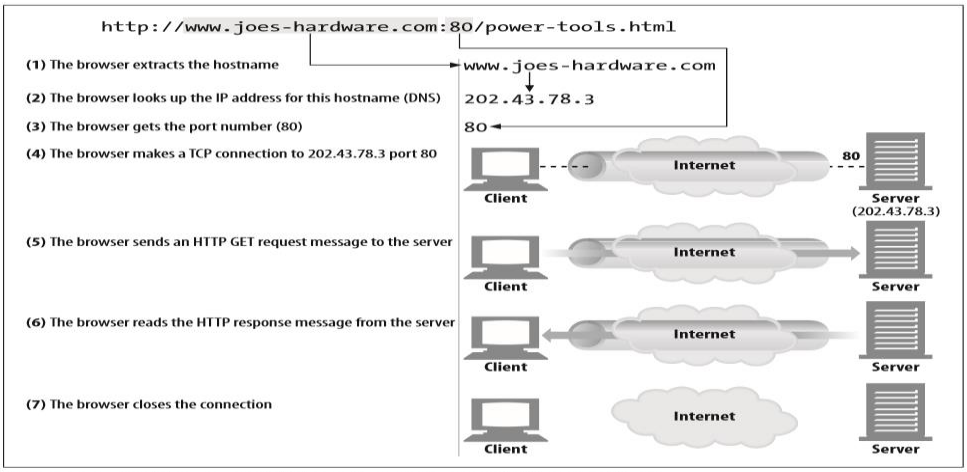

Figure 4-1. Web browsers talk to web servers over TCP connections

TCP GUVENILIR DATA PIPELARI

HTTP baglantilari TCP baglantilarindan ibarettir,artidan az cok kuralla onu nasil kullanacagimizi anlariz.TCP baglantilari guvenilir baglantilardi.Datayi isabetli ve seri bir sekilde gondermek icin,TCP basiclerini bilmeniz gerekli.

TCP HTTP'ye reliable bit pipe(guvenilir bit borulari ama ingilizce terimlere alismaniz gerekicek o yuzden buralari cevirmiyorum her yerde bilmeniz gereken terimler) verir.Bytelar bir yerde dolduruluyor TCP baglantisiyla diger taraftan dogru bir sekidle cikiyor.Resim geldi.

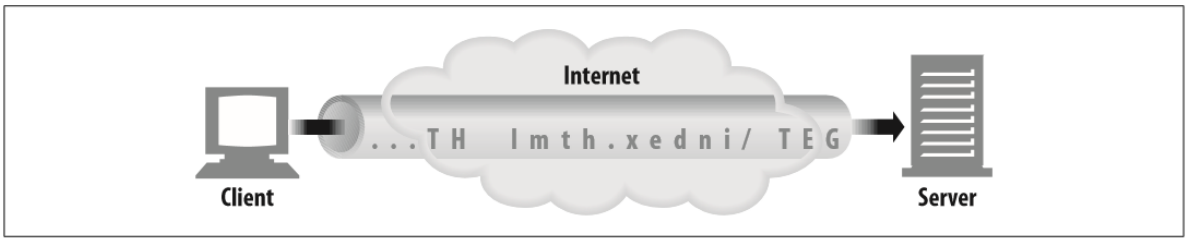

Figure 4-2. TCP carries HTTP data in order, and without corruption

TCP Paketleri Dilimlenir ve Ip paketleri tarafindan sevkiyati olusturulur.(Baya uzun oldu ama anlicaniz:D)

Tcp datalari IP paketi denen(veya IP datagramlari)kucuk yiginlarla yollar.Bu yola bakarsak,HTTP protocol yigitlarinda en ust katmandir.Resimde anlicaniz.Guvenli varyasyonlari var,HTTPS,kriptografik enkript katmani eklenmistir(TLS veya SSL) TCP ile HTTP arasina.

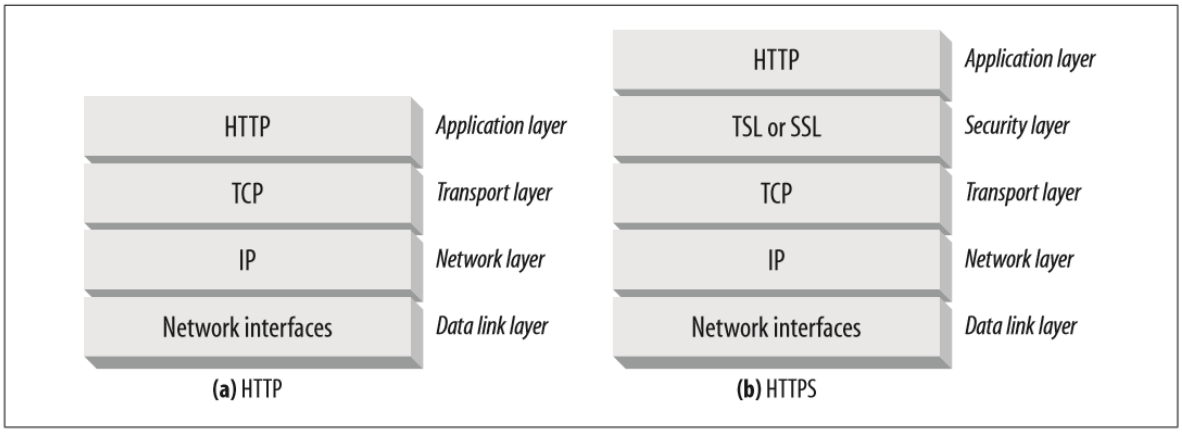

Figure 4-3. HTTP and HTTPS network protocol stacks

HTTP mesaji tasimak istedigi zaman,mesaj datasinin iceriklerini paketler,sirayla,TCP baglantisi acilir.TCP paketlenmis datayi alir,paketlenmis datayi parcalayarak segment dedigimiz seye donusturur,ve ulasim segments dedigimiz zarflar araciligiyla Internette dolasir biz bunlara IP paketleri diyoruz iste.TCP/IP ya hani baglantili iste.TCP/IP bakiyor buralara tabiki,HTTP armut pis azima dus hic bisi yapmiyor.Resimde gene anlicaksniz.

Her TCP segmenti IP packetlerini IP adresinden baska bir Ip adresine tasir.IP paketleri sunlari kapsar:

IP paket basligi(20 bytetir) Kattlettim ingilizceyi katlettik.

TCP segment basligi(20 bytetir)

TCP data yigini(0 da olabilir daha fazlada)

IP headeri kaynagin ve hedefin IP adresini,boyutunu,ve flaglari kapsar.TCP segment headeri TCP port numarasini,TCP kontrol flaglarini(bayraklari) ve data siparislerinin butunlugu kontrolu icin numarali deger tasir.

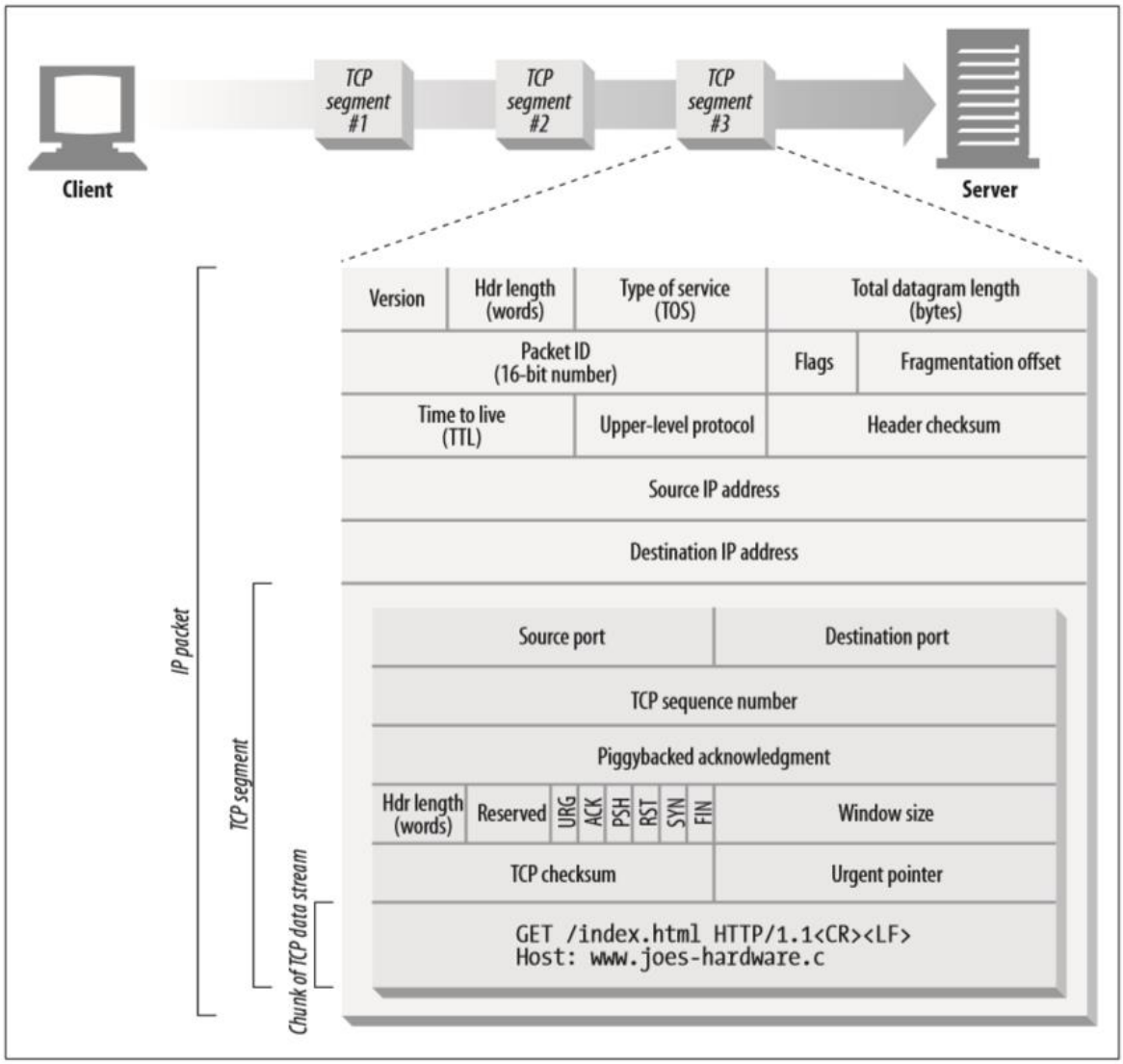

Figure 4-4. IP packets carry TCP segments, which carry chunks of the TCP data stream

TCP BAGLANTISINI DUZGUN SAGLAMAK

Bir bilgisisayar ayni anda birden fazla TCP baglantisi saglayabilir.TCP tum bu baglantilari port'a dogru gonderir.

Portlar calisanlarin sirketlerde uzatma kablolu telefonu varya onlara benzetebiliriz.Sirketteki ana telefon ile numarayi cevirdiginiz zaman istediginiz calisanin telefonuna ulasabilirsiniz.Ip adresi size dogru bilgisayara,port ise dogru uygulamaya yonlendirir.TCP baglantilari dort degerle taninir.

‹kaynak-IP-adress, kaynak-port, hefed-IP-adress,hedef-port›

Hepsi beraber,essiz bir baglanti saglar.Iki farkli TCP baglantisi 4 bileseninde ayni address olmasina izin vermez(ama bazi baglantilarda ayni degerler bazi bilesenlerde olabilir).

Resim geliyor.4 baglanti var.

Table 4-1. TCP connection values

Connection	Source IP address	Source port	Destination IP address	Destination port
A	209.1.32.34	2034	204.62.128.58	4133
B	209.1.32.35	3227	204.62.128.58	4140
C	209.1.32.35	3105	207.25.71.25	80
D	209.1.33.89	5100	207.25.71.25	80

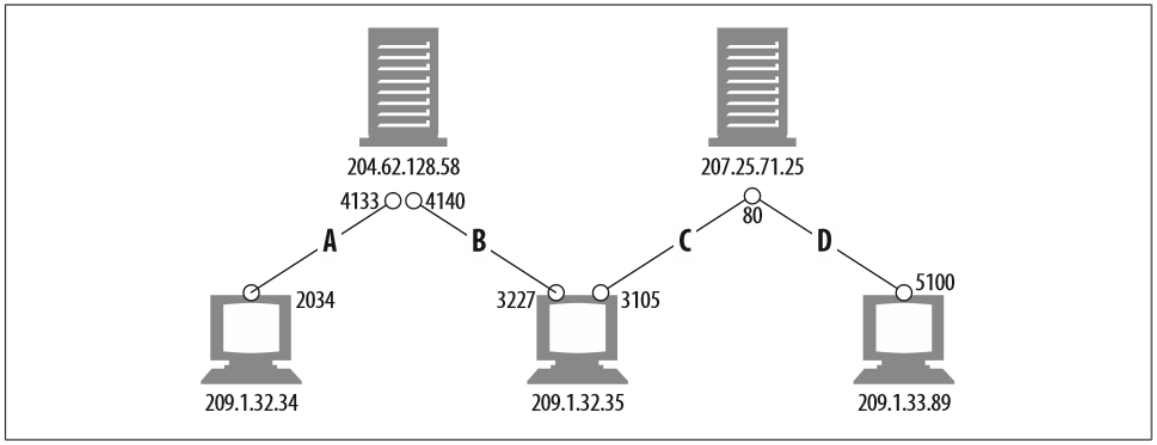

Figure 4-5. Four distinct TCP connections

Sunu bilin ki C ve D ayni hedef portu paylasiyorlar.Bazi baglantilar ayni IP adresine sahip(B ve C).Bazilari ayni hedef IP adresine sahip(A ve B,C ve D).Ama ayni anda 4 degeri paylasamiyorlar.

TCP SOKETLERI PROGRAMLAMAK

Isletim sistemleri TCP baglantilarini yonetmek icin bir suru kolayliklar saglar.Hizlica TCP programlama arayuzlerine bazi seyleri netlestirmek icin bakalim.Tablo gelicek simdi.Bunlar soket apileri tarafindan saglanan ana arayuzler.Bu soket API'leri HTTP programcisindan tum TCP ve IP detaylarini saklar.Soket API'leri UNIX isletim sistemi tarafindan ilk olarak tasarlandi.Zaten her seyin nerdeyse basi UNIX'te sonra herkes kendine pay cikarmis.Su an cok farli turleri var soketlerin her isletim sistemi ve dili icin.

Table 4-2. Common socket interface functions for programming TCP connections

Sockets API call	Description
s = socket(<parameters>)	Creates a new, unnamed, unattached socket.
bind(s, <local IP:port>)	Assigns a local port number and interface to the socket.

Table 4-2. Common socket interface functions for programming TCP connections (continued)

Sockets API call	Description
connect(s, <remote IP:port>)	Establishes a TCP connection to a local socket and a remote host and port.
listen(s,...)	Marks a local socket as legal to accept connections.
s2 = accept(s)	Waits for someone to establish a connection to a local port.
n = read(s,buffer,n)	Tries to read n bytes from the socket into the buffer.
n = write(s,buffer,n)	Tries to write n bytes from the buffer into the socket.
close(s)	Completely closes the TCP connection.
shutdown(s,<side>)	Closes just the input or the output of the TCP connection.
getsockopt(s, …)	Reads the value of an internal socket configuration option.
setsockopt(s, …)	Changes the value of an internal socket configuration option.

Soket API'leri TCP noktasinda veri yapisini tasarlamaniza izin verir.Bu noktalari uzak serverdaki TCP noktasiyla baglariz,ve data paketlerini okuyup veya uzerine bisiler yazabiliriz.TCP API'leri IP paketlerinden TCP data paketlerine donusturulurken yakalamayi,parcalara ayirmayi ve yeniden birlestirmeyi gibi tum detaylari altkatmandaki network protokollerinden saklar.

Ilk resimlerde,bir web tarayicisi power-tools.html web sayfasini Joe's Hardware deposundan HTTP'yi kullanarak nasil indiriyor gosteriyoruz.Simdi ki resimde soket API'leri kullanarak istemci ve server HTTP islemini nasil kullaniyor bakicaz.

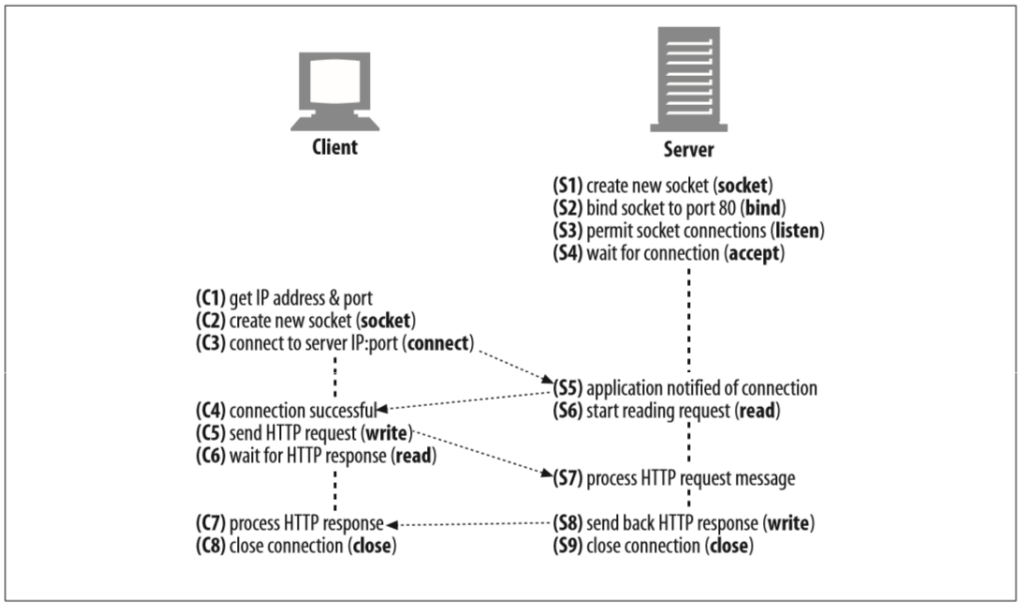

Figure 4-6. How TCP clients and servers communicate using the TCP sockets interface

S4'teki gibi web server baglanti bekliyor.Istemci IP adresi ve port'u URL'ye bakarak tanimlar ve server ile TCP baglantisi baslatir(C3).Birbirlerine baglanmak biraz zaman alabilir,serverin ne kadar uzakta oldugu,serverin yuklenmesi ve Internetin hizina bakar tabiki.

Bir kere baglanti kuruldu mu,istemci HTTP istegi yollar(C5),ve server bunu okur(S6).Server tum istek mesajini aldigi zaman,istek mesajini anlayip yorumlamaya baslar yani istegi isleme sokar(S7),ve istemciye istenen datayi geri yazar.Istemci okur(C6) ve cevap datasini isleme sokar(C7).

TCP PERFORMANS SIKINTILARI

HTTP TCP'de bir katmandir,HTTP islemlerinin performansi kritik bir sekilde TCP tesisatinin altinda yatan seylere baglidir.Bu kisimda TCP baglantilarinin belirli problemlerine ozetli bir sekilde bakicaz.TCP'nin karakteristik basic problemlerini anladikca,HTTP'nin baglanti optimizasyon yeniliklerini daha iyi anlayacaksiniz ve HTTP uygulamarini daha yuksek performansli yazabileceksiniz.

Bu bolum TCP protokolunun ic detaylarini bilmenizi gerektirecek.Eger ilgilenmiyorsaniz yada konuyu begenmediyseniz diger bolume gecebilirsiniz ama bence kalin.Cunku TCP karmasik bir konudur,Burada sadece kisa gene bir TCP performansindan bahsedicez.Sondaki TCP referanslarina bakabilirsiniz.

HTTP ISLEM GECIKMELERI

HTTP isteklerinde network gecikmelerinin nedenlerine TCP performansi uzerinden bir tur atalim.Asagidaki resim baglanti,transfer,ve islem gecikmesini HTTP isleminde gosteriyor.

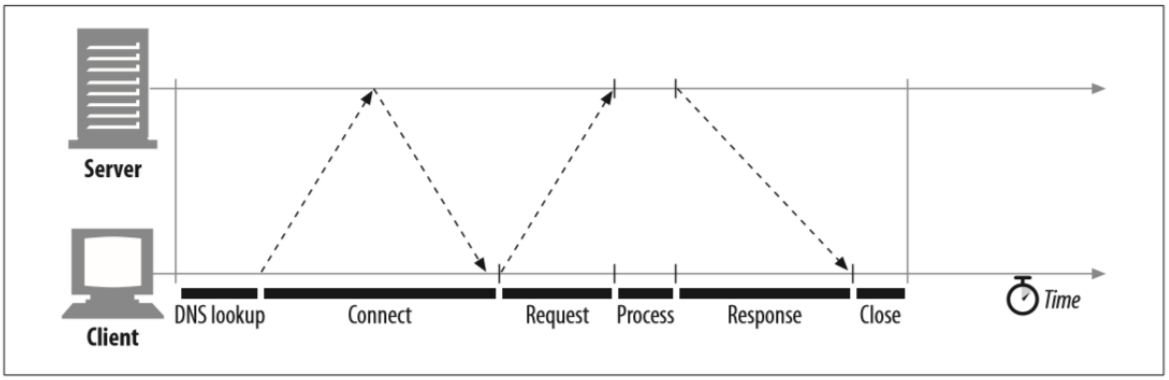

Figure 4-7. Timeline of a serial HTTP transaction

Sunu bilin ki islemler calisirken(Process bolumu yani) TCP baglantilari saglanirken ve istek,cevap mesajlarindaki zamanla oldukca kucuk fark vardir.Ancak istemci veya server fazla yukluyse veya karisik dinamik kaynaklari calistiriyorsa cogu HTTP gecikmesi TCP gecikmesine neden olur.

HTTP islemlerinde gorulebilecek gecikme sikintilari:

1.Istemci ilk olarak URL'den IP adresini ve portu tanimlamasi lazim.Eger URL'deki hostname son zamanlarda ziyaret edilmemisse,DNS cozumlemesini kullanarak URL'yi IP adresine cevirmek 10 saniye bile alabilir.

2.Sonra,istemci TCP baglanti istegi yollar servera ve serverin baglantiyi Kabul ettim demesini bekler.Baglanti yukleme gecikmesi her TCP baglantisi icin olur.Bu genellikle bir an yada iki zaman alir,ama yuzlerce HTTP islemi meydana geldiginde hizlica baglanti eklenebilir.

3.Bir kere baglanti saglandi mi,istemci TCP borularindan(pipe)HTTP istegi yollar.Web serveri kendisine TCP baglantisindan ulasan istek datasini alir ve isleme koyar.Istek mesaji icin Internet uzerinden gezinip serverda isleme girmek zaman alir.

4.Web serveri cevabi yazar ki buda zaman alir.

Bu onemli TCP ag gecikmeleri donanim hizina,network ve serverin yuklenisine,istegin ve cevabin boyutuna,istemci ve serverin uzakligina.Gecikmeler ayni zamanda mutlaq suretle TCP protokolunun teknik aksakliklarindan da meydana gelebilir.

PERFORMANS BOLGELERI

Su ana kadarki kisimda TCP ile ilgili gecikmeleri ana hatlariyla degerlendirdik.HTTP programcilarini nasil etkiledigini,ve hatta performans etkilerine neden oldugunu soyledik.Simdi maddelere bakalim:

1-TCP baglantisi el sikismasi

2-TCP yavas-basla tikaniklik kontrolu

3-Nagle'in algoritmasi data birlesimi icin

4-TCP'nin gecikmeli ACK(acknowledgement) algoritmasi icin sirtlama algoritmasi

5-TIME-WAIT gecikmeleri ve port tukenmesi

Eger yuksek performansli HTTP yazilimi yaziyorsaniz,bu faktorlerin hepsini anlamalisiniz.Eger bu seviyede bir performans optimizasyonuna ihtiyaciniz yoksa,bu bolumu gecmekte serbestsiniz.

TCP BAGLANTISI ELSIKISMA GECIKMESI

TCP baglantisi kuruldugunda,oncesinde veri gonderseniz bile,TCP yazilimi bir dizi IP paketinin degis tokus sartlari icin gorusucekdir.Resim geliyor bebegim.Bu degis tokuslar belirli sekilde HTTP performansini dusurebilir tabi eger baglantisi kucuk data transferleri icin kullaniliyorsa.

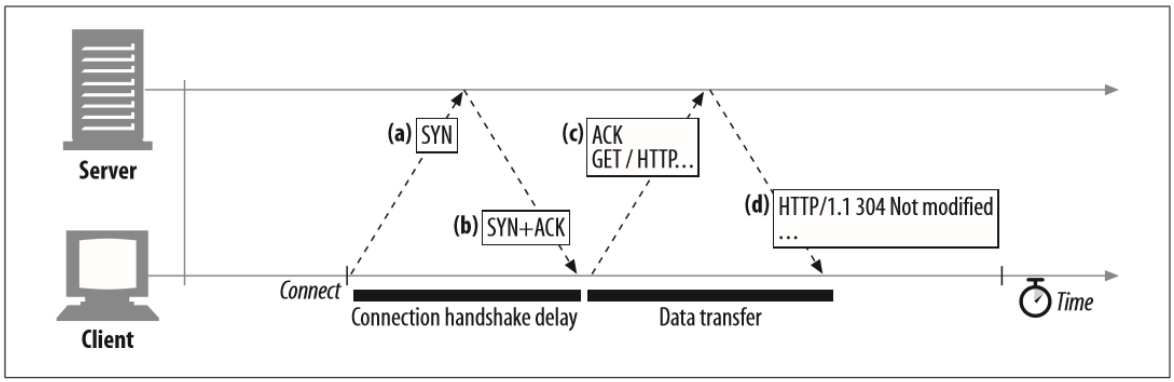

Figure 4-8. TCP requires two packet transfers to set up the connection before it can send data

TCP baglantisi el sikisma adimlari:

1.Istek icin yeni bir TCP baglantisi,istemci kucuk bir TCP paketi yollar(40-60 byte civari) servera.Paket "SYN" bayragina sahiptir.Baglanti istegi demek.a sikkinda gozukuyor.

2-Eger server baglantiyi Kabul ederse,bazi baglanti parametrelerini hesaplar ve istemciye geri TCP paketi yollar,hem "SYN" hemde "ACK" flagiyla beraber.Bu baglanti Kabul edildi demek.

3-Finalde,istemci "ACK" flagini servera geri yollar,derki biz baglandik arkadas haberin olsun.c sikki goruyorsunuz.Modern TCP yapisinda istemci ACK yollarken datada yollayabilir.

HTTP programcisi bu paketleri gormez,bunlar TCP/IP tarafindan gorunmez olarak halledilir.Tum HTTP programcilari TCP baglantisi yapildigi zaman sadece gecikmeyi gorur.

SYN/SYN+ACK el sikismalari resimde goruyorsunuz,HTTP islemi sirasinda cok fazla data alisverisi olmazsa olculebilir bir gecikmeye neden olur.Ki genelde bole olur.TCP baglantisinda ACK paketleri c sikkinda goruyorsunuz genellikle tum HTTP istek mesajini tasiyabilecek kadar buyuk olur,ve cogu HTTP server cevap mesajini bir IP paketine sigdirir.(Misal kucuk bir HTML sayfasi yada hata mesaji gibi).

Sonuc olarak kucuk HTTP islemleri %50 veya daha fazla sekilde zamanlari TCP kurulumu icin harcar.Sonraki bolumlerde tartisicaz her seyi.Gencler simdi bu ilk 4 bolumde pdfler attim ya ben githuba okuyup aklinizda bir sey kalmamis olabilir rahat olun bu ilk 4 bolumde her seyden bahsediyoruz sonra her bolum icin sadece kendimizi bir seye vericez o zaman her seyi mis gibi anlicaniz HEADSHOT yani.

ACK'DA GECIKME

Internetin kendisi guvenilir paket teslimini garanti etmiyor.(Internet routerlari paket cok yukluyse paketi imha edebilir mesela.)TCP uygulamasinin kendisi ACK semasinda data teslimini garanti eder.

Her TCP segmentinin sira numarasi vardir ve very butunlugu kontrolu yapilir.Alici her segmenti aldigi zaman servara kucuk ACK paketleri yollayarak kardesim paketin geldi der.Eger paketi gonderen(server) belirlenen cerceve zamanda ACK paketi almassa.Gonderici paket ya mahvoldu yada zarar gordu olarak varsayicak.

Cunku ACK paketleri kucuktur,yukarda 4.maddede sirtlama algoritmasi dedik ya iste o burda.TCP paketleri birdaha yollanir ACK cevabi donene kadar. Yani ACK data paketleri icinde beraber yollanir.TCP networku daha efektif kullanbilir.ACK paketlerinin hep gelmesini saglamak ve bir duzen oturtmak icin,cogu TCP duzeni "gecikmeli ACK" algoritmasini uygular.Gecikmeli ACK'nin olayi sudur.Gecikmeli ACK algoritmasi gidecek olan ACK'yi belirli bir cerceve zamaninda (genellikle 100-200 milisaniye) tutar. Yani her paket (100-200 milisaniyede bir gonderilir).Eger giden paket ben gittim diye geri ACK getirmiyorsa bu zaman dilimi icinde ACK kendi paketi icinde yollanir. Yani gencler paketler ust uste biner.

Ya dur bir dakka simdi anlicaniz.Gencler ACK paketleri bu algoritmada normalde cevapla beraber yollaniyor.Simdi cevapla beraber ACK'yi yolladim eger belirli zaman icinde (100-200)milisaniye icinde karsidan bir sey gelmesse paket bok yoluna gitmis varysayiyoruz ve sadece ACK yolluyoruz.Ulan zaten kullanilmio amma zorladik.:D

Maalesefki,cift modlu istek-cevap davranisi HTTP'nin ACK geri alma sansini engeller.Cift modlu istek-cevap davranisi derken su anki konudan bahsediyorum.Firat Aydinus gibi her seyi acikliyorum yav:D.Cogu paket istediginiz zaman ters yonde ilerlemez.Cogunluka bu algoritmayi devre disi birakmak bizi yeni belirli gecikmelerle tanistirdi.Isletim sisteminize bagli tabikide belkide suan da bilgisayarinizda bu algoritma var yada yok.

TCP yapisini yonetmeden once,ne yaptiginiza dikkat edin.TCP'nin icindeki algoritmalar zayif tasarlanmis uygulamalardan internet korumak icindir.Eger TCP ayarlariyla oynarsaniz,Tasarladiginiz TCP'nin kesinlikle uygulamada bir sorun cikarmayacagindan emin olmalisiniz.

TCP YAVAS BASLAR

TCP baglantisinin performansi TCP baglantisinin yasinada baglidir.TCP baglantilari gecen zamanla beraber kendilerini uydururlar,baslangicta baglantinin maksimum hizini limitlerler ve hizi zamanla arttirirlar data transferini basarili kilar,Bu ayak uydurmaya biz TCP yavas baslar(*TCP SLOW START)* diyoruz ve bu uygulama fazla yuklenmeyi ve tikanikligi onler.

TCP yavas baslar paketlerin sayisini kisar ve TCP noktasina her seferde bir kere ucus yaptirir.Kisacasi,her bir zamanda paketler basarili bir sekilde alinir,gonderici iki tane daha paket yollama izni alabilir.Eger HTTP islemi buyuk capli data yollamala ile ilgiliyse,tum paketleri tekte yollayamaz.Bir paket yollayip ACK icin bekleyip,sonra iki paket yollayip gene ACK taa ki 4 pakete kadar.Buna biz (tikanik pencereyi acmak diyoruz)"opening the congestion window."

Bu tikaniklik kontrolu olayi yuzunden,yeni baglantilar ayak uydurmus baglantilardan daha yavas ki bu baglantilar hemen makul miktarda datayi isleme sokuyor.Cunku alismis baglantilar daha hizli,HTTP kitabi yeniden varolan baglantilari kapsiyor,ilerleyen bolumlerde gorucez.

NAGLE'in ALGORITMASI VE TCP DUGUMLERI

TCP data akisi arayuzune sahiptir.Bu arayuzu uygulamalara istedikleri boyutta TCP datasi olusturmasina izin verir.-bir byte bile olabilir!.Ama her TCP segmenti en az flag ve headerlar

40 byte tasidigi icin,Network performansi agir bir sekilde bozulabilir eger TCP genis sayida paketlerde kucuk data yollarsa.

Nagle'in algoritmasi (John Nagle yaraticisi) paketleri yollamadan once TCP datalarini buyuk miktarda paketlemeyi amaclar,network efektikf yardimi RFC 896 algoritmasi "Congestion Control in IP/TCP Internetworks".Adamimizi baya ovmusler.

Nagle'in algoritmasi segmenti full sekilde gondermekten vazgecirmistir.(Lan icin 1500 byte,Internet icin bir kac yuz paket en fazla).Nagle'in algoritmasi size full dolu olmayan paket yollamaniza izin verir ancak diger paketlerin hepsi alindiysa izin verir.Eger diger paketler hala teslim ediliyorsa,kismi data korunur.Bu korunan data ya beklenen paketler alindiginda yada full sekilde yollanabilir diye hesaplandiginda yollanir.

Nagle'in algoritmasi birkac HTTP performans problemine sebep olur.Ilk olarak,kucuk HTTP mesajlari paketi doldurmaz,yani asla ulasmayacak bir datayi beklicekler.Ikinci olarak Nagle'in algoritmasi devredisi birakilmis ACK ler ile etkilesim icinde olucak.Nagle'in algoritmasi ACK ulasana kadar gonderilecek datayi tutar.Ama ACK algoritmasindan bildigimiz gibi ACK 100-200 ms gecikecek.

HTTP uygulamalari cogunlukla Nagle'in algoritmasi kapatir performansi arttirmak icin,TCP_NODELAY parametresini ayarlarlar(TCP DUGUMLERI).Eger bunu yaparsaniz.Buyuk datalarla ugrastiginiza emin olmalisiniz ki kucuk aceleci paketler olmasin.

TIME_WAIT HESAPLAMASI VE PORT TUKENMESI

TIME_WAIT port tukenmesi performansi etkileyen ciddi bir performans sorunudur,nadir bulunur.Ozellikle dikkat edilmelidir cunku cogu insan performans olayiyla ilgilenip bu sikintinin uzerine gitmis ama beklenmedik kotu performanslar elde etmislerdir.

TCP noktasi TCP baglantisini kapadigi zaman,kucuk kontrol bloklari seklinde son kapanan IP adresi ve portu hafizada barindirir.Bu bilgi kisa sure barindirilir.Segment zamanin yasama suresinin 2 kati(siklikla 2 dakka yani),ayni adrese ve port numarasina TCP baglantisi yapilmadigindan emin olmak icin bekler.Bu onceki baglantidan yolunu kaybetmis paketlerin yanlisikla yeni baglantiya enjekte edilmesini onler.Praktikte,bu algoritma Iki baglantinin ayni IP adresine ve port numarasina iki dakika icinde baglanip kapanmasina engel olur.

Bugunun en hizli routerlari o kadar ekstrem calisirki baglanti kapandiktan sonra bir dakika icinde serverin kapi esiginden paketler bitiverir.Bazi isletim sistemleri 2 dakikadan daha dusuk degerlere belirler.Ama bir seyin uzerine bir sey yazilmasin diye dikkatli olmaniz gerekir.Paketler kopyalanmis olur ve eger kopyalanmis paket ayni baglantiya gene eklenirse TCP verisi hasar gorur.

2 dakika olayi normalde sikinti degil ama kiyaslama durumlarinda olabilir.Bu siklikla bir veya cok az buluna yukleme-jenerasyon bilgisayarlarinda baglanti teslerinde bulunur,server baglanan istemci IP adreslerinin limitlerker.Ayrica server default olarak 80 portunu dinler,bu gibi durumlarda limit baglanti degerlerinin kombinasyonlaridir,bu durumda portlar TIME_WAIT tarafindan bloklanir.

4 degerimiz vardi TCP baglantisinda:

‹kaynak-IP=address,kaynak -port, hedef-IP=address,hedef-port›

3 u bunlari fix tir sadece kaynak portu degismekte ozgurdur.

‹istemci-IP,kaynak-port,server-IP,80›

Isetmci her servera baglanisinda,essiz baglanti yapmak icin yeni kaynak portu kullaniyor.Portlar limitli sayida ama(60000 tane de), 2 dakka desek 60000/120 den 500 islem cikiyor saniyede.

Eger optimizasyon yapmaya devam ederseniz ve serverin 500 islem/saniye den hizli degilse TIME_ WAIT port tukenmesiyle karsi karsiya kalmazsiniz.Daha fazla istemciyle yada daha fazla IP adresi kombinasyonuyla halledebilirsiniz sikintiyi.

Port tukenmisi problem cekmeseniz bile cok fazla sayida baglantiniz olmamasina dikkat edin.Bazi isletim sistemleri uzucu bir sekilde acik baglnati yada kontrol bloklari varsa yavasliyor.

HTTP BAGLANTI YONETIMI

Bu bolumdeki ilk iki kisim TCP baglantilarinin ve performarnslarinin uzerine bir turdu.Eger TCP networkuyle ilgili daha fazla sey ogrenmek istiyorsaniz,bolumun sonundaki kaynaklara bakabilirsiniz.

Olaylari hizlica degisiyoruz ve HTTP'ye geri donuyoruz,geri kalan kisimda HTTP teknolijisinin baglantilari nasil yonettigini ve optimize ettigini gosterecegiz.HTTP baglanti headerlari ile basliyoruz,siklikla yanlis anlasilmistir,ama HTTP baglanti yonetimi icin onemlidir.O zaman konusmaya baslayalim.

YANLIS ANLASILAN BAGLANTI HEADERLARI

HTTP aslinda HTTP zincirleri olarak istemci ve server arasinda aracilik yapar.HTTP mesajlari hoplaya hoplaya istemciden,araci cihazlarla,servera gider(yada tersi).

Bazi durumlarda,iki tane komsu HTTP uygulamasi paylasimli baglantilarinin ayarlarini duzenlemek isteyebilirler.HTTP baglanti headerlari virgule ayrilan listelenmis baglanti tokenleridir.Baglanti ayarlarini belirlerler.Ornegin,mesaj yollandiktan sonra baglantinin kapanmasi gerek:close

Baglanti headerlari bazen kafa karistirici olabilir,cunku uc farkli token tasiyor.

1-HTTP header alan adi,sadece o baglanti icin uygun headerlari listeler.

2-Keyfi token degerleri,o baglanti icin standard olmayan baglantilari tanimlar.

3-Deger "close", kalici baglanti olay bittiginde kapansin.

Eger baglanti tokenlari header Alani barindiriyorsa,header alani baglantiya ozel seyler barindirir.Herhangi bir header alani baglanti icin ekleniyosa mesaj yollanmadan once

silinmeli.Hoplaya ziplauya headeri deninen(hop-by-hop)headeri "koruyucu header olarakta bilinir",cunku Baglanti headerlari kazara iletilen local headerlardan korur.Resim geldi.

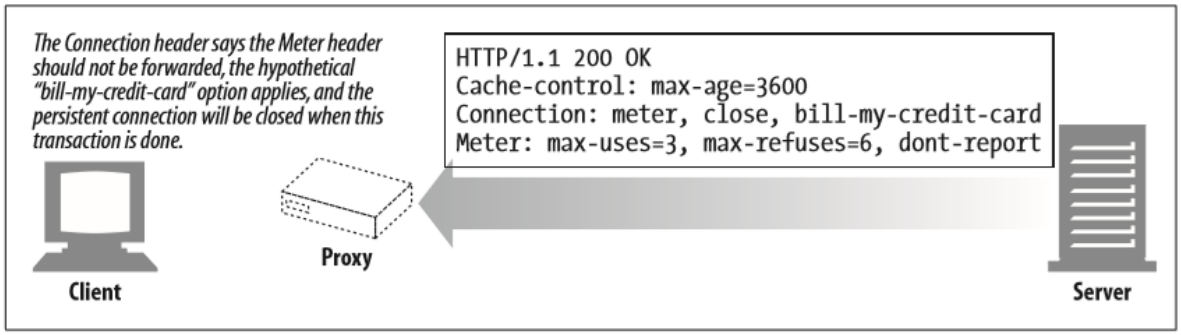

Figure 4-9. The Connection header allows the sender to specify connection-specific options

HTTP uygulamasi Baglanti headeri ile beraber mesaji aldigi zaman (Resimdeki Connection bolumu),Alici istemciden alinan tum ayarlari uygular.Sonra Baglanti headeri ve listelenmis diger tum headerlar mesaj diger noktaya iletilmeden once silinir.

SERI ISLEM GECIKMELERI

Eger baglanti cok bole nahos yonetiliyorsa,TCP performansi gecikmesi ortaya cikabilir.Ornegin,3 tane resme sahip olan bir web sayfaniz var.Tarayiciniz 4 tane HTTP isleminden gecmesi gerekiyorki.Ilki HTML sayfasi icin ve diger 3'u resimler icin,Her islem yeni bir baglanti ve baglantilar yavas-basla gecikmesine sebep olur.

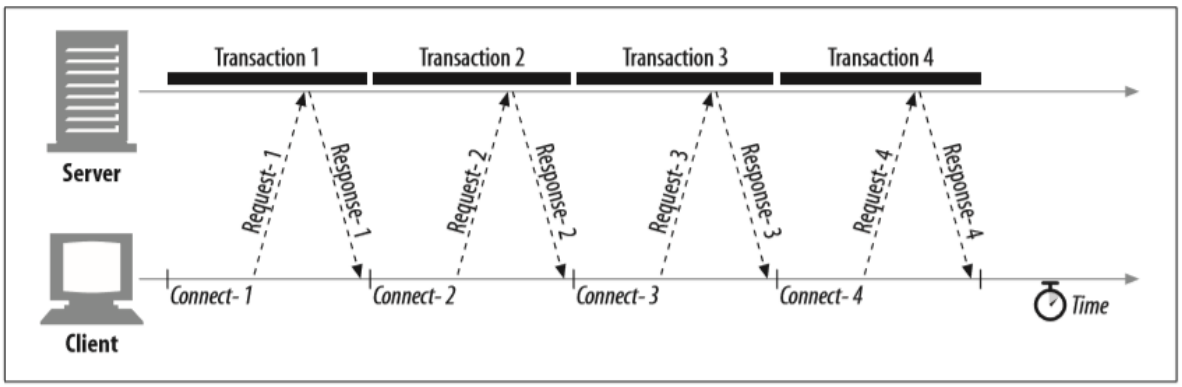

Figure 4-10. Four transactions (serial)

Ek olarak gercek zaman gecikmesi seri yuklemelerde gerceklesebilir,ayni zamanda psikolojik olarakta dusundugumuzde bir image yukleniyor ve sayfanin geri kalaninda hic bisi olmuyorsa burdada bir yavaslik soz konusu.Kullancilar genellikle ayni anda birden fazla resmin yuklenmesi tercih eder.

Seri yuklemelerin bir diger dezavantajida bazi tarayicilar yeterli obje yuklenmeden sayfada bir sey gostermezler,cunku objelerin boyutunu yuklenene kadar bilmiyorlar,ve daha resimlerin sayfada hangi konumda bulunduklarini falan soyleyecekler.Bu durumda,tarayici objeleri seri olarak yukleyebilir,ama kullanici belkide bos beyaz bir sayfa ile karsilasabilir,ve arkada islemlerin oldugundan haberi yok.

HTTP baglanti performansini guclendirmek icin birkac mevcut teknik var.Onumuzdeki kisimlarda bundan bahsedecegiz.

PARALEL BAGLANTILAR

Es zamanli HTTP istekleri birden fazla TCP baglantisiyla beraber

KALICI BAGLANTILAR

TCP baglantilarini yeniden kullanarak baglanti/kapatma gecikmesini gidermek

BORU HATTI BAGLANTILARI

Es zamanli HTTP istekleri paylasimli TCP baglantisiyla beraber

COKMESAJLI BAGLANTILAR

Istek ve cevaplarin yigin seklinde birbirine gecirilmesi

PARALEL BAGLANTILAR

Daha oncede bahsettigimiz gibi,tarayici narin bir sekilde istekteki HTML sayfasinin icindeki resimleri teker teker getirebilir.Ama lanet olsunki cok yavastir.

HTTP ayni zamanda istemciye birden fazla baglanti ve birden fazla HTTP islemini parallel olarak yapmasina olanak saglar.Ornegin,4 tane resim parallel olarak yukleniyor ve her islem TCP baglantisi ile saglaniyor.Asagidaki resimde goreceniz simdi.

PARALEL BAGLANTI SAYFAYI DAHA HIZLI YUKLER

Bilesik sayfalar olusan objeleri daha hizli yukler tabi olu zaman ve bandgenisligi limiti acisindan avantajlilarsa.

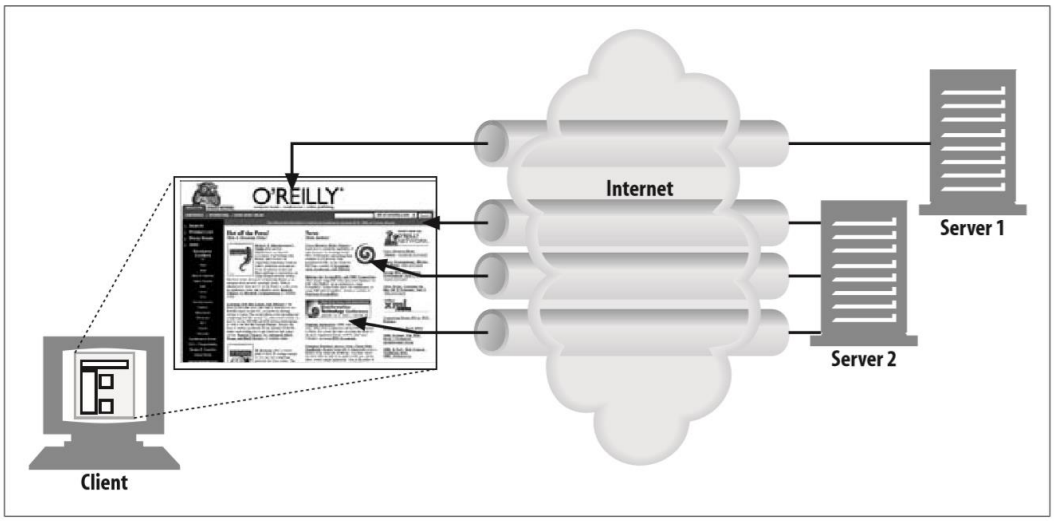

Figure 4-11. Each component of a page involves a separate HTTP transaction

Gecikmeler ustuste binebilir,ve eger tek bir baglanti bandgenisligini doldurmassa,kullanilmamis bandgenisligi fazla obje yuklemesine yardimci olacaktir.

Asagidaki resim parallel baglantilar icin bir zaman cizelgesi.Figure 4-10. Daki resimden kesinlikle daha hizli(resim yukarda bulursunuz hemen).Kapsanilan HTML sayfasi ilk yuklenir,geriye kalan uc islem eszamanli calistirilir,her biri kendi baglantisina sahip.Cunku resimler parallel yuklendi,baglanti gecikmesinin ustesinden gelindi.

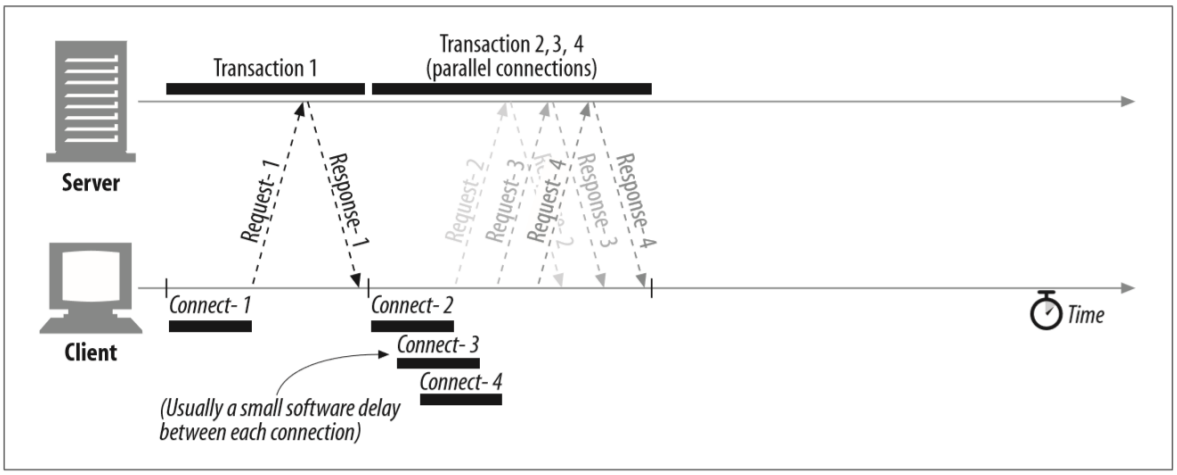

Figure 4-12. Four transactions (parallel)

PARALEL BAGLANTILAR HER ZAMAN HIZLI DEGILDIR

Paralel baglantilar hizli olabilsede,ne varki,her zaman hizli degillerdir.Istemcinin bandgenisligi kisitli oldugu zaman(ornegin,28.8-Kbps modem dusun az yani),cogu sadece zamanini data transfer etmeye yarayacaktir.Bu durumda,bir tane HTTP islemi,hizli bir serverla hasir nesir oldugunda modemin tum bandgenisligini harcayacaktir.Eger birden fazla obje parallel olarak yuklenir,Objeler limtli bandgenisligi icin birbirleriyle yarisacak,yani her obje yavasca yuklenicek,bahsettigimiz performans avantaji olmayacaktir.

Ayni zamanda,buyuk rakamlarla yapilan acik baglantilar cok fazla hafiza harcayip performans sikintisina sebep olacaktir.Karisi web sayfalarindan on veya yuzlerce obje gomuludur.Istemci yuzlerce baglanti acabilecek durumda olabilir,ama cok az web server bunu yapmak isteyecektir,cunku kullanicilar istegi gonderirken baska kullanicilarda ayni zamaanda ayni serverda takiliyolar caktin.Yuzlerce es zamanli kullanici,her biri **100** sayfa acsa,**10000** baglanti server yuklenicek.Serverin yavaslamasina sebep olur,ayni durum proxyler icinde gecerli.

Pratikte,tarayicilar hep parallel baglanti kullanirlar,ama parallel baglantidan sinirlari var genellikle(4 tane).Serverlar asiri yuklenimde baglantilari kapatmakta serbestler.

PARALEL BAGLANTILAR HIZLI HISSETTIREBILIR

Tamamdir,parallel baglantilar her zaman sayfayi hizli yuklemezler,aslinda sayfa transferini hizlandirmazlar,onceden de dedigimiz gibi,parallel baglantilar siklikla hizliymis gibi hissettirirler,cunku onlar birden fazla objenin bileseni parallel olarak gorundugunde direk ekrana getirdikleri icin ole gozukuyorlar.Insan cercevisnden bakinca daha hizli tabiki,ama sayfada cok fazla aksiyon donerse,hatta indirme hizi daha yavas olsa bile bizim gozumuzden ole gozukmuyor.

KALICI BAGLANTILAR

Web istemcileri(yani biz)siklikla ayni sitelere baglaniriz.Ornegin,cogu web sayfasindaki resim cogu zaman ayni web sitesinden gelir,ve onemli sayidaki objelerin linkleri bile ayni siteyi gosterir.Boylece,Servera HTTP istegi yapan uygulama daha fazla istek yollayabilir.Bu uygulamaya biz sit alani diyelim(site locality).

Bu nedenle,HTTP/1.1(ve HTTP/1.0'in gelismis versiyonlari)HTTP islemi tamamlandiktan sonra bile TCP baglantisini acik tutar gelecekteki HTTP istekleri icin.TCP baglantilari islem bittikten sonra acik kaliyor ya biz bunlara ***kalici baglantilar*** diyoruz.Kalici olmayanlar islem bitince kapanir.Kalici baglantilar ise islemden sonra acik kalir.Kapanmasina server yada istemci karar verir.

Hep bosta kalarak,kalici baglanti hedef serverda aciktir,yavas baglantiyi siz onleyebilirsiniz.Ek olarak,acik olan baglantilar yavas-basla olayini onlerler,daha hizli data transferi icin.

KALICI VS PARALEL BAGLANTILAR

Gordugumuz gibi,parallel baglantilar karisik sayfalarin transferlerini hizlandirabilir.Ama dezavantajlarida var tabi.

1-Her acil/kapan islemi yeni bir baglanti,buda zaman ve bandgenisligi demek.

2-Her yeni bir baglanti performansi zedeler cunku TCP yavas basladir.

3-Paralel baglantinin praktikte bir limiti var.

Kalici baglantilar parallel baglantilar uzerinde avantajlari var.Gecikme ve fazla baglanti olayi azaltiyorlar,baglantiyi duruma gore sagliyorlar,ve acik baglanti sayisini azaltiyorlar.Ancak,kalici baglantilar dikkatli yonetilmeli,veya cok fazla acik baglanti birakabilirsiniz,istemcideki ve serverdaki tum kaynaklari yer.

Suan bilgisayariniza bir netstatla baksaniz yada networkminerla baya bir acik baglanti gorursunuz.

Kalici baglantilar parallel baglantilarla beraber kullanildigina cok efektif olurlar.Bugun,cogu web uygulamasi kucuk sayida parallel baglanti aciyorlar ve her biri

kalici.Iki turlu kalici baglanti var:Yasli HTTP/1.0+ "keep-alive" baglantisi ve suan modern olan HTTP/1.1 'kalici" baglantisi.Ikisinede onumuzdeki kisimda bakicaz.

HTTP/1.0+Keep-Alive Baglantilari

Cogu HTTP/1.0 tarayicilari ve serverlari (1996'dan baslayip genisleyerek) eski yillarinda destekliyordu,kalici baglantinin deneysel tipi keep-alive baglantisidir.Gecmis yillardaki kalici baglanti birlikte calisma dizayni probleminden baya cekmistir.HTTP/1.1 da duzeltilmistir,ama cogu istemi ve server hala eski keep-alive baglantisini kullaniyor.

Bazi performans avantajlari asagidaki resimde gozukuyor,4 HTTP islemi olan seri baglantiyla ayni islemi kalici baglantiyla yapan baglantilari zaman cizelgesinde karsilastiriyor.Zaman cizelgesi sikistirilmis cunku baglanti ve giderler kapanip silinmistir.

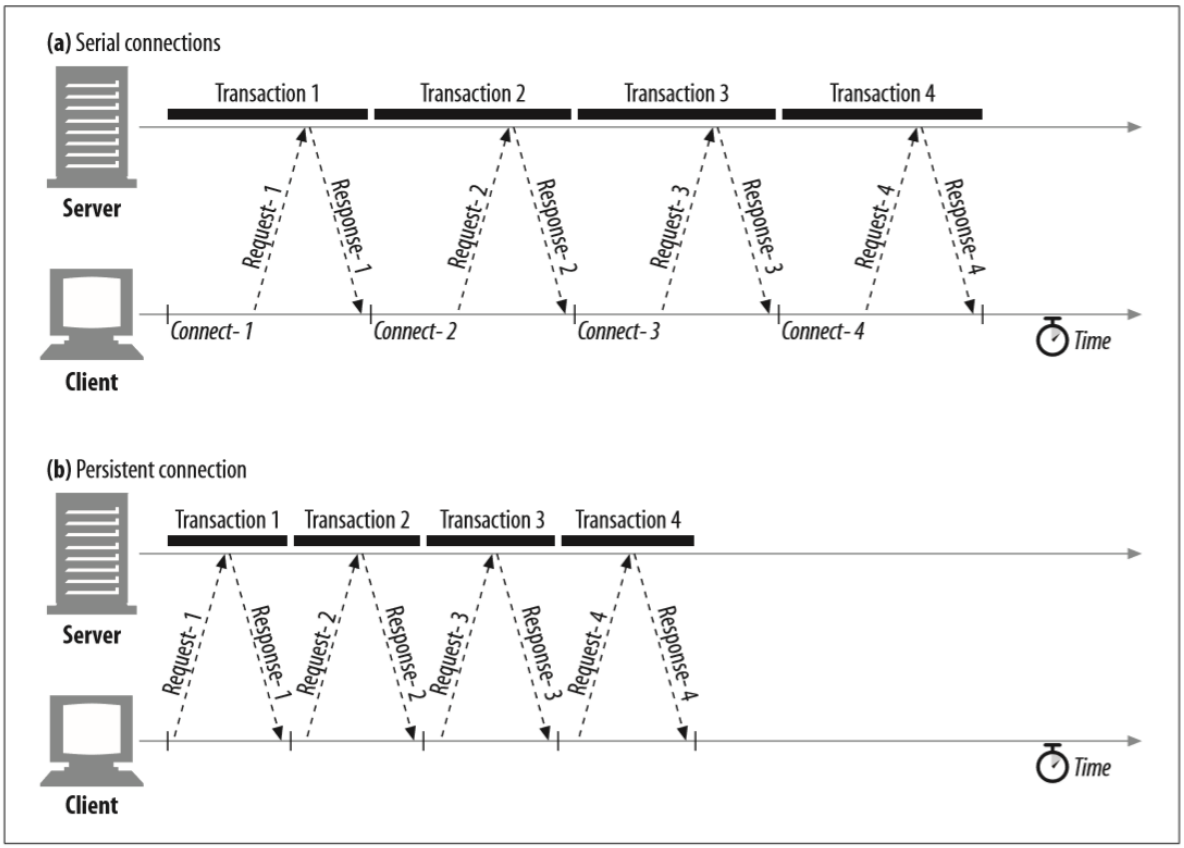

Figure 4-13. Four transactions (serial versus persistent)

KEEP-ALIVE CALISMASI

Keep-alive kullanimdan kaldirildi ve artik mevcut HTTP/1.1 documentlerinde mevcut degil.Ancak,keep-alive el sikismasi hala guvenilir server ve browserlar tarafindan kullaniliyor,yani HTTP uzerine calisanlar onu birlikte calisabilirlige hazirlamaya basladilar.Simdi keep-alive calismasina hizlica goz gezdiricez.HTTP/1.1 eski versiyonlarinin taslaklarinda keep-alive el sikismasi cok daha iyi aciklanir(RFC 2068).

Istemci HTTP/1.0 da keep-alive baglantisi isteklerde Connectionda:Keep-Alive istek headeri bulunursa acilabilir.

Eger server ilerideki istek icin baglantiyi acik tutmak isterse,ayni header ile cevap verir.Eger hic bir Connection:keep-alive headeri cevapta yoksa,istemci serverin keep-alive desteklemedigine ve server cevap mesajini geri gonderdiginde kapanacagini varsayar.

Keep-Alive Ayarlari

Sunu bilingi keep-alive headerlari istekte olur ve keep-alive(canli tut) baglantiyi canli tutmaya yarar.Istemciler ve serverler keep-alive istegi var diye Kabul etmek zorunda degillerdir.

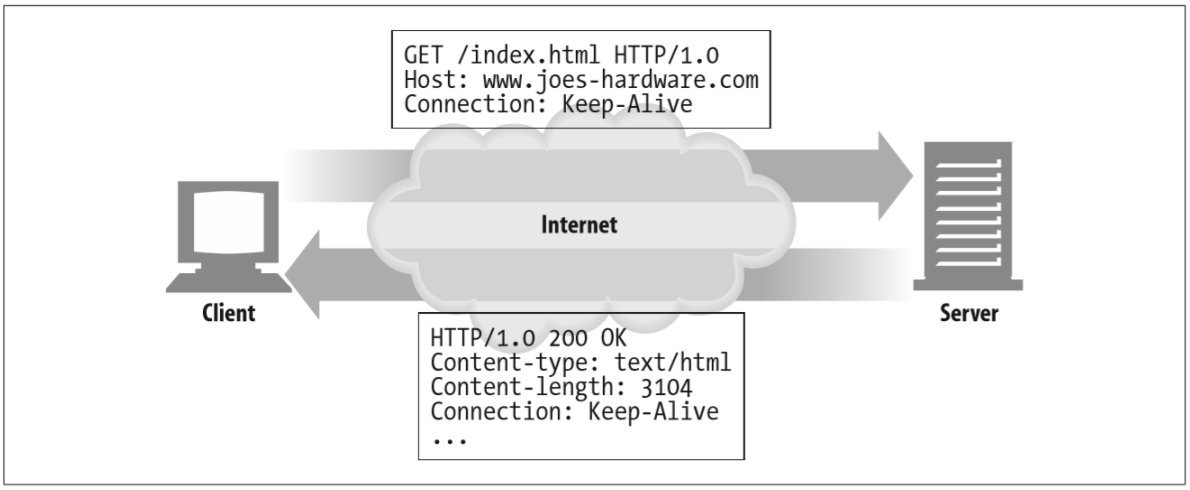

Figure 4-14. HTTP/1.0 keep-alive transaction header handshake

Keep-alive baglantilari herhangi bir zamanda kapatabilirler ve limit yoktur istedikleri kadar keep-alive baglanti isleminde bulunabilirler.

Keep-alive genel headeri ozellikleri virgule ile ayrilir:

Timeout:Keep-alive cevap headeridir.Serverin baglantiyi ne kadar canli tutacagini soyler.Garanti degildir.

Max:Keep-alive cevap headeridir.Serverda genellikle ne kadar sureyle keep-alive HTTP islemi oldugunu soyler.Garanti degildir.

Keep-alive headeri ayni zamanda keyfi olarak islem sureciyle alakali olmayan seyleri destekleyebilir.name=value gibi.

Keep-Alive headeri tamamiyla opsiyoneldir ama sadece Connection:Keep-Alive sunuldugu zaman gecerlidir.Asagida Keep-Alive ornegi yapicam simdi.

Connection:Keep-Alive

Keep-Alive:max =5, timeout=120

Burda serverin 5 veya daha fazla islem icin baglantinin tutuldugunu ve 2 dakikasi oldugunu soyler.

KEEP-ALIVE BAGLANTI KISITLARI VE KURALLARI

Burda simdi bazi kisitlar ve aciklamlar yapicaz keep-alive kullanimi ile ilgili:

Keep-alive HTTP/1.0 defaultinda gerceklesmez.Istemci Connection:Keep-Alive istek headerini yollamali ki keep-alive baglantisini aktif etsin.

Connection;Keep-Alive headeri tum mesajlari kalici ve devamli olarak yollamali.Eger istemci Connection:Keep-Alive headeri gondermezse,server istekten sonra baglantiyi koparacaktir.

Eger server Connection:Keep-Alive cevap headerini yollayip baglantiyi kapatirsa Istemci bunu server soyleyebilir.

Eger baglanti kapanmadan once mesajin boyutu tespit edilirse baglanti kapanmaz.

Proxyler ve gatewayler Conenction headeri kurallari uygulamak zorundalar;proxy ve gateway Connection headerindaki herhangi bir header alanine silebilir ve Connection headerinin kendisinide ,mesaj yollanmadan ve onbellege alinmadan once.

Resmi olarak,keep-alive baglantilari proxy serveriyla anlasmazlar,yani proxyler Connection headerini desteklemeyi garanti etmez,bu problemi onlemek icin dilsiz proxyler vardir asagida bakicaz zaten.Praktikte her zaman mumkun degildir ama.

Teknik olarak,herhangi bir Connection header alani(Connection:Keep-Alive da mevcut) HTTP/1.0 cihazindan alindi mi gormezden gelinir,cunku eski proxy server uzerinden iletilmis olabilir.Pratikte,bazi istemciler ve serverlar bu kurali kirabilir,ama eski proxylerle dans ederlerse isleri zor.

Istemciler istekleri tekrar etmeye hazir olmali cunku tum cevabi almadan once baglanti kapanirsa,istek tekrarlanir.

KEEP-ALIVE ve DILSIZ PROXYLER

Simdi su guzel problemimize keep-alive ve dilsiz proxylerle bakalim.Web istemcisi Connection : Keep-Alive headeriyla amacladigi sey istemciye sadece tek bir TCP birakmak.Bu yuzden adi "Connection headeri zaten.Eger istemci web serveriyla konusursa.Istemci Connection:Keep-Alive headeriyla server keep-alive yani baglantisinin hayatta kalmasini istedigini syoler.Server Connection:Keep-Alive baglantisi destekliyorsa geri gonderir.Desteklemiyorsa gondermez.

CONNECTION HEADERI VE KOR YAYIN

Proxylerle gelen problem-ozellikle,proxyler Connection headerini anlamaz ve bilmezler,ve proxyler gecirmeden once headeri silmek zorundadirlar.Cogu eski veya basit proxyler kor yayin olarak davranirlar,bytelari bir baglantidan digerine tunellerler,ozelikle Connection headeri isleme gecmeden.

Web istemcisinin web serveriyla dilsiz proxy aracigiliyla konustugunu hayal edin.Dilsiz proxy kor yayin gibi davranicak. Resimde goruceksiniz simdi.

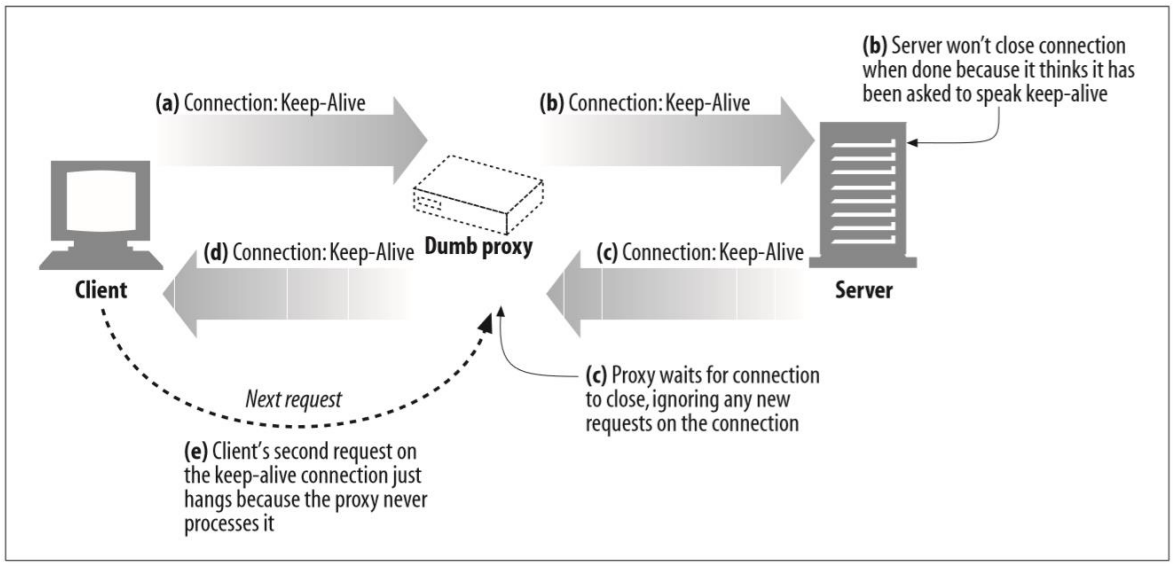

Figure 4-15. Keep-alive doesn't interoperate with proxies that don't support Connection headers

1.Yukardaki resimde a kisminda,web istemcisi proxye mesaj yolluyor.Dumb proxy(Dilsiz proxy).Connection:Keep-Alive headeri ile beraber yollar tabi bole bir baglanti mumkunse.Istemci cevabi beklerki istek keep-alive olayi saglanabiliyor mu ogrenmek icin.

2.B kisminda,Dilsiz proxy HTTP istegini alir,ama Connection headerini anlamaz(genisletilmis headeri diye davranir.)Proxy keep-alive nedir falan haberi yok adamin bilmiyor kara cahil.Yani mesaji servera direk dokunmadan gecirir.Ama Connection header her yerden hoplaya ziplaya gecen bir header sadece tek ulasim linkinde gecirilmeli ve proxy gecirmemeliydi.Kotu seyler olmak uzere.

3-Gene b kismindayiz,gecirilen HTTP istegi server ulasti.Server proxylenmis(tunnellenmis diyelim) Connection:Keep-Alive headerini aldigi zaman,Server soyle bir sonuca varir(serverin gozunde proxy bir istemcidir nihayetinde).Bu istemci arkadas keep-alive sekilde konusmak istiyor yaaav! der,tabi bu bir hatadir.Server icin hava hostur,keep-alive olayini Kabul eder ve Connection:Keep-Alive cevap headerini c sikkindaki gibi yollar.Simdi,bu noktada,server proxy ile keep-alive sekilde konusucagini dusunur ve keep-alive kurallari cercevesinde baglantiya devam eder.Ama proxy keep-alive ne onu bile bilmiyor.Hadi buyrun,amerikan baskani dahil herkesi devreye sokun uzaylilar tarafindan kacirildim,evet tarafindan.Ugandaca size herkesin bisi soyledigini dusunun ve sizden cevap bekliyolar cunku bilionuz saniolar.O yuzden dilsiz proxy diyoruz zaten caktin!.

4-D kisminda,dilsiz proxy serverdan aldigini cevap olarak istemciye iletir,Connection:Keep-Alive headerinida gecirir.Istemci headeri gorur ve proxy keep-alive konusmayi Kabul etti diye varsayar.Bu noktada istemci ve server keep-alive sekilde konusuyoruz sanar.Ama konustuklari proxy keep-alive diye bir sey bilmiyor arkadaslar.

5.Cunku gene diyoruz Proxy bilmiyor keep-alive nedir.Bu salak saniyorki serverdan gelen datalari istemciye yollicam ve server baglantiyi kapatsin diye beklicem.Ama server

kapatmiyor baglantiyi arkadasim aciyorum bu proxye ye yav.Istemci inaniyorki proxy acikca servera baglantiyi acik tut dedi.Proxyde baglanti kapansin diye takiliyo oralarda ole.

6-Istemci d sikkindaki gibi cevap mesajini alinca,diger istegi hemen yolluyor,keep-alive baglantisi var diye yolluyor proxye istegi ama e sikkinda gordugunuz gibi ih ih.Cunku proxy baska bir istegi ayni baglantidan beklemiyor,istek gormezden geliniyor ve istemci zbam zbam yardiriyo istekleri ama hic bir islem yok gencler.

7-Bu yanlis iletisim browserin istemcinin veya serverin zaman asimindan baglantiyi kapatmasina kadar takiliyor kendi dunyasinda:D.

Proxyler ve atlayan headerlar

Bu tarz yanlis anlasilan iletisim hatalarini engellemek icin,modern proxyler asla Connection headeri veya herhangi bir headeri icinde Connection degeri bulunduran diyelim hicbir headeri gecirmemeli.Eger Proxy Connection:Keep-Alive headeri alirsa,hicbir turlu gecirmemeli herhangi bir Connection Headerida dahil.

Ek olarak,azcik hoplayan ziplayan atlayan yerinde duramayan hiperaktif(benim gibi)headerlar var Connection headeri gibi, bunlar gecirilmemeli ve hizmete sunulmamali.Proxy-Authenticate,Proxy-Connection,Transfer-Encoding,ve Upgrade gibi headerlarda dahil.Daha fazlasi icin geriye donup *Siklikla yanlis anlasilan baglanti headerlari basligina bakabilirsiniz.*

PROXY-BAGLANTISI HACKLEME

Istemci ve proxyi yaratanlar(Netscape sirketi) onerdilerki akillica bir gecici cozum kor yayin problemi icin.Tum web uygulamalari HTTP gelismis versiyonlarini desteklemesine gerek kalmadan halledelim.Gecici cozumu Proxy-Connection headeri ile tanitildi ve kor yayin istemciye yapildiktan sonra orda olayi cozebilcek sekilde devereye girdi,ama diger durumlari icin bi rise yaramadi.Proxy-Connection proxyler acikca ayarlanip ve anlisilir hale geldikten sonra modern tarayicilar tarafindan uygulandi.

Fikir suydu,dilsiz proxyler sikintiya sebep oluyor cunku onlar atlayan ziplayan her headeri geciriyor Connection:Keep-Alive gibi.Atlayan ziplayan headerlar sadece tek,ozel baglantilar icin guvenli ve hala da deniyor ki iletilmemeli.Bu headerlar yanlis yorumlanarak serverlara istek olarak proxy tarafindan iletildigi zaman sikintiya sebep oluyor.

NetScape de gecici cozum su.Bilinen resmi olarak desteklenen Connection headeri yerine Tarayici standart olmayan genisletilmis Proxy-Connection headerini proxy'ye yollasin, Eger proxy kor yayinsa,amcasiz olan Proxy-Connection headerida serverda belircek zaten,ki sonradan zararsizca gormezden geliniyo bu header.Ama eger proxy akilliysa(yani kalici elsikisilan baglantiyi anlayabilecekse yani keep-alive olayini cakabilicekse)Proxy-Connection headeri ile Connection headerini yer degistirip,sonra servera yollayacak ve arzu edilen efekt gerceklesmis olacak.

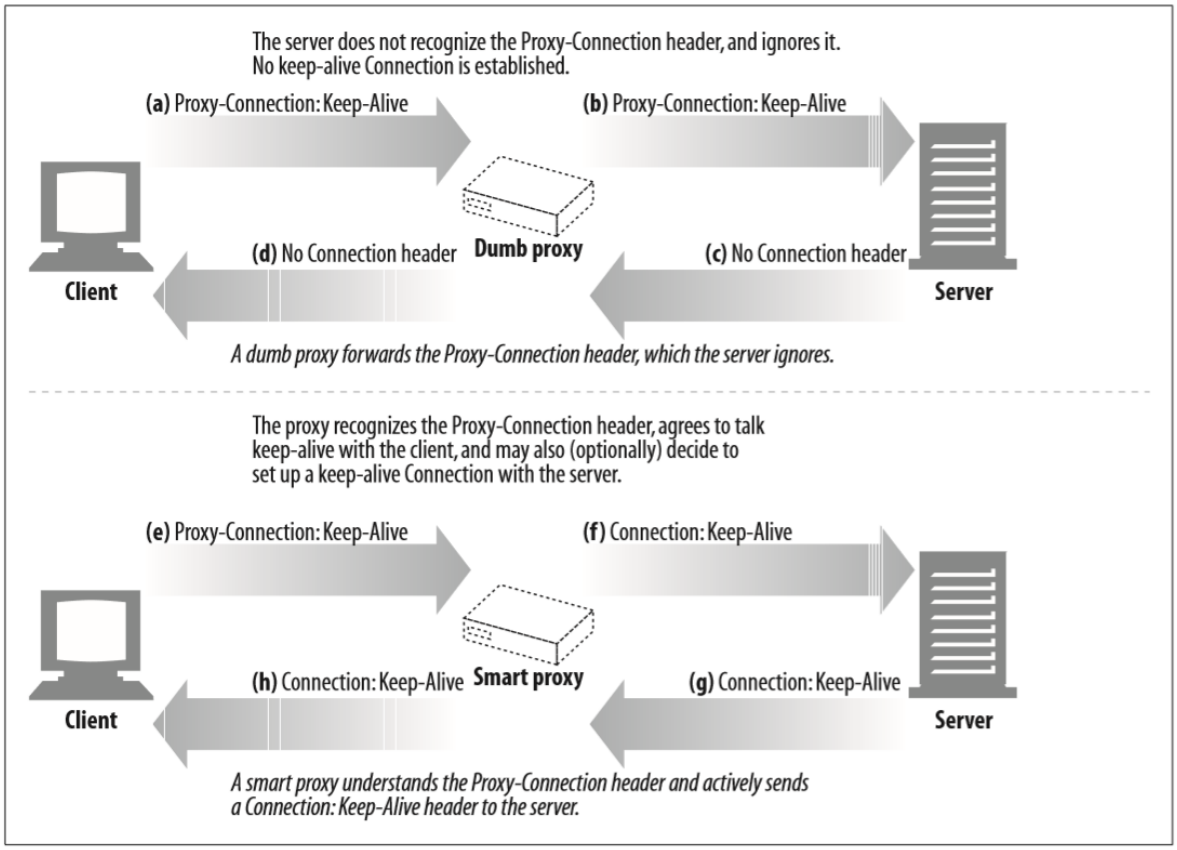

Figure 4-16. Proxy-Connection header fixes single blind relay

Resimdeki a ve d gosteriyorki Proxy-Connection headeri servera kor yayin olarak zararsizca iletiliyor,server headeri gormezden geliyor yukarda dedigimiz gibi,ve keep-alive baglantisi olmamasina sebep oluyor.Obur turlu noluyor biliyorsunuz.Akilli adamin hali bir baska diyoruz ve akilli proxy e ve h sikkinda gordugunuz gibi Proxy-Connection headerini anliyor ve keep-alive istegi olarak davraniyor ve kendi keep-alive baglantisini sagliyor.

Bu sema istemci ve server arasinda bir proxy oldugu zaman calisiyor.Ama proxyinin herhangi bir tarafinda dilsiz proxy(mal proxy) varsa problem gene cosup bagiriyor asagida resmedicez konusmamiz bitsin.

Ayrica,yavasca networklarda,firewallarda,cache(onbellek) kesmelerinde,veya ters proxy hizlandiricilarinda gorunmez proxy olarak bulunmaya basladi,Cunku bu cihazlar tarayici icin gorunmez,tarayici onlara Proxy-Connection headeri yollamiyor.Seffaf Web uygulamalarinin kalici olarak dogru bir baglanti saglamasi icin kritik bir durum.

HTTP/1.1 Kalici Baglantilari

HTTP/1.1 keep-alive baglantilarini asamali olarak destekledi.keep-alive ile daha iyi tasarlanmis *kalici baglanti* dedigimiz seyi degistir.Kalici baglantida amac,keep-alive ile ayni ama mekanizma tabiki daha iyi isliyor.

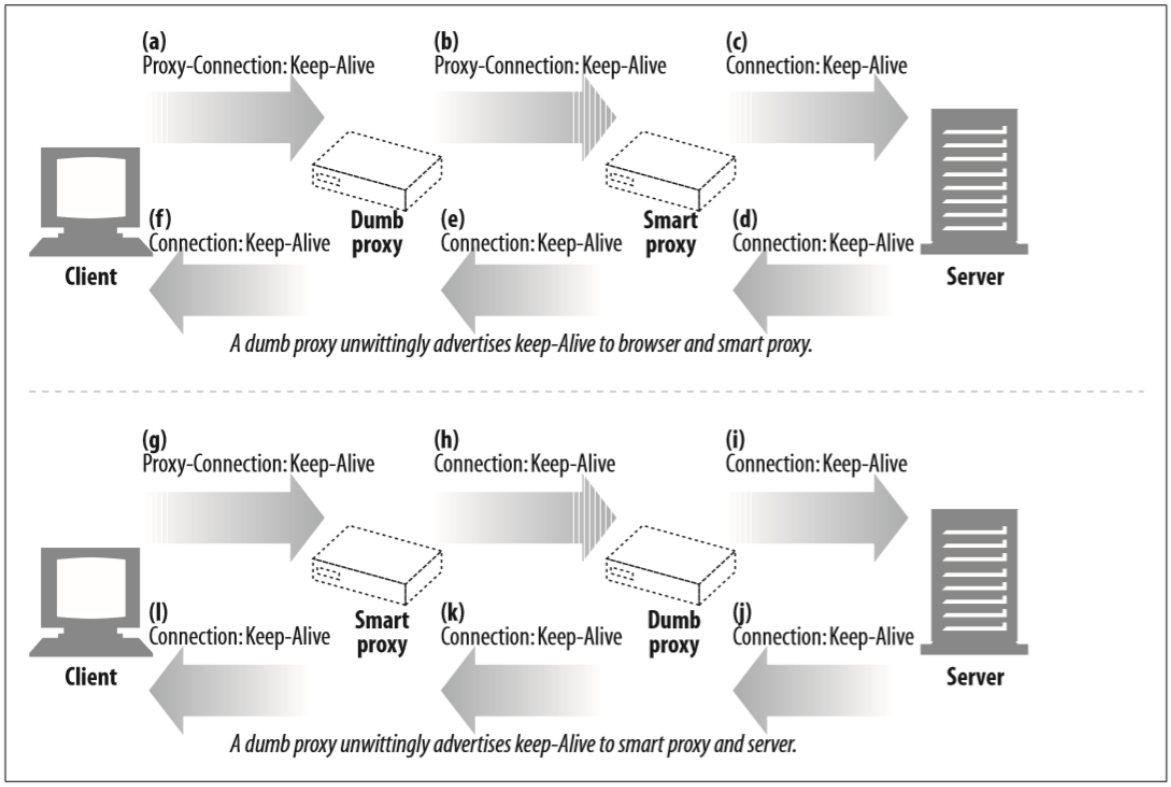

Figure 4-17. Proxy-Connection still fails for deeper hierarchies of proxies

HTTP/1.0+ keep-alive baglantilarina nazaran,HTTP/1.1 kalici baglantilari default olarak aktiftir.HTTP/1.1 tum baglantilari aksi bir durum olmadikca kalici sayar.HTTP/1.1 uygulumalari acikca Connection: close headerini mesaja eklemeliki islemler tamamlandiktan sonra baglanti kapansin.Bu HTTP'nin onceki versiyonlarindan onemli fark oldugunu isaret eder.Biliyosunuz keep-alive baglantisi opsiyonel ve tamamiyla desteklenmez bir olaydi.

Bir HTTP/1.1 istemcisi HTTP/1.1 baglantisinin cevaptan sonra acik olarak kaldigini varsayar,tabi cevap Connection:close headerina sahip degilse.close degeri varsa baglanti kapanacaktir.Ancak,istemci ve serverler hala baglantiyi kapatabilir.Connection:close headeri yok diye sonsuza kadar acik kalcak diye bir sey yok.

Kalici Baglanti Kisitlamalari ve Kurallari

Kalici baglantilari icin kullanilmasi tasvif edilen kisitlamalar:

Connection:close headerini istekte gonderdikten,istemci o baglantidan daha fazla istek yollayamaz.

Eger istemci baglantidan baska bir istek yapmak istemiyorsa,son isteginde Connection:close headerini yollamasi gerekir.

Baglanti sadece tum mesajlar dogruysa kalici kalabilir,mesajin body kisminin boyutu Content-Length headerinda dogru verilmesi gibi veya encode edilcek datanin encode halde olmasi.

HTTP/1.1 proxyleri kalici baglantilari istemci ve server icin ayri ayri yonetmeli,her kalici baglanti tek ulasim sekmesi icin gecerlidir.

HTTP/1.1 proxy serverlari HTTP/1.0 istemcisi ile kalici baglanti saglamamali(cunku eski proxyler Connection headerlarini geciriyor biliyorsunuz.)ancak istemcinin kapasitesini biliyor.Bu,pratikte,zor ve cogu satici bu kurallari cigniyor sonra noluo vulnerability cikiyor.

Connection headerlarinin degerlerini dikkate almadan,HTTP/1.1 cihazlari baglantiyi herhangi bir zamanda kapatabilir,saglam serverlar islemin ortasinda kapanmaya direnecektir ve her zaman en azindan bir istek oncesinde kapanmaya calisacaktir.

HTTP/1.1 uygulamalari senkronizasyon saglanmayan uygulamalardan kendisini kurtaracaktir.Istemciler istegi yeterince yan etki gostermeden tekrarlamaya calisacaktir.

Istemciler eger tum cevabi almadan once baglanti kapanirsa istegi tekrar yapmaya hazirlikli olmalilar,tabii istek hep tekrarlanirsa yukarda dedigimiz gibi yan etkileri olur.

Tek bir istemci kullanicisi herhangi bir server veya proxye en fazla iki tane kalici baglanti barindirabilir,server fazla yuklenmeden kurtarmak icin.Cunku proxyler server eszamanli kullanicilarla calisabilsin diye yardim etmesi icin daha fazla baglantiya ihtiyaci olabilir,bir proxy servera en fazla 2N kadar baglanti barindirmali ,eger N tane kullanici servera erismek istiyorsa.

Pipeline Baglantilari

HTTP/1.1 kalici baglanti uzerinden opsiyonel istek pipeline baglantisina izin verebilir.Bu keep-alive baglantisinin uzerinden daha iyi performans olayini taniyan bir sey.Birden fazla istek cevap ulasmadan once kuyruga girebilirler.Ilk istek network uzerinden servera dogru islerken dunyanin obur tarafinda,ikinci ve ucuncu istekler alttan alttan devam edebilir.Bu yuksek gecikmeli network durumlarinin performans artimina sebep olabilir,network tur gezilerini azaltarak tabi.

Simdi gelicek resimde a-c sikki kalici baglantini TCP baglanti gecikmesini nasil yok ettigini ve c sikkinda pipeline isteginin nasil transfer gecikmesini yok ettigini goruceksiniz.

Pipeline ile ilgili birkac kisitlama var:

1-HTTP istemcisi kalici baglanti olduguna emin olmadan pipeline olayini sunmaz.

2-HTTP cevaplari istekler gibi ayni sekilde donmeli.HTTP mesajlari sira numaralariyla etiketlenmemistir.Yani eger cevaplar siparisten farkliysa istekle cevabi eslestirip cevaba ulasmak gibi bir sey soz konusu degil.

3-HTTP istemcileri baglantinin herhangi bir zaman da kapanabilecegine ve pipeline'li istegin bitmeden yeniden baslatilmasi gibi bir seye hazirlikli olamli.Eger istemci kalici baglanti acarsa ve aniden 10 istek sunarsa,server 5 tanesini yerine getirdikten sonra baglantiyi kapatmakta serbest ve 5 tane istek fail olcak.Istemci bu erken kapanan baglantiyi yonetmeye calismali ve istegi tekrarlamali.

HTTP istemcisi eger istegin yan etkisi varsa (POST gibi).Genelde,hata oldugunda,pipeline istegi istemci tarafindan onlenir ve hangi pipeline isteginin server tarafindan calistirildigi

bilinir.Cunku tartisilmicak istekler POST gibi guvenli olarak tekrarlanmayabilir,hata durumlarinda asla calistirilmicak bir method calistirabilirsiniz.

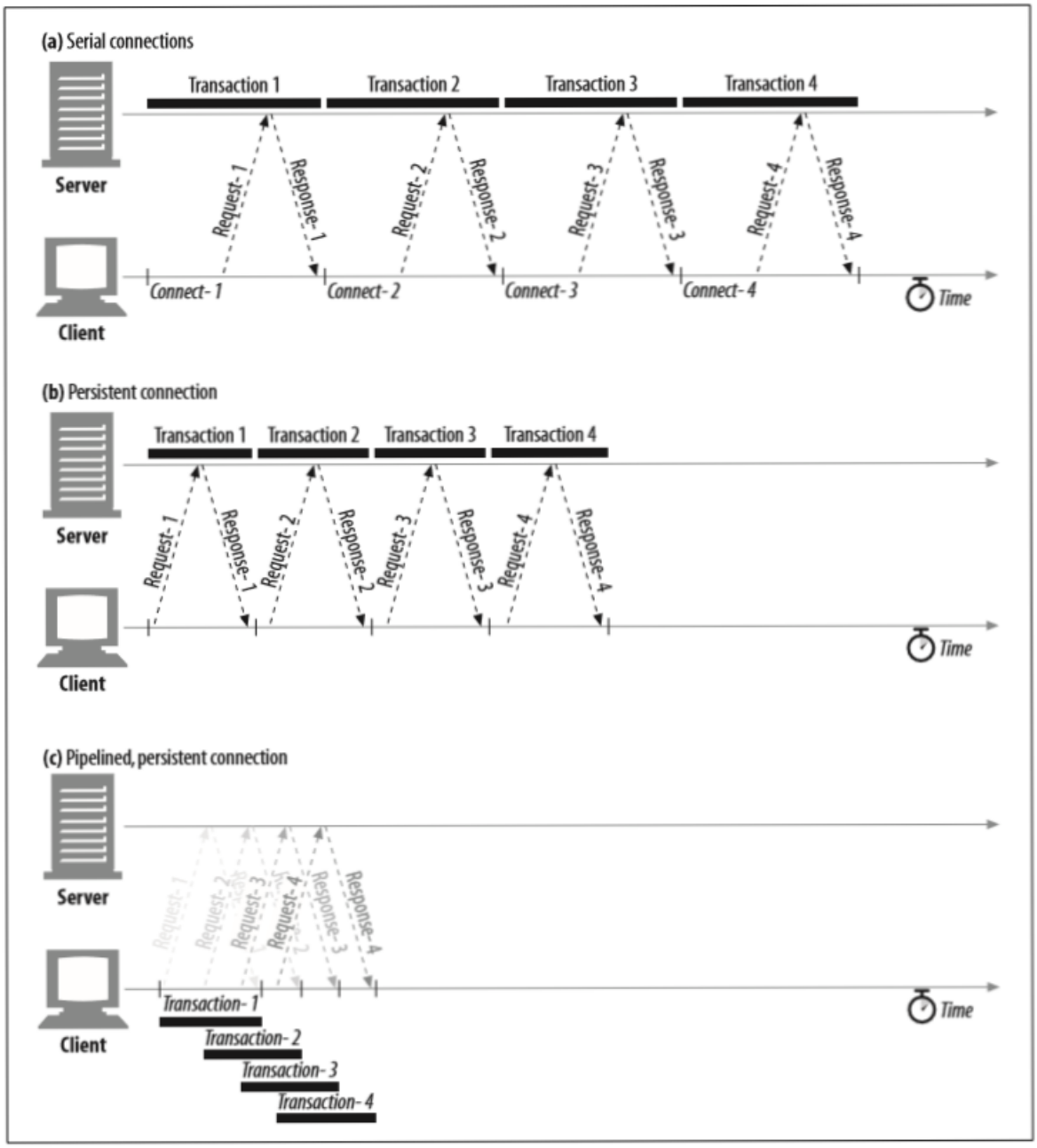

Figure 4-18. Four transactions (pipelined connections)

Gizemli bir sekilde kapanan baglantilar

Baglanti yonetimi-kisaca ne zaman ve nasil baglantinin kapanacagi-HTTP'nin en Pratik gerektiren kara sanatlaridir.Bu sorun cogu developerin fark ettiginden daha ince ve daha az uzerinde durulan bir sorun.

"OLACAK" KOPMASI (AT WILL NASIL ANLATAYIM OLACAK DEDIM :D)

Herhangi HTTP istemcisi,server,veya proxy TCP baglantisini zbam diye kapatabilir.Baglantilar normalde baglantiyi bitir mesajiyla biter,ama hata durumlarinda,baglanti header satiri okunurken veya daha degisik yerlerde kapanabilirdi.

Bu durum kalici pipeline baglantilarda siklasti.HTTP uygulamalari herhangi bir zamanda kalici baglantiyi kapatmakta serbest.Ornegin kalici baglanti isletildikten bir sure sonra,server bunu kapatmaya karar vericektir.

Ancak,server asla emin olucak sekilde islenen baglantiyi kapatirken istemcinin satirin sonuna gelindiginde ayni zamanda baska bir data yollamakta oldugunu bilemez.Eger bu olursa,isemci istek mesajinin ortasinda baglanti hatasini gorur.

CONTENT-LENGTH VE KESME

Her HTTP cevabi,cevabin body kisminin boyutunu belirtmesi icin hesaplayip Content-length headarina yazmasi gerekir.Bazi eski HTTP serverlari Content-length headerini veya hatali boyutu server baglantisininin gercek data boyutunu yazmadan baglantiyi kapatip yazmayi atliyor.Istemci veya proxy HTTP cevabini aldiktan sonra baglantiyi sonlandiriyor,ve gercek boyut Content-length ile uyusmuyor(veya Content-length yok),alici su soruyu sormasi lazim dogru boyut nerde.

Eger alici onbellekli proxy ise,alici cevabi onbellege almamali.Proxy tartisabilir mesaji dogru Content-Legnth mi diye dusunmeden anlamsal duzeni korumak icin mesaji gecirmeli.

BAGLANTI KAPANMA TOLERANSI,DENEMELER

Baglantilar herhangi bir zamanda kapanabilir,hata durumu olmasa dahi.HTTP uygulamalari beklenmdik kapanmalari yonetmekte onceden hazir olmak zorunda.Eger baglanti istemci islem yaparken kapanirsa,istemci baglantiyi bir kere daha acmayi denemeli,fakat islemlerin yan etkileride var tabi.Durum pipeline baglantilardan daha kotu.Istemci cok sayida istegi kuyruga koyabilir,ama ana server baglantiyi kapatabilir,sayisiz istek islenmemis olur ve yeniden planlamasina ihtiyac vardir.

Yan etkiler onemlidir.Birkac istek gonderilip cevap geldikten sonra baglanti kapandigi zaman,istemci serverin islemin ne kadarini yaptigindan %100 emin olamaz.Bazi islemler HTML sayfasini GET etmek gibi,tekrarlanir,tekrarlanir ve hic bisi degismeden gene tekrarlanir.Bazi islemler online kitap evine POST etmek gibi,tekrar edilmemeli,yoksa birden cok riske girersiniz.

Bir islem eger ne olursa olsun bir yada birden fazla calistirdiginizda ayni sonucu veriyorsa idempotenttir(Matristen gelsin kafaniza).GET,HEAD,PUT,DELETE,TRACE,OPTIONS methodlari bu ozelligi paylasiyor diye varsayilabilir.Mesela GET tabanli bir form her zaman duzgun calismali yani idempotent olmali.Istemci idempotent olmayan istekleri (POST gibi) pipeline etmemeli.Aksi takdirde,erken sonlandirmak baglantiyi hesaplanamaz sonuclara yol

acabilir.Eger idempotent olmayan istek yollamak istiyorsaniz,onceki istegin status kodunu beklemeniz gerek.

Idempotent olmayan methodlar veya siralar otomatik olarak tekrarlanmamali,Insanlarin kullandigi kullanici ajanlari(browser vb.) zaten istegin kullanici tarafindan tekrarlanmasini onerir.Ornegin,cogu tarayici bir diyalog formunun POST cevabi on bellege alindiginda post islemi tekrar edilmek istiyorsa bunu yeninden isticektir.

ZARIF BAGLANTI KAPANMALARI

TCP baglantilari cift yonludur,asagida gosterildigi gibi.Her TCP baglanti tarafinin input kuyrugu ve output kuyrugu vardir,datanin yazilmasi veya okunmasi icin.Data bir yerden output oldugunda diger daraftan input olur.

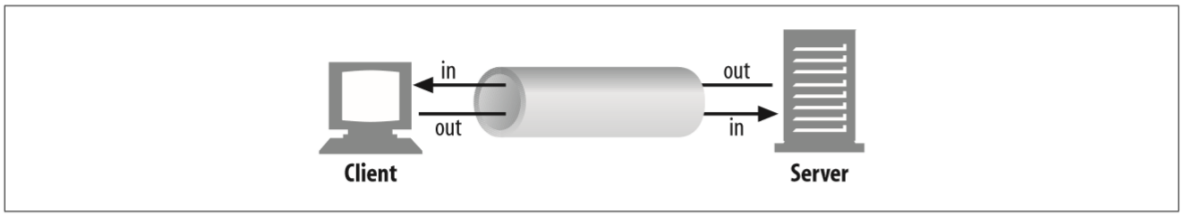

Figure 4-19. TCP connections are bidirectional

FULL VE YARI KAPANMALAR

Bir uygulamasi TCP kanalinda iki taraftan birinde yada iki taraftada kapanabilir.Soketleri gormustuk yukarda.Bir close() soketi TCP baglantisi icin input ve output iki tarafida kapatir.

Buna full kapanma denir.Resimde goreceniz sakin.shutdown() soketide kullanilabilir.input veya output kanalina bireysel olarak kapamanizi saglar.Buda half closedur.

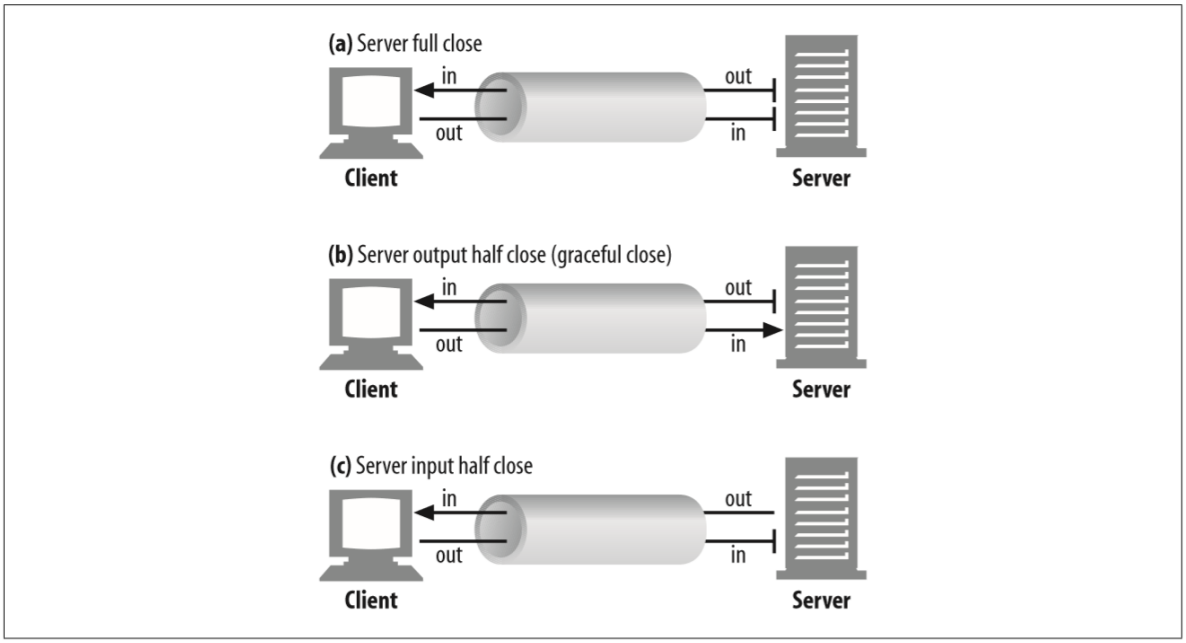

Figure 4-20. Full and half close

TCP kapanmasi ve reset hatalari

Basit HTTP uygulamalari sadece full close kullanabilir›Ama ne zaman uygulamalar diger tip HTTP istemcileri,serverlar ve proxyler gibi seylerle konusmaya basladi ve kalici baglantilarda pipeline kullanmaya basladi,half close onlar icin daha onemli olmaya basladi,cunku beklenmedik hatalar onlenecekti.

Genelde,output kanalini kapatmak her zaman guvenlidir.Baglantinin obur tarafindaki akranlari bildircektirki,kapadiginiz baglanti son akis alindiktan sonra tum data okundu.

Input kanalini kapamak daha riskli,tabi diger taraf daha fazla data gondermiyorsa ve bunu biliyorsaniz isler degisir.Eger diger taraf kapadiginiz input kanalinda data yolluyorsa,isletim sistemi "baglanti kanal tarafindan tekrarlanacaktir" mesajini diger makineye asagidaki resimdeki gibi yollayacaktir.Cogu isletim sistemi bunu ciddi bir hata olarak gorur ve diger tarafin okumadigi datalari bellekten siler.Bu pipeline baglantilar icin kotudur.

Diyeliki kalici baglanti uzerinden 10 tane pipeline istegi yaptiniz ve cevaplar size ulasti ve isletim sisteminizdeki bellekte duruyor(uygulama daha okumadi yalniz).Istek 11'I yolladiniz diyelim,ama server karar Verdi bu baglanti cok uzun suredir kullaniliyor ve kapadi.Isteginiz kapanmis baglanti uzerinden ulasacak ve size reset olarak geri donecektir.Bu reset input bellegini silecektir.

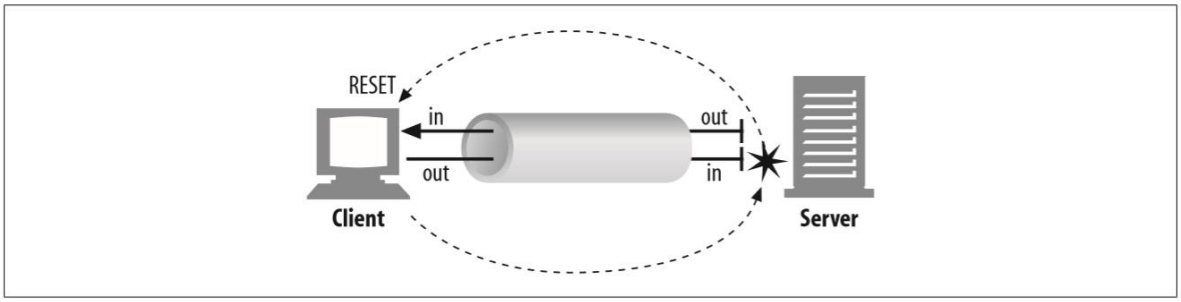

Figure 4-21. Data arriving at closed connection generates "connection reset by peer" error

Sonunda datayi okumayi basladiginda,baglanti reset islemini diger taraftan hata olarak alicaz,ve saklanmis belleklenmis,okunmamis cevaplar kaybolacaktir,makinenize en basarili sekilde ulasanlar bile.

ZARIF KAPANMALAR

HTTP uzerinde calisanlarin tavsiyeleri,istemci veya server baglantiyi beklenmedik bir sekilde kapamak istiyorsa,tavsiyeleri su"zarif bir sekilde baglantiyi kapasinlar" ama bunu nasil yapicaklarini tarif etmemisler.

Genelde,uygulamalar zarif kapamayi ilk output kanalinda uygulayacaklar ve sonra karsi taraftaki akranlarinin output kanalini kapatmasini bekleyecekler.Iki taraf birbirine daha fazla data yollamayacagini soyledigi zaman,baglanti full bir sekilde kapanacak ve reset riski yok.

Maalesefki,iki tarafin uygulayacagi veya half kapanma icin kontrol edileceginin garantisi yok.Bu nedenle,uygulamalar zarif bir sekilde output kanallarini kapatmalilar ve periyodik

bir sekilde input kanallarinin durumunu kontrol etmeliler.Eger input kanali karsi taraf tarafindan bazi zaman asimi periyotlariyla beraber kapatilmassa,uygulama baglantiyi kapamayi zorlayacak kaynaklari kurtarmak icin.