HTB Linux Kuyruklama Düzeni Kullanıcı Kılavuzu

Yazan:
Martin "devik" Devera
<devik (at) cdi.cz>

Yazan:
Don Cohen

Çeviren:
Nilgün Belma Bugüner
<nilgun (at) belgeler·gen·tr>

Ekim 2006

Özet

HTB (Hierarchical Token Buckets – Hiyerarşik Dizgecik Demetleri), Linux'taki CBQ kuyruklama düzeni algoritmasının yerini almak üzere tasarlanmış olup ondan daha hızlı, daha sezgisel ve daha kolay anlaşılır bir kuyruklama düzeni algoritmasıdır. Bu belge HTB algoritmasını nasıl kullanacağınızı gösterir. Çoğu bölümü örnekler, çizgeler (ölçülmüş verilerle) ve bellibaşlı sorunların incelemelerini içerir.

Konu Başlıkları

Geçmiş

1.0 İlk çeviri	Ekim 2006	NBB
? Özgün sürüm	5 Mayıs 2002	MD ve DC

Yasal Uyarı

1. Giriş

HTB (Hierarchical Token Buckets – Hiyerarşik Dizgecik Demetleri), Linux'taki CBQ kuyruklama düzeni algoritmasının yerini almak üzere tasarlanmış olup ondan daha hızlı, daha sezgisel ve daha kolay anlaşılır bir kuyruklama düzeni algoritmasıdır. CBQ ve HTB, her ikisi de belli bir bağlantı üzerinde çıkış band genişliği kullanımını denetim altında tutmaya yardımcı olurlar. Her ikisi de tek bir fiziksel bağlantıyı çok sayıda daha yavaş bağlantı şeklinde taklit ederler ve farklı trafik türlerini farklı taklit edilmiş bağlantılardan gönderirler. Her iki düzende de, fiziksel bağlantının nasıl taklit bağlantılara bölüneceğini ve hangi taklit bağlantı üzerinden hangi paketlerin gideceğine nasıl karar verileceğini belirtebilirsiniz.

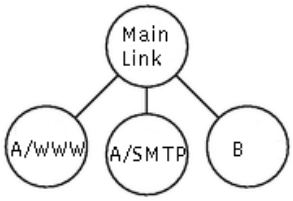
Bu belge HTB algoritmasını nasıl kullanacağınızı gösterir. Çoğu bölümü örnekler, çizgeler (ölçülmüş verilerle) ve bellibaşlı sorunların incelemelerini içerir.

HTB'nin bu sürümünün daha ölçeklenebilir olması lazım. Karşılaştırma için HTB ana sayfasına[B2] bakınız.

Lütfen okuyun:

tc aracı (sadece HTB değil), hız birimlerin belirtmek için kısaltmaları kullanır. **kbps**, kilo**bayt** anlamına ve **kbit** kilo**bit** anlamına gelir! Bu, Linux'ta **tc** ile ilgili olarak en sık sorulan sorudur.

2. Bağlantı Paylaşımı



Örnek 1. Sorunumuz:

A ve B diye iki müşterimiz olsun, her ikisi de ağa `eth0` üzerinden bağlanıyor olsun. B'ye 60 kbps, A'ya 40 kbps ayırmak istiyor olalım. Ayrıca, A'nın band genişliğinin 30 kbps'sini WWW için, 10 kbps'sini de kalan herşey için kullanılmak üzere bölmek istiyor olalım. Kullanılmayan band genişlikleri de ihtiyaç olduğunda sınıflar (alt bağlantılar) tarafından (paylarına göre) kullanılsın istiyoruz.

HTB, her sınıfa, o sınıfa atanmış olan miktarı, talep olduğunda, asgari miktar olarak garanti eder. Bir sınıf kendine adanmış miktardan daha azını isterse, kalan band genişliği hizmet isteyen diğer sınıflara dağıtılır.

Ayrıca, HTB teorisi[B3] ile ilgili belgeye de bakınız; yukarıdaki görevi daha ayrıntılı açıklar.

"borrowing" hakkında

Literatürde buna (band genişliğinde fazlalığın ihtiyacı olana verilmesi) "borrowing" (borçlanma/borçlandırma) deniyormuş. Bence "yardımlaşma" amaca daha uygun; çünkü, planlı bir geri ödeme durumu sözkonusu değil. Yazar da bunu belirtmiş ama bir terim önermeyip literatüre bağlı kalmış. Bizim literatür yeni oluştuğu için biz en uygun terimi seçebiliriz.

Trafiğin yukarıdaki farklı çeşitleri HTB'de sınıflarla ifade edilir. En basit yaklaşım resimle göstermektir. Gelin, kullanılacak komutlara bakalım:

```
tc qdisc add dev eth0 root handle 1: htb default 12
```

Bu komut, HTB kuyruklama düzenini `eth0` ile ilişkilendirir ve ona bir belirteç (`handle`), **`1:`** atar. Bu sadece gerektiğinde ona başvururken kullanılacak bir isimdir. **`default 12`** ise başka türlü sınıflanmamış her türlü trafiğin `1:12` sınıfına ait olacağını belirtir.

Bilgi

Genelde (sadece HTB için değil, **`tc`**'deki tüm kuyruklama düzenleri ve sınıfları için), belirteçler `x:y` biçiminde yazılırlar; burada `x` bir kuyruklama düzenini[1] tanımlayan bir tamsayı, `y` ise bu kuyruklama düzeninin üyesi olan bir sınıfı tanımlayan bir tamsayıdır. Kuyruklama düzeninin kendisini belirten belirtecin `y` değeri sıfır olmalı, bir sınıfı belirten belirtecin ki ise sıfırdan farklı bir sayı olmalıdır. Yukarıdaki **`1:`**, **`1:0`** olarak ele alınır.

```
tc class add dev eth0 parent 1: classid 1:1 htb rate 100kbps ceil 100kbps
tc class add dev eth0 parent 1:1 classid 1:10 htb rate 30kbps ceil 100kbps
tc class add dev eth0 parent 1:1 classid 1:11 htb rate 10kbps ceil 100kbps
tc class add dev eth0 parent 1:1 classid 1:12 htb rate 60kbps ceil 100kbps
```

İlk satır **`1:`** kuyruklama düzeni altında bir kök ("root") sınıf, **`1:1`** sınıfını oluşturur. Bir kök sınıfın tanımı, atası gibi tek başına bir HTB kuyruklama düzeni olmaktır. Bir kök sınıf, HTB kuyruklama düzeninin altındaki diğer sınıflar gibi kendi çocuklarının aralarında yardımlaşmasına izin verir, fakat bir kök sınıf kendi seviyesindeki bir başka sınıfla yardımlaşamaz. Biz diğer üç sınıfı doğrudan HTB düzeni altında oluşturmuştuk, ancak birinin band genişliğindeki bir fazlalığın bir diğerine yararı olmayacaktı. Halbuki biz yardımlaşma olsun istiyorduk. Bu durumda, kök sınıf olarak hizmet verecek ek bir sınıf oluşturup, gerçek veriyi taşıyacak sınıfları bunun altına koymalıydık. Bunlar son üç satırda tanımlanmışlardır. `ceil` parametresi aşağıda açıklanacaktır.

Bilgi

Bazan bana soruyorlar, "`dev eth0` neden sürekli tekrarlanıyor, zaten `handle` veya `parent` kullanılıyor." diye. Bunun sebebi belirteçlerin arabirime özgü olmaları, yani, eth0 ve eth1'in her ikisi de kendi 1:1 sınıfına sahip olabilir.

Ayrıca, hangi paketlerin hangi sınıflara ait olacaklarını da tanımlamalıyız. Bu gerçekte HTB düzeni ile ilgili değildir. Ayrıntılar için tc filter belgesine bakınız. Komutlar şöyle görünecektir:

```
tc filter add dev eth0 protocol ip parent 1:0 prio 1 u32 \
   match ip src 1.2.3.4 match ip dport 80 0xffff flowid 1:10
tc filter add dev eth0 protocol ip parent 1:0 prio 1 u32 \
   match ip src 1.2.3.4 flowid 1:11
```

(A'yı burada IP adresine göre tanımladık; IP adresinin 1.2.3.4 olduğunu varsaydık.)

Bilgi

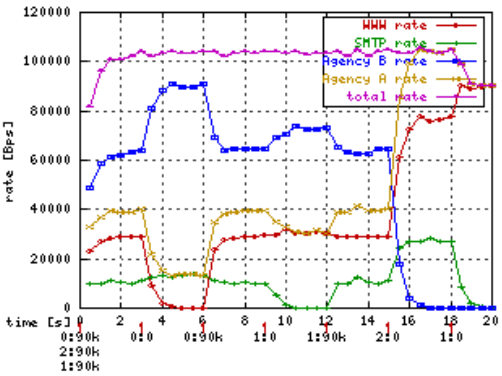
U32 sınıflayıcının belgelenmemiş bir tasarım hatası var; U32 sınıflayıcıları farklı `prio` (öncelik) değerleri ile kullandığınızda "**`tc filter show`**" yinelenmiş girdiler listelenmesine sebep oluyor.

`1:12` sınıfı için bir filtre oluşturmadığımızı söyleyebilirsiniz. Bir filtre oluştursaydık herşey belirginleşirdi belki ama o zaman `default` kullanımını gösteremezdik. Yukarıdaki iki kural tarafından sınıflanmamış her paket (kaynak adresi 1.2.3.4 olmayan her paket) `1:12` sınıfına konulacaktır.

Artık, kuyruklama düzenlerini istediğimiz gibi uç sınıflarla ilişkilendirebiliriz. Eğer hiçbirşey belirtilmemişse `pfifo` öntanımlıdır.

```
tc qdisc add dev eth0 parent 1:10 handle 20: pfifo limit 5
tc qdisc add dev eth0 parent 1:11 handle 30: pfifo limit 5
tc qdisc add dev eth0 parent 1:12 handle 40: sfq perturb 10
```

İhtiyaç duyduğumuz tüm komutlar bu kadar. Paketleri 90 kbps hızda her sınıfa gönderirsek ve her seferinde bir sınıfın paketlerini göndermeyi durdurursak, ne olur görelim. Çizgenin altında dizilmiş olan "0:90k" benzeri yaftaları görmüşsünüzdür. Yaftaların orta noktaları (9'un yanında, ayrıca kırmızı 1 ile imlenmiş olarak) sınıfın trafik hızının değiştiği zamanları işaret eder. Bu yaftalarda ikinokta iminin solunda kalan sayı sınıfı (1:10 sınıfı için 0, 1:11 sınıfı için 1, 1:12 sınıfı için 2), sağındaki sayı ise yaftanın işaret ettiği zamanda başlatılan trafiğin hızını gösterir. Örneğin, 0 sınıfının hızı 0. zaman noktasında 90k, 3. zaman noktasında 0k ve 6. zaman noktasında tekrar 90k yapılmaktadır.



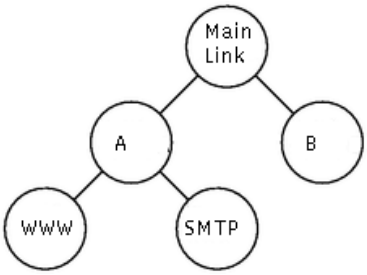
İlk olarak tüm sınıflar 90kb üretmektedir. Bu belirtilmiş hızların her birinden büyük olduğundan her sınıf kendi hızını ayarlandığı hızla sınırlar. 3. zaman noktasında 0 sınıfının paketlerini göndermeyi durdurduğumuzda, 0 sınıfı için ayrılan hız diğer iki sınıfa hız oranları nispetinde paylaştırılır; 1 sınıfına 1 pay, 2 sınıfına 6 pay. (1 sınıfındaki hız değişikliğini çizgede görmek zordur, çünkü hızı sadece 4kbps'tir.) Aynı şekilde, 9. zaman noktasında 1 sınıfının trafiği durdurulmakta ve onun band genişliği diğer iki sınıfa pay edilmektedir. (Bu kez de, 0 sınıfındaki artışı gözlemlemek zor olacaktır.) 15. zaman noktasında 2 sınıfının trafiği durdurulunca 0 ve 1 sınıflarındaki artışı gözlemlemek zor olmayacaktır. 18. zaman noktasında 1 ve 2 sınıflarının trafiği durunca, 0 sınıfı tüm trafiği (90kbps) almaktadır.

Katkıpayı'ndan bahsetmek için tam yeri. Aslında, sınıflar sahipsiz kalan band genişliğinden paylarına düşeni alırken çekiştikleri diğer sınıflara bir miktar bayt verirler. Buna katkıpayı (quantum) adı verilir. Eğer ata sınıfın band genişliği için çok sayıda sınıf çekişiyorsa, bunların paylarını katkıpayları nispetinde aldıklarını görürdünüz. İşlemlerinin titizce yürütülebilmesi için katkıpaylarının olabildiğince küçük ama MTU'dan büyük olması gerektiğini bilmek önemlidir.

HTB önceden hesaplanmış değerleri kendi seçtiği için normalde katkıpaylarını sizin belirtmeniz gerekmez. Sınıfın katkıpayı, hızının `r2q` genel parametresine bölünmesi ile bulunur (sınıf ekleme ve değişiklik sırasında). `r2q`'nun öntanımlı değeri 10'dur ve MTU genellikle 1500 olduğundan hız için bu değer 15 kBps (120kbit)'ten iyidir. Kuyruk düzeni oluşturulurken daha küçük asgari hızlar belirtmek için, yeterli olması gereken 12 kbit'ten iyidir (burada, özgün metinde bir belirsizlik var; sanki yazar cümlenin sonunu getirememiş ya da cümle kısmen biçim dönüşümüne kurban gitmiş gibi görünüyor). İsterseniz, sınıf eklerken veya değiştirirken katkıpayını elle kendiniz verebilirsiniz. Eğer önhesaplamalı değer kötü olursa günlük iletileri arasında uyarılara rastlayabilirsiniz. Katkıpayını komut satırında belirttiğinizde, `r2q` sınıf için yoksayılır.

A ve B farklı müşteriler değilse bu çözüm iyi gibi görünebilir. Bununla birlikte, A eğer 40kbps için bir ücret ödüyorsa, kullanmadığı WWW band genişliğinin kendi hizmetine değil de B'ye tahsis edilmesini muhtemelen tercih etmeyecektir. Bu gereksinim HTB'de sınıf hiyerarşisi tarafından karşılanır.

3. Paylaşım Hiyerarşisi



Örnek 2. Sorunumuz:

Önceki bölümdeki sorun bu resimdeki sınıf hiyerarşisine göre çözümlenecek. Müşteri A, şimdi doğrudan kendi sınıfı ile temsil edilecektir.

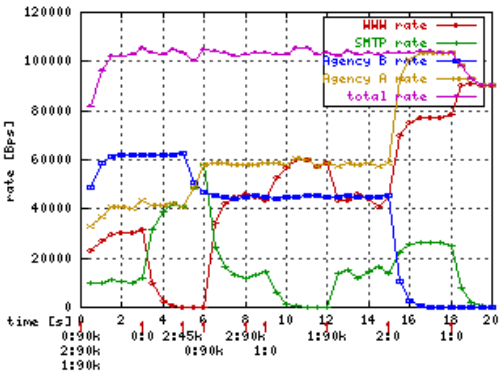
Tekrar, **HTB, her uç sınıfa, o sınıfa atanmış olan miktarı, talep olduğunda, asgari miktar olarak garanti eder**. Bu başka HTB sınıflarının atası olmayan HTB sınıflarına uygulanır. Bunlara uç sınıflar diyoruz. Başka HTB sınıflarının atası olan HTB sınıflarına ise iç sınıflar diyoruz ve bunlara **HTB, her iç sınıfa, o sınıfın çocuklarına atanmış olan toplam miktarı, talep olduğunda, asgari miktar olarak garanti eder** kuralını uyguluyoruz. Bu durumda, 40kbps'i A'ya atarsak, eğer A WWW için ayrılan hızdan azını isterse, artan miktar toplamda 40kbps'i sağlayacak şekilde A'nın başka bir trafiğine (eğer istek varsa) kullanılacaktır.

Bilgi

Paket sınıflandırma kuralları iç düğümlere de atanabilir. Bunun için iç düğümü başka bir filtre listesi ile ilişkilendirmelisiniz. Sonuçta, ya bir uç sınıf ya da özel bir 1:0 sınıfı elde edilmesi gerekir. Bir ata sınıf için sağlanan hız onun çocuklarının hızlarının toplamı olmalıdır.

Yeni komutlarımız şöyle olacaktır:

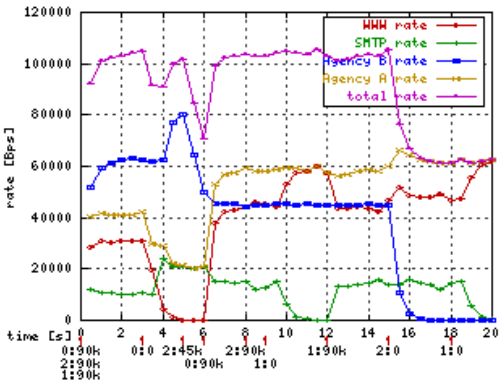
```
tc class add dev eth0 parent 1:  classid 1:1 htb rate 100kbps ceil 100kbps
tc class add dev eth0 parent 1:1 classid 1:2  htb rate 40kbps ceil 100kbps
tc class add dev eth0 parent 1:2 classid 1:10 htb rate 30kbps ceil 100kbps
tc class add dev eth0 parent 1:2 classid 1:11 htb rate 10kbps ceil 100kbps
tc class add dev eth0 parent 1:1 classid 1:12 htb rate 60kbps ceil 100kbps
```



Şimdi, hiyerarşik çözümün sonucunu gösteren çizgeye bakalım. A'nın WWW trafiği durduğunda, ona atanan band genişliği A'nın diğer trafiğine tahsis edilir, böylece A'nın toplam band genişliği hala 40kbps olarak kalır. Eğer A toplamda 40kbps'den azını talep ederse artan miktar B'ye tahsis edilir.

4. Hız Tavanı

`ceil` argümanı bir sınıfın kullanabileceği band genişliğinin azamisini belirtir. Bu, sınıfın yardım alarak ulaşabileceği band genişliğinin sınırıdır. Öntanımlı tavan aynı hız gibidir. Biz şimdi 1:2 sınıfları (A) için `ceil 100kbps` ve önceki bölümdeki 1:11 sınıfını (A'nın diğer sınıfı; WWW için olan) `ceil 60kbps` ve diğerini de `ceil 20kbps` yapalım.



Bu çizge önceki çizgeye göre 3. zaman noktasında farklıdır, çünkü A'nın diğer sınıfının hızı 20kbps ile sınırlanmıştır. Bu nedenle müşteri A toplamda sadece 20kbps almış olur ve kalan miktar B'ye tahsis edilir.

İkinci fark 15. zaman noktasında B durduğunda görülür. A için hız tavanı belirtilmemiş olsaydı, B'nin tüm band genişliği A'ya tahsis edilecekti, ama şimdi, A sadece 60kbps kullanabilmekte ve artan 40kbps kullanılmamaktadır.

Bu özellik ISS'ler için yararlıdır, çünkü bir müşterinin alabileceği hizmet miktarını diğer müşteriler hizmet almadıkları zaman bile sınırlamak isterler. (Müşterilerinin daha iyi hizmet için daha fazla para ödemelerini istediklerinden olsa gerek.) Kök sınıfların yardımlaşmaya katılmadıklarına, dolayısıyla, onlar için bir tavan belirtilmesinin gerçekte gerekmediğine dikkat ediniz.

Bilgi

Bir sınıf için tavan daima mümkün olan en yüksek hız olmalı, ayrıca daima en azından çocuklarının her birininin tavanı kadar yüksek olmalıdır.

5. Sıçrama

Ağ donanımı bir seferde sadece bir paket gönderebilir ve bunun da hızı donanımın hızına bağlıdır. Bağlantı paylaşım yazılımı çok sayıda bağlantıyı farklı (daha düşük) hızlarda çalıştırmak amacıyla yaklaşık olarak sadece bu imkanı kullanabilir. Bu nedenle, hız ve tavan gerçekte anlık olarak ölçülen değerler değillerdir, çok sayıda paketin gönderimi sırasında ölçülen değenlerin ortalamalarıdırlar. Gerçekte olan şudur: bir sınıftaki trafik bir kaç paketi bir defada azami hızda gönderdikten sonra diğer sınıflara geçilir. `burst` ve `cburst` parametreleri, başka bir sınıfın hizmet vermesini denemeksizin, azami hızda (donanım hızında) gönderilebilecek veri miktarını denetim altına alır.

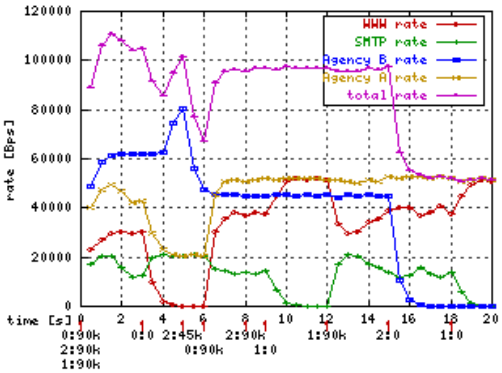
`cburst` yeterince küçükse (ideali bir paket boyudur), TBF'deki `peakrate`'in yaptığı gibi, `ceil` hızını aşmayan sıçramalar şekillendirir.

Ata sınıf için bir çocuğununkinden daha küçük `burst` belirttiğinizde, ata sınıfın bazan saplanıp kalacağını beklemelisiniz (çünkü çocuk, atasının elde edebileceğinden fazlasını tüketir). HTB bu negatif `burst` değerlerini 1 dakika süreyle hatırlayacaktır.

Neden sıçrama istiyormuşum diye sorabilirsiniz. Fazla dolu bir bağlantı üzerinde yanıt verme süresini iyileştirmenin ucuz ve basit bir yoludur da ondan. Örneğin www trafiği sıçramalı bir trafiktir. Sayfa istediğinizde isteğiniz bir sıçramaya sığar ve onu alır okursunuz. Boşta kaldığı sürede sıçrama tekrar "şarj" edilir.

Bilgi

Bir sınıfın `burst` ve `cburst`'ü daima en azından çocuğununki kadar yüksek olmalıdır.



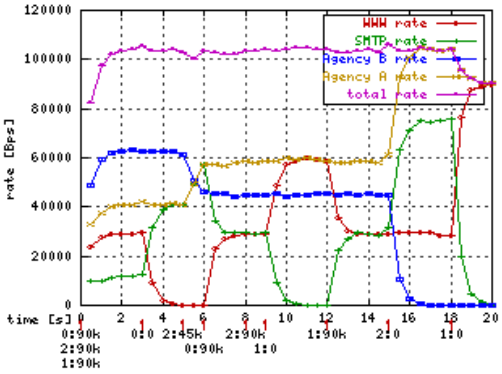
Bu çizgede bir önceki bölümde yer alan örnek durumda `burst`'ün kırmızı ve sarı sınıflar (agency A) için 20 kb yapıldığı durum görülmektedir, `cburst` aynı bırakılmıştır. SMTP sınıfının `burst` değerinden dolayı yeşil çizginin tepe yaptığı zaman noktası 13'tür. 9. zaman noktasından itibaren sınırın altında kalmış ve artarak 20 kb `burst`'e gelmiştir. Tepe noktasında 20 kb'e kadar çıkmıştır (burada tavan değeriyle sınıra ulaşmıştır çünkü `cburst` paket değerine yaklaşmıştır). Dikkatli okuyucular neden 7. zaman noktasında kırmızı ve sarı çizgilerin tepe yapmadığını merak etmiştir. Bunun nedeni sarı çizginin zaten tavan değerine ulaşmış olması ve daha fazla `burst` için yeri kalmamış olmasıdır. Burada en azından bir yerde istenmeyen bir etki oluşmuştur: Mor renkli çizgi 4. zaman noktasında çukur yapmıştır. Bunun nedeni de kök bağlantı (1:1) sınıfı için `burst` eklemeyi "unutmuş" olmamdır. Bu sınıf 1. zaman noktasında tepe yaptığını hatırlamış, 4. zaman noktasında mavi sınıf sarı sınıftan bant genişliği almak istediğinde onu reddetmiş ve kendini buna göre dengelemiştir.

Sınırlama:

Düşük çözünürlüklü bir zamanlayıcısı olan bir bilgisayar üzerinde yüksek hızlarla çalıştığınız zaman, tüm sınıflar için size biraz küçük tutulmuş `burst` ve `cburst` gerekir. Zamanlayıcı çözünürlüğü i386 sistemlerinde 10ms, Alfalarda ise 1 ms'dir. Küçükçe bir `burst` değeri `azami_hız*zamanlayıcı_çözünürlüğü` olarak hesaplanabilir. Dolayısıyla, düz i386'da 10Mbit için 12kb'lık bir `burst` değerine ihtiyacınız olur.

Çok küçük bir `burst` belirtirseniz, ayarladığınızdan daha düşük hızla karşılaşırsınız. En son **`tc`** sürümü, belirtmediğiniz takdirde olası en düşük `burst` değerini hesaplayacak ve atayacaktır.

6. Paylaşımda Öncelik



Trafiği öncelikli duruma getirme konusunun iki yüzü vardır. İlki band genişliğindeki fazlalığın kardeşler arasında nasıl dağıtılacağını etkiler. Şimdiye kadar gördüğümüz, band genişliğindeki fazlalığın hız oranlarına göre dağıtıldığı şeklindeydi. Şimdi, *Paylaşım Hiyerarşisi* (sayfa: 5) bölümündeki temel yapılandırmayı (tavan hız ve sıçrama olmaksızın hiyerarşi) kullanıp önceliği SMTP (yeşil) için 0'a (yüksek), diğerleri için 1'e değiştirdim.

Paylaşım bakımından öncelikli sınıfın artan band genişliğinin tamamını aldığını görürsünüz. Burada kural **artan band genişliği, önce daha yüksek öncelikli sınıflara ikram edilir** şeklindedir. Ancak garantili hız ve hız tavanı ile ilgili kurallar hala geçerlidir.

Sorunun bir de ikinci yüzü vardır: paketin toplam gecikmesi. Çok hızlı olan ethernette gecikme ölçümü görece zordur (gecikme ihmal edilebilir miktardadır). Fakat küçük bir yardımla bu sorun aşılabilir. Hız sınırı 100 kbps'in altında olan bir sınıfa sahip bir HTB'den başka bunun çocuğu olarak ikinci bir HTB ekleyebiliriz (ölçmeyi bu ikincisinde yapacağız). Böylece, daha büyük gecikmeli daha yavaş bağlantıyı taklit edebiliriz. Basit olması için iki sınıflı bir senaryo kullanacağım:

```
# gecikme bireştirici için kuyruklama düzeni
tc qdisc add dev eth0 root handle 100: htb
tc class add dev eth0 parent 100: classid 100:1 htb rate 90kbps

# gerçek ölçütlerde kuyruklama düzeni
tc qdisc add dev eth0 parent 100:1 handle 1: htb
AC="tc class add dev eth0 parent"
$AC 1:  classid 1:1 htb rate 100kbps
$AC 1:2 classid 1:10 htb rate 50kbps ceil 100kbps prio 1
$AC 1:2 classid 1:11 htb rate 50kbps ceil 100kbps prio 1
tc qdisc add dev eth0 parent 1:10 handle 20: pfifo limit 2
tc qdisc add dev eth0 parent 1:11 handle 21: pfifo limit 2
```

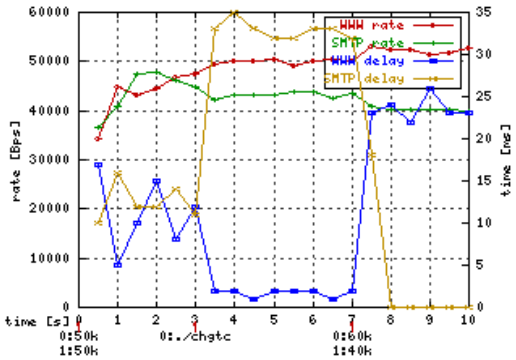
Bilgi

HTB, başka bir HTB'nin çocuğu olarak, aynı HTB'nin içindeki başka bir sınıfın altındaki sınıfla aynı DEĞİLdir. HTB içindeki sınıf gönderebildiğinde donanımın gönderebileceği kadar erken göndereceği için bu böyledir. Dolayısıyla, sınır altındaki sınıfın gecikmesi sadece donanım tarafından sınırlanır, ata sınıflar tarafından değil.

HTB altında HTB durumunda, dıştaki HTB herşeyiyle yeni donanımı taklit eder (daha büyük gecikmeyle).

Bireştirici her iki sınıf için 50 kbps üretmeye ayarlanır ve 3. zaman noktasında şu komutu çalıştırır:

```
tc class change dev eth0 parent 1:2 classid 1:10 htb \
 rate 50kbps ceil 100kbps burst 2k prio 0
```



Gördüğünüz gibi SMTP'nin gecikmesi artarken WWW sınıfının gecikmesi sıfıra yakın değerlere düşmektedir. Gecikmeyi iyileştirmek için önceliği arttırırsanız daima diğer sınıfın gecikmesinin kötüye gitmesine yol açarsınız.

Daha sonra (7. zaman noktasında) bireştirici WWW'yi 60 kbps hızda, SMTP'yi de 40 kbps hızda üretmeye başlar. Bundan sonraki ilginç davranışı gözlemleyebilirsiniz. Sınıf (WWW) sınırı aştığında HTB band genişliği için sınırın altında kalana öncelik verir.

Hangi sınıfa öncelik vermelisiniz? Genelde bunlar gerçekte düşük gecikme gerektiren yerlerdeki sınıflardır. Örnek, görsel veya işitsel trafik (trafiğin başka bir sınıf tarafından öldürülmemesi için burada gerçekten doğru hız kullanmanız gerekirdi) veya doğasında sıçramalar bulunan etkileşimli bir trafik (telnet, SSH gibi) olabilirdi ve diğer akışları olumsuz etkilemezdi.

Tamamı yararlı bağlantılarda bile hoş ping gecikmeleri almak için bilinen bir yöntem ICMP'ye öncelik vermektir (fakat bağlantılılığı ölçümlerken teknik açıdan istediğimiz bu değildi).

7. İstatistikleri Yorumlamak

tc aracı Linux'taki kuyruklama düzenlerinin istatistiklerini toplamayı mümkün kılar. Talihsizliğe bakın ki, istatistik sonuçları yazarlar tarafından açıklanmamıştır (Neyi açıklamışlar ki zaten; tc bir sır küpü) dolayısıyla çoğunlukla onları kullanamazsınız. Burada HTB istatistiklerini anlamanıza yardımcı olmaya çalışacağım. Önce HTB istatistiklerine bir bütün olarak bakalım. Aşağıdaki satırlar *Paylaşım Hiyerarşisi* (sayfa: 5) bölümündeki bireştirim sırasında alınmıştır.

```
# tc -s -d qdisc show dev eth0
 qdisc pfifo 22: limit 5p
 Sent 0 bytes 0 pkts (dropped 0, overlimits 0)

 qdisc pfifo 21: limit 5p
 Sent 2891500 bytes 5783 pkts (dropped 820, overlimits 0)

 qdisc pfifo 20: limit 5p
 Sent 1760000 bytes 3520 pkts (dropped 3320, overlimits 0)

 qdisc htb 1: r2q 10 default 1 direct_packets_stat 0
 Sent 4651500 bytes 9303 pkts (dropped 4140, overlimits 34251)
```

İlk üç düzen HTB'nin çocuklarıdır. PFIFO istatistikleri kendilerini zaten açıkladıkları için onları yoksayabiliriz.

`overlimits`, düzenin bir paketi kaç kere geciktirdiğini söyler.

`direct_packets_stat`, kaç paketin doğrudan kuyruktan gönderildiğini söyler. Diğer istatistikler kendi kendilerini zaten açıklıyor. Sınıfların istatistiklerine bakalım:

```
tc -s -d class show dev eth0
class htb 1:1 root prio 0 rate 800Kbit ceil 800Kbit burst 2Kb/8 mpu 0b
    cburst 2Kb/8 mpu 0b quantum 10240 level 3
 Sent 5914000 bytes 11828 pkts (dropped 0, overlimits 0)
 rate 70196bps 141pps
 lended: 6872 borrowed: 0 giants: 0

class htb 1:2 parent 1:1 prio 0 rate 320Kbit ceil 4000Kbit burst 2Kb/8 mpu 0b
    cburst 2Kb/8 mpu 0b quantum 4096 level 2
 Sent 5914000 bytes 11828 pkts (dropped 0, overlimits 0)
 rate 70196bps 141pps
 lended: 1017 borrowed: 6872 giants: 0
```

```
class htb 1:10 parent 1:2 leaf 20: prio 1 rate 224Kbit ceil 800Kbit burst ¬
2Kb/8 mpu 0b
    cburst 2Kb/8 mpu 0b quantum 2867 level 0
 Sent 2269000 bytes 4538 pkts (dropped 4400, overlimits 36358)
 rate 14635bps 29pps
 lended: 2939 borrowed: 1599 giants: 0
```

Çıktıyı kısaltmak için 1:11 ve 1:12 sınıflarını sildim. Gördüğünüz gibi bizim ayarladığımız parametreler var. Ayrıca `level` (seviye) ve DRR `quantum` (katkıpayı) bilgileri de var.

`overlimits`, sınıfın paketi kaç kere göndermek istediğini fakat hız/tavan sınırlamalarından dolayı yapamadığını gösterir (şu an sadece terkedilenler sayılmış).

`rate` ve `pps` sınıf üzerinden giden güncel hızı (10 saniyelik ortalama) söyler. Bu geçit tarafından kullanılan hızla aynıdır.

`lended`, bu sınıf tarafından (hızından) bağışlanmış paket sayısıdır. `borrowed` sınıfa atadan bağışlanan paket sayısıdır. Başka sınıflara bağışlananlar daima yardımlaşma geçişliyken sınıf için hesaplanır (1:10, 1:2'den bağış alırken 1:2 de 1:1'den bağış alır; 1:10 ve 1:2 bağış sayaçlarının ikisi de arttırımlıdır).

`giants`, **`tc`** komutunda ayarlı MTU'dan büyük olan paketlerin sayısıdır. HTB bunlarla çalışır fakat hızlar hepsinde doğru olmayacaktır. **`tc`**'nize MTU ekleyin (öntanımlı olarak 1600 bayttır).

8. Yamama, Hata Ayıklama ve Hata Raporu Gönderme

2.4.20 veya daha yeni bir çekirdeğe sahipseniz yamamaya ihtiyaç olmaz – herşey pakette mevcuttur. Tek ihtiyacınız **`tc`** aracı olacaktır. HTB–3.6 arşiv paketini indirip içindeki **`tc`**'yi kullanabilirsiniz (Ç.N. – bu çeviri yapılırken çekirdek 2.6.20'leri bulmuş ve standart **`tc`** HTB'yi zaten desteklemekteydi).

Daha eski çekirdeklerle çalışır hale getirmek için çekirdeği yamamalısınız. Çekirdeğin kaynak paketini indirdikten sonra **`patch -p1 -i htb3_2.X.X.diff`** komutuyla yamayı uygulayabilirsiniz. Sonra da evvelce yaptığınız gibi **`make menuconfig;make bzImage`** komutlarını kullanın. QoS ve HTB'yi etkinleştirmeyi unutmayın.

Ayrıca, yamanmış **`tc`** aracına da ihtiyacınız olacak. Bu yamayı ayrı olarak indirebileceğiniz gibi önceden derlenmiş ikiliği de indirebilirsiniz.

Eğer bir hata bulduğunuzu düşünüyorsanız hata raporunuza minnettar kalacağım. Oops'lar için **`ksymoops`** çıktısına ihtiyacım var. Tuhaf qdisc davranışı için **`tc qdisc add .... htb`** komutunuza **`debug 3333333`** parametresini ekleyin. bu parametre sayesinde syslog oluşumuna megabaytlarca günlük yazacaktır. Muhtemelen `/etc/syslog.conf` dosyanıza şöyle bir satır eklemek isteyeceksiniz:
`kern.debug -/var/log/debug`
Bunden sonra bu günlük çıktısını bzipleyip sorunu ve zamanını açıklayan bir eposta ekinde bana gönderin (bziplenmiş olarak 10MB'a kadar).

Notlar

Belge içinde dipnotlar ve dış bağlantılar varsa, bunlarla ilgili bilgiler bulundukları sayfanın sonunda dipnot olarak verilmeyip, hepsi toplu olarak burada listelenmiş olacaktır.

[B2] `http://luxik.cdi.cz/~devik/qos/htb/`

[B3] `http://luxik.cdi.cz/~devik/qos/htb/manual/theory.htm`

(1) Ç.N. – qdisc, Türkçeye "kuyruklama düzeni" olarak çevrilebilen "queue discipline" kısaltmasıdır; bir disk çeşidi değildir.

Bu dosya (htb-qdisc.pdf), belgenin XML biçiminin TEXLive ve belgeler-xsl paketlerindeki araçlar kullanılarak PDF biçimine dönüştürülmesiyle elde edilmiştir.

20 Ocak 2007