“BitTorrent och traffic shaping”

2G1305 Internetworking, KTH 2006-10-26
Examinator: Prof. Gerald Q. Maguire

Erik Carlborg Fröberg
André Eriksson
Polly Yeung

Sammanfattning

BitTorrent är det mest använda peer-to-peer (P2P-)protokollet och dess trafik står för minst 30% av den totala Internettrafiken i världen.[1] Protokollet fungerar på det sättet att det effektiviserar överföringen av stora filer över nätet genom att användare laddar upp så snart de har erhållit en liten del av filen. En användare som saknar just den biten kan då ladda ner den och således går det snabbare för alla att få de resterande delarna. ”Rare first”-algoritmen som tillämpas i BitTorrent-protokollet leder till att de flesta delar så småningom kommer att bli tillgängliga i nätverket.

BitTorrents egenskaper, hur geniala de än må vara, skapar problem för Internetleverantörerna eftersom ett stort antal uppkopplingar som använder mycket bandbredd upprättas mellan olika Internetleverantörers nät. Ett sätt att rätta till de problemen är genom ”traffic shaping”, vilket i praktiken oftast innebär en bandbreddsbegränsning för BitTorrent-trafiken. För att kunna implementera en sådan begränsning måste naturligtvis Internetleverantörerna identifiera vad som är BitTorrent-trafik och vad som inte är det. Detta kan göras genom att man inspekterar portar som normalt används för BitTorrent-trafik eller ”deep packet inspection”, d.v.s. att man undersöker paketen som skickas för att se om de är BitTorrent-paket. Dessa två tillvägagångssätt kan BitTorrent-klienter stoppa genom att använda många olika portar samt genom att införa krypteringsalgoritmer. Ett tredje sätt att särskilja BitTorrent-trafik vore att hitta en unik karakteristik som skiljer den från övrig trafik. Detta är svårare att tillämpa än de två tidigare nämnda tillvägagångssätten, men är också svårare för BitTorrent-klienterna att bygga runt.

Traffic shaping är dock inte särskilt populärt hos de kunder som vill använda BitTorrent, och därför bör man som Internetleverantör verkligen tänka sig för innan man implementerar det. Ett annat alternativ för att komma till rätta med BitTorrent är att använda sig av ”Cache Discovery Protocol”, d.v.s att man laddar ner de för tillfället populäraste filerna och hostar dem på Internetleverantörernas egen server. När en kund sedan vill ladda ner filen laddas den från Internetleverantören istället för en massa användare runt om i världen, vilket ger snabbare nedladdning för kunden och mindre belastning för Internetleverantören.

För dagens utbud av Internettrafik som består av en blandning av realtidsberoende trafik och traditionell trafik, behöver Internetleverantörerna använda olika mekanismer för att garantera en bra kvalité på näten. Genom att identifiera BitTorrent-trafik och placera just denna trafik i egna köer och dedikerade en specifik bandbredd för denna applikation kan övrig trafik fungera på ett tillfredställande sätt.

INNEHÅLLSFÖRTECKNING

1. Inledning

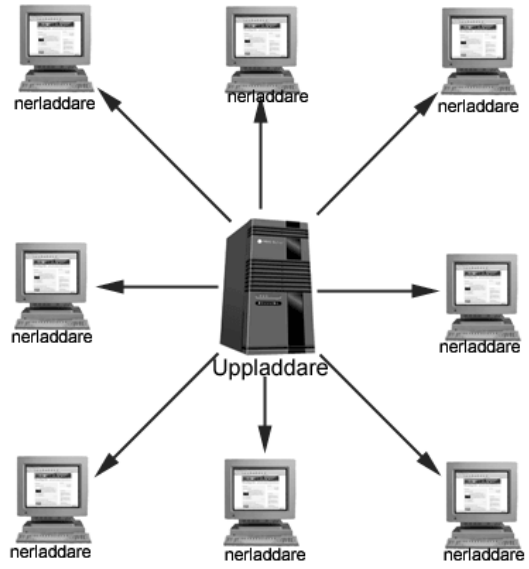
Internettrafiken ökar i rasande fart. År 2002 var den totala Internettrafiken cirka 100 000 gånger större än 1990.[2] Av den fantastiska mängden Internettrafik står P2P-protokollet BitTorrent för mer än en fjärdedel. Världens Internetleverantörer behöver mer och mer kapacitet för att kunna tillgodose kundernas behov. Kanske finns det dock annat att göra än att endast bygga ut kapaciteten?

Vi tyckte att det skulle vara intressant att fördjupa oss i ämnet just eftersom BitTorrent är ett så välkänt program som används vid distribuering av stora filer. Vi har i denna uppsats, inom ramen för kursen Internetworking 2G1305, undersökt hur BitTorrent och traffic shaping av BitTorrent-trafik fungerar. Vårt syfte med uppsatsen är att försöka fördjupa oss i frågor som: Hur kan man som utomstående observatör särskilja BitTorrent-trafik från annan Internettrafik? Hur kan man som utomstående förbättra eller försämra villkoren för BitTorrent-trafik och varför skulle man som Internetleverantör vilja göra det?

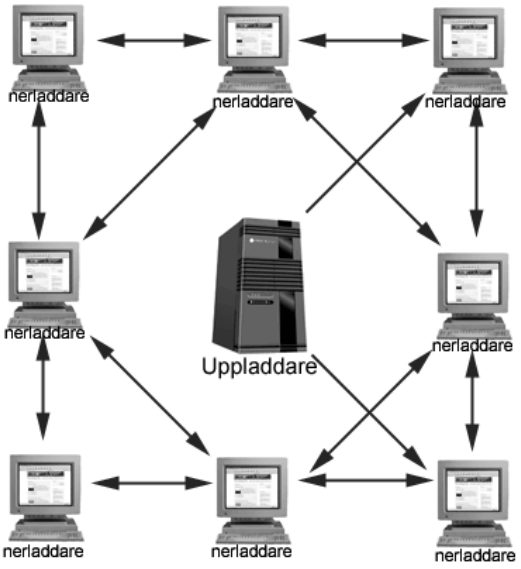
För att erhålla en praktisk förståelse för problematiken kring dessa frågor har vi utfört en rad tester. Testerna har gått ut på att simulera åtgärder som Internetleverantörer kan vidta för att påverka vilka effekter BitTorrent-trafik får på deras nät .

2. Introduktion till BitTorrent

BitTorrent är namnet på ett P2P-fildelningsprotokoll och den mjukvara som används för att implementera det. Båda designades och skapades av programmeraren Bram Cohen, som skrev programmet i Python. Hans syfte var att effektivisera överföring av data på nätet. Effektiviseringen ligger i att man genom redan under nedladdningen delar med sig av de delar man laddat ned till andra användare som ännu inte har dessa delar. Se nedanstående bilder på fildelning utan och med BitTorrent:



Figur 2.1 Fildelning utan BitTorrent



Figur 2.2 Fildelning med BitTorrent

Det protokollet egentligen gör är att bryta ner filen i mindre fragment, vanligtvis med en storlek på en kvarts megabyte (256 kb). BitTorrent är konstruerat så att det automatiskt väljer ut de användare som har bästa nätverksuppkopplingen utav alla som har de resterande fragmenten av filen. Vidare fungerar BitTorrent-klienterna så att de begär de filfragment som är tillgängliga på minst antal ”peers” vilket är samlingsnamnet för både ”seeders” (enbart uppladdare) och ”leechers” (nedladdare och uppladdare). ”Rare first”-algoritmen, d.v.s förfrågan efter de mest sällsynta delar, leder till att de flesta fragmenten kommer att bli tillgängliga på flera datorer. På detta sätt undviker man flaskhalsar. Filfragmenten laddas inte ned i sekventiell nivå eftersom man får en snabbare nedladdning när sekvens offras. Några nackdelar som finns är till exempel att man inte kan streama en film som laddas ned med BitTorrent.

När man väl har fått ner ett fragment används hashalgoritmen ”Secure Hash Algorithm**”** (SHA-1). I det här sammanhanget tillämpas den för att kontrollera att datan inte är korrupt. Man ”hashar” alltså varje filfragment, så snart den laddats ned i sin helhet, och jämför det med torrent-filen för att se om allt stämmer.[3]

Allt detta leder som sagt till att användarna, nedladdarna, normalt får en komplett fil mycket snabbare än vid nedladdning med ett P2P-fildelningsprogram som endast har en länk mellan en uppladdare och flera nedladdare. Dessutom så besparas en enda uppladdare från att bli överbelastad. Fortfarande krävs det dock att det åtminstone finns en seeder per torrent, d.v.s en användare som har den kompletta filen. Annars spelare det ingen roll hur många nedladdare som finns, eftersom ingen av dem har de delar som saknas i nätverket.

En annan faktor som gör att BitTorrent skiljer sig från vanliga fildelningsprogram är att det inte går att söka efter filer. Snarare är varje fil definierad genom sin egen torrent-fil. Denna fil är egentligen en liten pekare som innehåller följande metadata: filnamnet, storleken och varje filfragments hashkod (se Appendix B). En sådan fil öppnar man med en BitTorrent-klient, t.ex. Azereus, µTorrent som sedan startas upp som en ”seed node”, vilket låter andra användare att koppla upp sig och börja ladda ner. När mjukvaran väl är uppe och kör sker följande procedur automatiskt; torrent-filen pekar ut en BitTorrent-server, också känt som ”tracker” (se Appendix C), där man sedan får information om filen och eventuell användare. Den mängd användare som är intresserade av samma fil brukar betecknas som svärm och det är trackern, servern, som koordinerar kommunikationen mellan dessa peers.

På sistone har man lyckats göra ett tillägg till protokollet, en extra funktion som gör att man kan fortsätta dela ut filer som inte har någon aktiv server. Användarna frågar då varandra efter filen som identifieras genom ett matematiskt fingeravtryck (fingerprinting), ett hashvärde, vilket innebär att man kan spåra varandra.

När det gäller nedladdningshastigheten beror den givetvis på antalet seeders respektive antalet leechers d.v.s hur stor svärmen är, vilket bredband man har etc. När för många försöker att ladda ner en fil samtidigt kan detta även hämma nerladdningshastigheten eftersom servrar och nätförbindelserna blir överbelastade. Detta brukar hända med väldigt eftertraktade filer i den sekunden som de släpps. Notera dock att BitTorrent är gjort så att den ger bästa nedladdningskapacitet till de personer som laddar upp mest. Detta fenomen kallas för ”leech resistance” eftersom det motverkar att leechers försöker ladda ner filen utan att ladda upp någonting till någon annan.

2.1 De tekniska aspekterna av protokollet BitTorrent

En BitTorrent-fil består av en metainfo-fil (med filändelse .torrent) och en tracker-fil. [4]

Metainfo-filen är den fil som användaren först stiftar bekantskap med vid nedladdandet av torrent-filer och detta oftast via ett http-protokoll. Vanligast hittar man en torrent-fil i sin webbläsare och väljer att ladda ner den. Metainfo-filens karaktär beskrivs i Appendix B. När metafilen körs i en BitTorrent-klient ansluts nedladdaren till tracker-filen.

Tracker-filen är den del av BitTirrent-protokollet som kopplar samman peers. Trackerns karaktär och egenskaper beskrivs i Appendix C. Svaren från trackern är ”Bencoded dictionaries”, vilket är den typ av kodning som BitTorrent använder för att strukturera och skicka alla typer av sammanhängande data.[5]

En tracker response har antingen en nyckel, ”failure reason” eller två nycklar, ”interval” och peers. Failure reason pekar på en sträng som beskriver anledningen till felet. Interval pekar på antalet sekunder som nedladdaren ska vänta innan en förfrågan skickas igen. Nyckeln peers pekar på listan som innehåller nycklarna peer ID, IP-adress och port.

2.1.1 BitTorrents peer protokoll

BitTorrents ”peer protocol” opererar över TCP och fungerar effektivt utan att socketinställningar behöver göras. Peer-uppkopplingarna är symmetriska och meddelandena som skickas i båda riktningarna ser likadana ut och data kan överföras åt ena eller andra hållet. Så snart ett fragment laddats ner och matchar hashningen annonseras till nätverket att användaren har erhållit den just nedladdade delen.

Uppkopplingar innehåller två tillstånd, var och en representerad av en etta eller nolla, vid varje ände; ”choked” eller ”unchoked” samt ”interested” eller ”not interested”. Choking indikerar att inga data kommer att skickas förrän unchoking sker. Nödvändigheten med choking grundar sig på flera faktorer men en av dem är faktumet att ”TCP Congestion Control” inte fungerar effektivt när det skickas över flera uppkopplingar. ”TCP Congestion Control” har uppgiften att förhindra överbelastning på nätverket. Choking tillåter även användare att använda algoritmer för att försäkra sig om att hålla en konsekvent nedladdningshastighet.

En bra choking-algoritm ska kunna göra följande: reglera antalet simultana uppladdningar för att åstadkomma en god TCP-prestation, snabbt kunna undvika växelvis choking och unchoking vilket kallas för ”fibrillation” och det gör man genom att ändra vem som är choked var tionde sekund. Detta tidsintervall är tillräckligt för att TCP ska hinna sätta upp nya uppkopplingar och låta dem komma upp till deras maximala kapacitet. Sedan ska denna algoritm också samverka med de användare som låter en att ladda ner från dem och sist men inte minst ska den kunna då och då testa oanvända uppkopplingar för att se om det finns några som är bättre än de nuvarande – detta fenomen har fått namnet ”Optimistic unchoking”.[6]

Optimistic unchoking går ut på att unchoka en enda peer oberoende av dess uppladdningshastighet så länge denne är intresserad. Vilken peer som blir optimistiskt unchoked roteras var 30:e sekund. Detta tidsintervall är tillräckligt långt för att användaren ska hinna ladda upp en komplett del.[7]

Varje BitTorrent-användare unchokar automatiskt ett fixt antal andra användare, vanligtvis är denna siffra satt till 4, och beslutet att unchoka vilka användare beror helt på den nuvarande nedladdningshastigheten. Att beräkna den nuvarande nedladdningshastigheten är svårare än man tror och den nuvarande implementationen använder en rullande 20 sekunds period för varje beräkning. Tidigare använda choking-algoritmer utnyttjade informationen om långsiktig överföringsmängd på nätet. Det var dock väldigt ineffektiv eftersom värdet för bandbredden ändras alldeles för snabbt över tiden beroende på vilka källor som försvinner och vilka som blir tillgängliga. Det är givetvis bara de användare som har sämst uppladdningshastighet, d.v.s kommer med sämsta bidraget till ens nedladdningshastighet, som chokas. Om användaren har en komplett fil och agerar seeder, då används uppladdningshastigheten istället för nedladdningshastigheten för att avgöra vem som ska unchokas.

Uppkopplingar etableras när den ena änden är choked och den andra är not interested. Själva dataöverföringen sker dock inte förrän den ena sidan är interested och den andra sidan av uppkopplingen är unchoked. Detta är orsaken till att intressestatusen alltid bör vara uppdaterad. En nedladdare som för närvarande inte har något som den skulle vilja fråga efter från en annan peer i unchoked positionen måste visa brist på intresse även om denne är choked. Detta är svårt att implementera korrekt, men möjliggör för nedladdarna att se vilka peers som kommer att börja ladda ner direkt när de unchokas.

När data väl överförs bör nedladdarna ha flera ”piece requests” köat på en gång för att få en god TCP-prestation, även är känt som ”pipelining”. De requests som inte kan skrivas direkt till TCP-bufferten bör istället köas upp i minnet istället för i nätverksbufferten på applikationsnivån. Då kan de nämligen slängas när en choke inträffar.[8]

2.1.2 BitTorrents Peer wire

”Peer wire”-protokollet består av en handskakning som följs av en löpande och alternerande ström av meddelande.

Handskakningen börjar med talet 19, följd av strängen ”BitTorrent- protokoll” (se figur 2.4 på sid. 6). Efter dessa headers kommer 8 reserverade bytes som har värdet 0 i den nuvarande implementationen. Sedan kommer en 20 bytes SHA-1-hashning av den Bencodade formen av infovärdet från metainfo-filen. Om inte samma värde skickas från båda hållen kopplas uppkopplingen ner. Sist i handskakningen kommer en 20 bytes ”Peer ID” som rapporteras i tracker requests och finns tillgänglig i peer-listorna i tracker responses. Om peer ID:et på mottagandets sida inte stämmer överens med det som man på sändarsidan förväntat sig bryts uppkopplingen.

2.1.3 BitTorrent-meddelande

Som vi nämnde ovan finns det en rad möjliga meddelanden som kan skickas via BitTorrent-protokollet. Dessa är unchoke, choke, interested, not interested, have, bitfield, request, piece och cancel. Meddelandena börjar med en enda byte som anger dess typ:

- 0 - choke
- 1 - unchoke
- 2 - interested
- 3 - not interested
- 4 - have
- 5 - bitfield
- 6 - request
- 7 - piece
- 8 - cancel

Meddelandena choke, unchoked samt interested och not interested beskrevs ovan.

Have-meddelandet innehåller ett nummer som beskriver indexet för den del som just nedladdats och verifierats korrekt. Meddelandet bitfield skickas som första meddelande efter handskakningen. Denna del indikerar vilka delar som den nedladdande klienten har tagit emot och därmed kan dela ut själv.

Request har tre fält: ”index” för delen, ”begin” som betecknar vad inom delen som är begärt och ”length” som är delens längd. Piece-meddelandet är uppbyggt på samma sätt som request då dessa är kopplade till varandra. Slutligen finns cancel som generellt sett endast skickas i slutet av nedladdningen av en torrent-fil för att indikera att efterfrågat paket har mottagits.

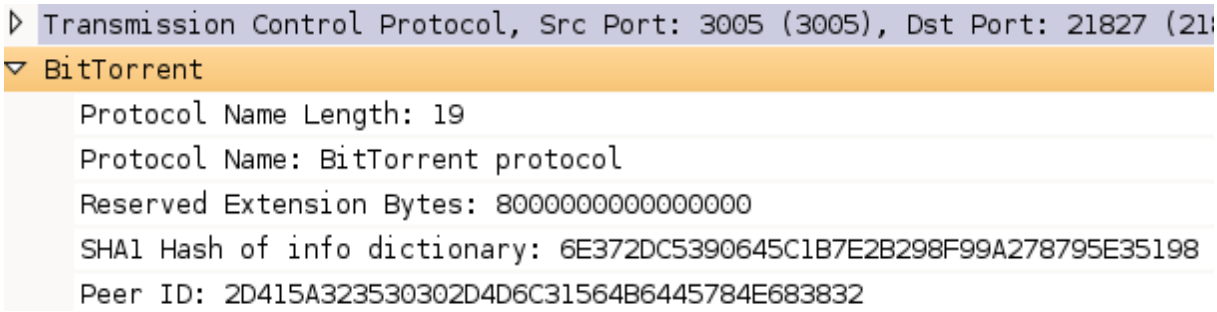
2.2 OSI-modellen och BitTorrent-protokollet

I figur 2.3 illustreras hur BitTorrent-protokollet förhåller sig till och använder sig av andra protokoll för överföring av data. Vi ser att BitTorrent är ett högnivå-applikationsprotokoll som använder TCP för transport av data. Vidare används UDP som transport protokoll för att söka upp peers till den aktuella torrenten.

BitTorrent		**Applikation**
TCP	**Src Port: 3005 Dst Port: 21827**	**Transport**
IP	**Src 192.168.0.3 Dst 60.51.89.41**	**Nätverk**
Ethernet II	**Src (00:0d:9d:5d:22:76) Dst (00:17:08:2f:dd:12)**	**Datalänk**
Frame 16	**(122 bytes on wire)**	**Fysisk**

Figur 2.3. Exempel på ett handskakningspaket tillhörande BitTorrent trafik fångat med hjälp av Ethereal.

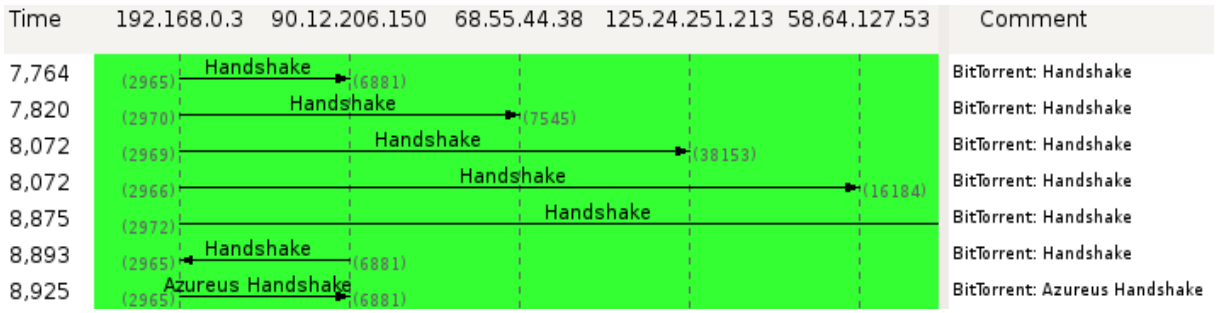
Om man tittar närmare på BitTorrent-protokollet i paketet ovan syns att Ethereal fångar upp bland annat SHA1 hash-koden och Peer ID. Peer ID:et innehåller information om bland annat vilken typ av klient som användaren kör sina torrent-nedladdningar i.[9]



Figur 2.4. Exempel på BitTorrent-protokollets detaljer

2.3 Nedladdning – det typiska förloppet

För att bekanta oss med BitTorrent som protokoll undersökte vi de meddelande som togs emot vid en nedladdning. När en torrent-fil börjar laddas ner observeras först ett 10-tal paket av typen handshake kopplade till klienterna. Därefter ser vi att det kommer en rad paket av typen bitfield. Näst i tur är typiskt paket av typen interested, unchoke och request piece. Dessa paket kommer det inte lika många av generellt som de tidigare nämnda. Därefter sätter själva dataöverföringen igång.



Figur 2.5 Illustrerar ett tidigt intervall i nedladdningsprocessen med BitTorrent. Vi ser att x-axeln betecknar den tidpunkt då handskakning sker mellan IP-adresserna i bildens överkant. Numren associerade med varje pil står för porten som kommunikationen sker över.

Det kännetecknas av att en stor mängd paket anländer av typen ”continuation data”. Kontinuerligt under nedladdningstiden förekommer en hel del paket av typen unchoke, choke och request piece. Mot slutet av nedladdningen tas paket av typen not interrested, have piece och cancel piece. Man kan konstatera att continuation data är den typ av paket som förekommer mest, vilket är naturligt då detta är kopplat till den tjänst som BitTorrent utför (leverera en viss fil) och de andra typerna mer eller mindre är overhead för att BitTorrent ska kunna leverera filen.

2.4 Den nya utvecklingen

BitTorrent-protokollet utvecklas för fullt hela tiden och detta brukar leda till förbättrad effektivitet och säkerhet. Maj 2005 lanserade Bram Cohen en betaversion av sitt BitTorrent-program som eliminerade behovet att ha en webbsida som en central värd för trackers. En så kallad tracker-fri klient vars DHT-implementering tillåter klienten att ladda ner torrents som skapats utan att en BitTorrent-tracker använts. Det finns även en aktuell funktion som kombinerar RSS och BitTorrent för att skapa ett innehållsleveranssystem kallat ”broadcatching”. Multitrackers är ännu en ny funktion som tillåter flera trackers per torrent-fil. Om en tracker går ner kan andra ta vid och fortsätta stödja filöverföringen. Senaste nytt är kryptering av IP-paketen i BitTorrent som några BitTorrent- klienter redan implementerat, även om det inte är helt klart hur effektiv denna kryptering egentligen är.

2.5 Laglighetsaspekterna

Än så länge är BitTorrent det enda P2P-protokollet som officiellt används för laglig distribuering vid t.ex. stora uppdateringar för spel, förhandsvisningar för filmer, distribuering av egna album och musik. Senaste nytt om ”laglig” användning av BitTorrent kommer från Warner Brothers Entertainment Studio som i maj annonserade att de kommer att använda detta protokoll för att sälja respektive hyra ut sina filmer och TV serier över Internet. De kommer att bli det första stora förlaget som provar på detta koncept. Det kommer givetvis att kosta att ladda ner dessa produkter, men i nuläget satsar man på att det ska kosta en dollar per episod medan en film ska kosta lika mycket som en vanlig DVD.

Om en person eller sida delar ut en fil som är upphovsrättsskyddad utan att ha upphovsmannens tillåtelse, begås ett brott. Använder man BitTorrent är ju dock själva torrent-filen i sig inte det upphovsrättsskyddade materialet, utan endast en länk till det. Polisen i Sverige genomförde för en tid sedan en kritiserad razzia mot The Pirate Bays (thepiratebay.org) tillhåll i Stockholm, på grund av att de hostade just torrent-filer. Många mer eller mindre konspiratoriska teorier förekommer om varför razzian genomfördes, exempelvis påstås den amerikanska organisationen Motion Picture Association of America ha påverkat den svenska regeringen att agera mot The Pirate Bay.[10] Vi får låta framtiden utvisa huruvida hostandet av en länk (torrent-fil) till upphovsrättsskyddat material strider mot upphovsrättslagstiftningen.

2.6 Klienter och andra applikationer

Tack vare den öppna källkoden för BitTorrent-protokollet har en stor mängd olika klienter skapats för att operera på olika plattformar. Förutom det enkla BitTorrent-programmet Mainline, som Bram Cohen själv skrev till sitt protokoll, finns följande populära klienter:

- **Azureus**: skrivet i Java och stödjer pluginarkitektur.

- **BitComet**: skrivet i C++ och är därför mer effektivt än många av Python- och Java-skrivna klienter resursmässigt.
- **BitTornado**: Ett av de första alternativen till originalet BitTorrent och är fortfarande ganska populärt.
- **µTorrent:** också skriven i C++ av en väldigt liten exekverbar storlek, men har ändå många av de funktionerna som finns i andra klienter.

3. BitTorrent och Traffic shaping

Fler och fler BitTorrentanvändare har börjat märka att några Internetleverantörer, som Rogers i Canada försöker, hämma all torrent-trafik genom att tillämpa ”traffic shaping”. Två av dagens mest populära BitTorrent-klienter, Azureus och µTorrent, jobbar dock på att komma runt detta problem genom att implementera RC4-kryptering för att motverka traffic shaping. Det som i första hand krypteras är IP headern på alla paket för att dölja faktumet att de skickas med BitTorrent.[11]

Hur effektiv krypteringen är vet man dock inte än. Bram Cohen, skaparen till BitTorrent, såg negativt på denna utveckling och var skeptisk på huruvida krypteringen är nödvändig, eftersom det trots allt är ett ganska litet antal Internetleverantörer som aktivt försöker motverka BitTorrent-trafik. Ändå så tillämpade även han kryptering i sitt BitTorrent-program i en senare version.

3.1 Tekniken bakom Traffic shaping

”Traffic shaping” är ett begrepp för den mekanism som möjliggör kontroll av mängden trafik, som skickats in i ett nätverk respektive den takt med vilken trafiken sänds ut i.[12]

Teorin säger att traffic shaping, när trafik flyter genom en logisk eller fysisk flaskhals, ska leda till: kortare fördröjning, färre förlorade paket samt mindre abrupta och oväntade förändringar i en eller flera egenskaper hos signalerna. Detta är intressant på så sätt att det tillåter att kontrollera vilka noder i ett nätverk som skall ”se” bra respektive dåliga ut. Som vi nämnde ovan är detta något som kan användas av Internetleverantörerna för att exempelvis begränsa P2P-trafik som den via BitTorrent och andra fildelningsapplikationer. En logisk anledning till detta kan vara att P2P-trafiken upptar mer och mer av Internetleverantörernas kapacitet och man kan därför vilja begränsa denna till förmån för annan trafik som surfa på webben eller spela onlinespel.[13]

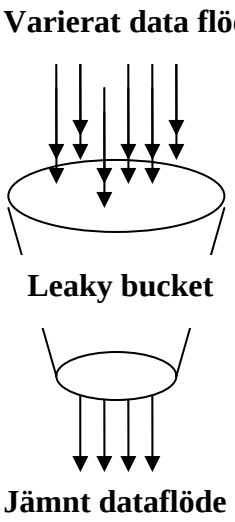
3.2 Flödeskontrollerande algoritmer

Det finns två huvudsakliga algoritmer som används för kontroll av dataflöden, d.v.s traffic shaping, ”Leaky bucket” och ”Token bucket”. Syftet med de båda metoderna är att utjämna ”briserande” trafikmönster till stadiga och jämna flöden. Det innebär att den trafikkontrollerande maskinen behöver köa upp en stor mängd trafik för att kontinuerligt släppa ut datan i en bestämd takt.

Ofta uppstår ”briserande” dataflöden då trafik passerar genom en flaskhals i nätverk. Genom att införa trafikformande maskiner i nätverk minskar de negativa effekter som ”briserande” datatrafik ger upphov till. Det gäller bland annat minskat antal förlorade paket, mindre oönskade variationer i signalens karakteristik och mindre fördröjningen mellan att ett paket börjat skickas till dess att det börjat tas emot.

3.2.1 Leaky bucket

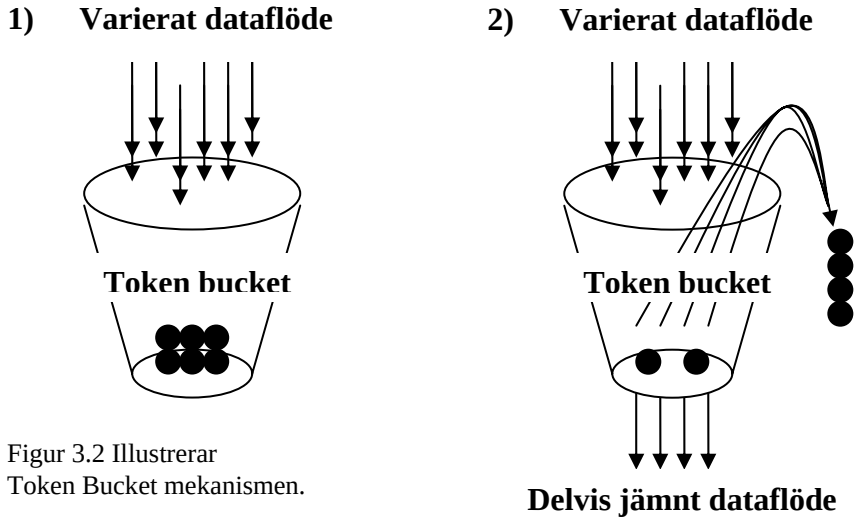
Leaky bucket är den metod som använder en så kallad hård gräns för maxtrafiken ut ur den trafikkontrollerande enheten. Det är särskilt användbart för att buffra UDP-trafik som anländer i en ojämn takt för att sedan mata ut den i en jämn takt. Fördelen med denna algoritm är att trafikmängder som är större än kapaciteten kan kontrolleras så att de flyter på ett smidigt sätt. Samtidigt är nackdelen dock att trafiken måste köas upp i den maskin som utför algoritmen och om väldiga mängder trafik kommer på samma gång kan ”hinken” fyllas, d.v.s kön blir full.



Figur 3.1 Illustrerar Token Bucket mekanismen.

3.2.2 Token bucket

Token bucket är en metod som liknar Leaky bucket men med den skillnaden att den tillåter en viss ojämn utsändnings-takt. Det innebär att en implementation av Token bucket släpper fram mindre mängder av ”briserande” datatrafik. Trafik tillåts sändas ut i paritet med hur många tokens som finns i hinken. För ett visst tidsintervall addera ytterligare en token till hinken. För varje gång ny trafik anländer tas tokens bort i förhållande till den datamängd som anländer. Det gör över tid att man får en gräns på utsändningen som är lika med medelvärdet på den inkommande trafiken.



Figur 3.2 Illustrerar Token Bucket mekanismen.

I den mening BitTorrent ger upphov till att stora mängder paket anländer i plötsliga anstormningar till routrar och därmed fördärvar just det nätets kvalitet. En lösning för Internetleverantörerna kan vara att tillämpa exempelvis algoritmen Token bucket för att jämna ut BitTorrent-dataflödena och få en bättre genomströmning i nätverket som helhet.

3.3 Quality of service

”Quality of service” (QoS) är ett begrepp inom telekommunikation för sannolikheten att ett givet nätverk möter den prestanda som slutits i kontrakt med olika användare.[14] Internetleverantörer berörs av detta eftersom de har en begränsad mängd bandbredd att erbjuda sina kunder. De får därmed lösa frågan hur de lämpligast bör fördela bandbredden så att sannolikheten blir som störst att de kan leverera all trafik i deras nät inom en önskad

latensperiod. Det kommer idag allt fler typer av tjänster i datanätverken som kräver en liten mängd jitter och fördröjningar för att de ska upplevas som justa att använda av kunderna. Exempel är IP-telefoni (VoIP), strömmande multimedia och videokonferenser. Dessa tjänster skulle snart förlora sin mening om Internetleverantören inte kunde garantera QoS.

3.3.1 Quality of Service mekanismer

Ett sätt för Internetleverantörer att uppnå QoS är att erbjuda kunderna flera storleksordningar mindre bandbredd än vad Internetleverantören har i sina nät mellan kunderna. På så sätt är man säker på att man har överkompenserat bandbredden och garanterat QoS. Voice over IP (VoIP) är dock en sådan tjänst som med denna metod och under hög belastning hos Internetleverantören skulle drabbas med dålig kvalitet på samtalen.

För att bemöta detta behöver man titta på paketen som strömmar genom nätverket för att reservera plats för paket. En tidig metod var IntServ. Denna metod innebar att stora routers i systemet behövde ta emot och acceptera gigantiska mängder reservationerna av bandbredd, något som skulle skifta deras fokus från att skicka paket.

Här kommer tillvägagångssättet ”DiffServ”, differentierade tjänster in. I denna modell är paketen markerade efter vilken typ av tjänst de behöver. Det innebär rent konkret att alla paket behandlas olika beroende på vad har för behov. Paketen som strömmar in i en router delas upp i olika köer som behandlas lite olika beroende på vilken implementation som används. De olika köerna tilldelas ofta en viss max-bandbredd. Denna övergripande mekanik innebär exempelvis att problemet med fördröjd VoIP kan minskas då dessa paket kan prioriteras annan köande trafik. På detta sätt är QoS mekanismer betydande för att hantera den relativt nya kombinationen av realtids- och icke realtids-trafik som strömmar över Internet.

En annan metod att förbättra genomflödet i nätverk är att använda sig av klassbaserade köer (CBQ). CBQ implementerar prioritetsköer i nätverket. En router identifierar vilken typ av paket som anländer genom att titta på paketens IP-header. Utifrån typ av paket delas trafiken upp i klasser. Dessa klasser tilldelas en specifik del av systemets totala bandbredd. Det är vanligast att denna algoritm implementeras vid nätverkets “entry points” d.v.s. in till ett specifikt nätverk. Det hela liknar i mångt och mycket vanlig dynamisk bandbreddstilldelning. CBQ fungerar på samma sätt som CIR (committed information rate) frame relay men CBQ verkar på OSI-layer 3, nätverksnivån.[15] Detta är något som skulle kunna användas av Internetleverantörer om BitTorrent-trafik identifieras och tilldelas en egen kö samt en dedikerad bandbredd. Internetleverantören kan vidare använda ”rate control” genom att manipulerar transmissions-hastigheten för paket. Med denna metod ändras acknowledge-fönstrets storlek i TCP-protokollet. Om tjänsten är av en typ som använder sig av UDP (som BitTorrent för uppsökning av peers) tillämpas en algoritm som Leaky bucket med fördel.

4. De ekonomiska aspekterna

Precis som alla andra företag strävar Internetleverantörerna efter lönsamhet. Vad inom övervakning och traffic shaping kan då tänkas vara lönsamt för en Internetleverantör och varför? Vi har listat några punkter som vi anser viktiga.

- Begränsa ”storkunders” nedladdning

Det vanligaste avtalet för bredband idag är s.k. flat rate, d.v.s. att man som kund betalar en i förväg bestämd summa pengar, oavsett hur mycket man använder sin Internetuppkoppling.

För Internetleverantören påverkas alltså intäkterna av antal abbonenter, inte totalanvändningen av nätet. Kostnaderna däremot, ökar naturligtvis med kundernas nätanvändande eftersom mer nätkapacitet krävs för att klara av all trafik. Den bästa kunden för en bredbandsleverantör är alltså den som betalar för en dyr och snabb lina, månad efter månad, men aldrig använder den. En kund som laddar ner sina 10Mbit/s dygnet runt är inte lika lönsam.

Det skulle alltså kunna löna sig för en Internetleverantör att försämra hastigheten för de typer av trafik som de ”tunga” kunderna utnyttjar mycket, exempelvis BitTorrent. Även om det skulle innebära att en del kunder bytte leverantör skulle det ändå kunna vara kortsiktigt lönsamt, i och med att man då frigör mycket nätkapacitet.

Detta är dock en mycket riskabel taktik. Dels kan detta förfarande ge upphov till avtalstvister, men framförallt kan det urholka kundbasens förtroende på lång sikt och därmed sänka varumärkets värde. Ett dåligt rykte sprider sig väldigt snabbt på Internet. Exempelvis har Azureus, en populär BitTorrent-klient, svartlistat de operatörer som på detta, eller andra sätt, skapar problem för BitTorrent-användare.[16] Dessutom är det oklart om det verkligen är så att en liten del Internetanvändare står för en stor del av P2P-trafiken. Företaget CacheLogic som utvecklar CDP och därmed visserligen är partiska i fallet, hävdar motsatsen, att P2P-trafiken är relativt jämnt fördelad mellan Internetanvändare.[17] Vi anser inte att man som Internetleverantör bör begränsa någon speciell form av trafik, utom möjligtvis om man tydligt klargör att så är fallet innan avtal tecknas med en kund.

- Cache Discovery Protocol (CDP)

Internetleverantörers intentioner att identifiera BitTorrent-trafik behöver inte vara ”onda”. CacheLogic och BitTorrent har tillsammans utvecklat ett protokoll kallat Cache Discovery Protocol.[18] Detta protokoll låter Internetleverantörer att hitta de populäraste torrentfilerna och cache:a dem. När någon av deras användare sedan vill ladda ned filen laddas den ned från Internetlverantören istället från diverse peers. Detta avlastar Internetleverantörerna kapacitetsmässigt samt besparar dem kostnader, eftersom de slipper betala andra Internetleverantörer i högre hierarkisk ordning för att nå alla peers, som kan befinna sig på andra sidan jordklotet. Denna nya teknik gynnar även kunden, eftersom han kan förvänta sig snabbare nedladdning. Dock lär detta nya protokoll endast användas för kommersiellt licensierade produkter, den stora mängden s.k. piratkopierat material lär inte hostas av någon seriös Internetleverantör så länge nuvarande lagstiftning råder i Sverige.

- Mutade av antipirat-lobbyn.

Vi känner inte till något sådant fall, men man kan tänka sig att starka lobbyorganisationer som Svenska Antipiratbyrån eller Motion Picture Association of America kan komma att försöka påverka Internetleverantörer att dela ut deras kundloggar till dem, för att de ska kunna sätta dit fildelare och andra ”pirater”. Detta ska man som Internetleverantör naturligtvis inte acceptera. Förutom att det mycket väl skulle kunna strida mot lagen, så skulle en sådan utlämning innebära ett svek gentemot kunden som garanterat inte kommer att tala väl om en i fortsättningen. Som Internetleverantör ska man logga det som enligt lag ska loggas, varken mer eller mindre, och endast dela ut dessa loggar till polisen, när de har rätt att begära den.

5. BitTorrent och kryptering

För att motverka Internetleverantörernas försök till traffic shaping av BitTorrent-trafik har flera BitTorrent-klienter implementerat kryptering. ”Protocol encryption” (PE), ”Message stream encryption” (MSE) och ”Protocol header encryption” (PHE) är det nuvarande resultatet av denna strävan.

PHE implementerades i de äldre versionerna av BitComet, men höll inte måttet eftersom bara en del av strömmen var krypterad vilket gjorde att man fortfarande kunde urskönja BitTorrent trafiken. Protokollet MSE utvecklades av Azureus i slutet av januari 2006, men den första versionen kritiserades för att den saknade flera viktiga egenskaper. Samarbete mellan olika utvecklare av BitTorrent-klienter ledde dock till en ny lösning som snabbt implementerades som betaversion i både Azureus- och µTorrent-klienterna. Dessa utvecklare kallade det nya protokollet för PE och därför betecknar man den slutliga versionen som MSE/PE. Idag används protokollet även i andra klienter som BitTornado, KTorrent och Mainline.

Enligt specifikationen ska användaren kunna välja mellan att bara kryptera headern eller hela uppkopplingen. Det är, så vitt vi vet, endast bara Azureus och µTorrent som tillåter detta, andra klienter har krypteringen på default. För att kunna uppnå kompatibilitet med de klienter som inte stödjer denna specifikation får man som användaren även välja om utgående och inkommande uppkopplingar ska vara krypterade. Allmänt är det dock så att om de stödjande klienterna får en krypterad ingående uppkoppling kommer krypteringen för den utgående uppkopplingen att tillämpas även om användaren inte valt det.

Meningen med krypteringen är att göra det svårare för Internetleverantörerna att spåra BitTorrent-trafik och därmed även svårare att sakta ner den. Den är dock inte designat för att ge anonymitet för användarna. [19]

5.1 MSE/PE

MSE/PE använder en ”Diffie-Hellman key exchange” i kombination med torrent-filens infohash för att etablera nyckeln och sedan används RC4-krypteringen för att kryptera datan. Diffie-Hellman key exchange är ett kryptografiskt protokoll som skapar en symmetrisk, d.v.s gemensam, engångsnyckel för två ökända parter.[20] Orsaken till att man tillämpar just Diffie-Hellman är dess bidrag till att minimera risken av passiva lyssnare och infohashet hjälper till att undvika ”man-in-the-middle attacks”. RC4 valdes för dess snabbhet, men man har modifierat den lite genom att förkasta det första kilobytet av RC4-data. Detta för att förhindra Fluhrer, Mantin och Shamir attacker från att lyckas knäcka protokollet.

Det påstås att några kanadensiska Internetleverantörer har lyckats blocka krypterad BitTorrent-trafik, men än så länge den är fortfarande effektiv gentemot andra Internetleverantörer. [21]

5.1.2 Diffie-Hellman

Diffie-Hellman, fungerar som följande: säg att man har Polly och André som vill använda detta protokoll för att skapa en delad nyckel. Talet N är känt för båda, men André väljer ett stort tal x och beräknar sedan $R1 = G^x \bmod N$ och skickar R1 till Polly. Hon i sin tur väljer ett stort tal y och beräknar sedan $R2 = G^y \bmod N$. Sedan skickar hon det till André som nu beräknar $K = (R2)^x \bmod N$ medan hon själv beräknar $K = (R1)^y \bmod N$. K är den symmetriska nyckeln som både nu känner till.

Faktumet att detta gäller: $(G^x \bmod N)^y$ mode N = $(G^y \bmod N)^x \bmod N = G^{xy} \bmod N$ gör att man kan tillämpa denna beräkningsmetod.

Diffie-Hellman har dock en svaghet och det är man-in-the-middle attack. Även om inkräktaren Erik inte vet värdena för x och y för att kunna attackera protokollet kan han ändå lura de ursprungliga parterna genom att skapa två nycklar; en mellan sig själv och André och en annan mellan sig själv och Polly. Detta beror på att när R1 och R2 skickas är det inkräktaren som mottar dem eftersom denne är placerat mellan Polly och André. Det vet dock inte Polly och André som tror att de skickar till varandra när trafiken egentligen hamnar hos Erik. [22]

5.1.3 RC4-kryptering

RC4, som står för Rivest Cipher 4, även känd som ARC4 och ARCFOUR, är den mest välkända mjukvara för ”stream cipher” som används i populära protokoll som SSL (Secure Sockets Layer). En ”stream cipher” är ett symmetriskt chiffer där varje tecken i klartexten krypteras en åt gången och där transformationen av varje successivt tecken varierar under krypteringen.

Det som utmärker RC4 är dess enkelhet, men samtidigt kan detta saknande av komplexitet vara en stor nackdel eftersom det leder till en låg säkerhetsnivå. Generellt anses RC4 att ha för låg säkerhetsnivå och brukar därför inte rekommenderas för implementation i nya system. [23]

Det som denna kryptering gör är att generera en ”pseudo-random” ström av bitar, även kallat ”keystream”, och för krypteringen kombineras den med klartexten med hjälp av en XOR-operation Dekryptering görs på samma sätt som vid själva krypteringen. Det är även samma nyckel används vid kryptering som vid dekryptering och detta är en av orsakerna till varför säkerheten för RC4 kryptering inte särskilt hög. Fördelen med symmetriska chiffer är dock att proceduren är snabb och därför med lämpad att tillämpas vid kryptering av långa meddelanden.

RC4 kryptering består av två delar: KSA som står för ”Key Scheduling Algorithm” och PRGA som står för ”Pseudo-Random Generator Algorithm”. Det är KSA som skapar en variabel längd ”key” som i sin tur initialiserar permutationen och det är först därefter som strömmen av bitar, keystream, kan genereras med hjälp av PRGA.
Den klarar sig inte mot Fluhrer-, Mantin- och Shamir-attacker som använder statistiken över de första bytes av data från keystream för att lista ut nyckeln till krypteringen.

5.2 Azureus och MSE/PE

BitTorrent-klienten Azureus är en av dem som har tagit upp kampen med Internetleverantörer som på olika sätt använder sig av traffic shaping av torrent-trafik. I programmet tar det sig uttryck genom att det går att ställa in olika säkerhetsnivåer. Den lägsta nivån är att Azureus inte krypterar torrent-trafiken men använder andra portar än standardportarna för kommunikationen. Den stora fördelen med detta är att CPU:n inte belastas extra med krypteringen, vilket den gör om man exempelvis tillämpar krypteringsalgoritmen AES (Advanced Encryption Standard). AES både kräver mycket CPU-tid samt många Diffie-Hellman-nycklar för att uppnå den höga säkerhet som algoritmen presenterar och det går emot BitTorrent-protokollets ursprungliga syfte att vara snabb och effektiv.

Att använda sig av den lägsta säkerhetsnivån hindrar dock bara Internetleverantörerna från att tillämpa traffic shaping om de använder sig av den metod som går ut på att lyssna till trafik på BitTorrent-specifika standardportar. Nästa nivå av säkerhet som klienten Azureus använder är en enkel kryptering av BitTorrent-headern. Det innebär att allt annat, d.v.s all annan data, sänds som vanligt i klartext. Metoden tar inte speciellt mycket CPU-kraft och man uppnår oftast en kompatibilitet med andra BitTorrent-klienter. Lite mer sofistikerade trafikformande noder upptäcker dock relativt lätt att det rör sig om BitTorrent-trafik då det mesta av datamängden som sagt sänds i klartext.

För att ytterligare ta säkerheten en nivå så kan användaren ställa in RC4-kryptering i sin applikation. Med denna inställning får man fortsatt hög kompatibilitet och undviker egentligen helt traffic shaping för trafik som i alla fall går ut från klienten. En klar nackdel är att det fortfarande är fullt möjligt att upptäcka vilken port som BitTorrent använder för sin inkommande kommunikation. För att svara upp mot detta problem finns nästa nivå som inte tillåter mekanismer vars syfte är att identifiera av inkommande portar för BitTorrent-trafik på klientdatorn. Det blir ett relativt kraftfullt försvar mot traffic shaping av BitTorrent-trafik som helhet. Problemet är bara kompatibiliteten mot äldre klienter och servrar. Filöverföringen blir med denna inställning endast möjlig mellan klienter som inte använder det klassiska sättet att ansluta till inporten för BitTorrent.

Vidare finns ytterligare ett alternativ och det är att även blockera identifiering av utgående anslutningsport för BitTorrent. Med denna inställning finns ingen bakåtkompatibilitet och klassiska BitTorrent-anslutningar hindras helt. Om väldigt få användare har denna inställning blir det svårt att överhuvud taget uppfattas som en peer i BitTorrent-nätverket. Slutligen, den högsta säkerheten innebär detsamma som nivån ovan beskriven, dessutom införs en funktion som gör att tracker-filen inte ger ut datorns IP-adress och därmed förhindras alla anslutnings försök till klienten. Denna säkerhetsnivå innebär givetvis att användaren får en viss anonymitet (även om det går att få fram IP-adressen) för sin filöverföring, men i dagsläget så begränsar den också kraftigt antalet peers som kan ansluta till användaren.

5.3 Etiska frågor kring kryptering av BitTorrent trafik

Om nu alla BitTorrent-program börjar kryptera trafiken som skickas, kommer den totala mängden av krypterad P2P-trafik naturligtvis att öka lavinartat. Detta leder till att övervakande organ, exempelvis polismyndigheter, omöjligt kan dekryptera och kontrollera all information, även om det är primitiva krypteringsnycklar som används. Det leder i sin tur till att terrorister, pedofiler och andra brottsliga nätverk kan skicka krypterad information utan att dra uppmärksamhet till sig, och på så sätt kan undvika att kontrolleras. Alltså vilar ett stort ansvar på programmerarna som inför kryptering i sina BitTorrent-program. Tyvärr försvåras situationen ytterliggare av att en stor mängd av BitTorrent-trafiken är upphovsrättsskyddad, vilket leder till att många vanliga användare kan råka illa ut om trafiken avslöjas för polisiära organ. En ändring i lagstiftningen angående upphovsrätten kan således vara nödvändig för att kunna sålla ut den verkligt farliga trafiken från det stora havet av krypterad P2P-trafik på Internet.

5.4 Identifiering av BitTorrent-trafik

Det finns ett antal metoder som Internetleverantörer kan använda för att upptäcka BitTorrent-trafik. Ett sätt är att undersöka paketen som kommer och ”packa upp” dem för att kolla vad de är för typ av protokoll. Utifrån detta kan sedan ett beslut tas om just det protokollet ska prioriteras, bandbegränsas eller kanske blockeras. Denna metod kallas ”Deep packet

inspection” (DPI) och en implementering innebär att en enhet måste finnas i nätverket som kan titta på de anländande paketen på en nivå motsvarande Layer 2 och uppåt i OSI modellen. Som svar på denna typ av analys i nätverken har många klienter som vi tidigare nämnt börjat kryptera BitTorrent-trafik för att på så sätt dölja vilken typ av trafik det rör sig om.

Ett annat möjligt tillvägagångssätt att identifiera specifikt BitTorrent-trafik är att lyssna på vissa portar. Det faktum att många BitTorrent-klienter kommunicerar över ett bestämt intervall av portar gör denna metod relativt möjlig. Den klient vi använde i våra tester använde exempelvis konsekvent portarna 6881-89 för inkommande trafik innan den ger upp att kommunicera. Dessa portar är generellt vanligast för inkommande BitTorrent-trafik. Problemet med denna approach är dock att klienter och servers som alltid använder olika portar blir omöjliga att upptäcka.

Ytterligare ett förfarande för Internetleverantören är att analysera karaktäristiken på trafiken för att avslöja att det rör sig om BitTorrent-trafik. Den stora fördelen med detta sätt är att de går att göra även om trafiken går på olika portar och BitTorrent-headern är krypterad. Denna metod syns alltså idag vara den kanske mest heltäckande lösningen för att identifiera BitTorrent-trafik.

6. Tester

Med hjälp av nätverksemulatorn NISTnet genomförde vi ett antal tester för att undersöka hur BitTorrent-trafik kan manipuleras. Syftet med testerna var att undersöka möjliga sätt för hur en Internetleverantör av olika anledningar skulle kunna försämra eller möjligtvis förbättra villkoren för BitTorrent-trafik.

6.1 Testmiljö

Nätverket konfigurerades så att ett lokalt nätverk upprättas mellan två PC-laptops, som vi kallade Polly och André (efter dessas rättmäktiga ägare). På Polly kördes operativsystemet Windows XP Home Edition med SP2 och på André Ubuntu 6.06 ”The Dapper Drake”. En mer detaljerad hårdvaruspecifikation återfinns i appendix A. André kopplades upp mot Internet med WLAN och André lät vi tilldelas en IP-adress av KTH-Open i Fylkesalarna på KTH-campus Valhallavägen. Den angivna kapacitet för denna länk var 54 Mbps. Polly kopplades till André via ethernetlänk med hjälp av en korsad UPT-kabel. Länken mellan Polly och André stödde en maxhastighet på 100 Mbps. Vi tilldelade Andrés ethernetuppkoppling IP-adressen 192.168.0.2 och Pollys 192.168.0.3. Pollys standardgateway sattes till 192.168.0.2 (d.v.s Andrés adress). Båda datorernas ethernetuppkoppling gav vi nätmasken (netmask) 255.255.255.0.

Från Polly initierades sedan via BitTorrent-klienten Azureus torrentnedladdningar som bevakades samt manipulerades på André via emulatorn NISTnet. I samtliga test laddades samma fil ner med hjälp av samma torrent tracker. Filens storlek var 3,24 Mb och tillgänglig för fri distribution. Vi loggade all BitTorrent-trafik under nedladdningen med hjälp av programvaran Ethereal på André.

6.2 Test – varierande parametrar i NistNet

Vi inledde med att logga hur nedladdningstiden påverkades av olika trafikformande variabler inställda i NISTnet på André. Syftet var att se om de finns någon speciell fördelaktig metod för att bandbegränsa BitTorrent-trafik. Eftersom nedladdningstiden av en torrent påverkas kraftigt av antalet uppladdande parter, peers, så valde vi att lägga in en låg övre bandbreddsgräns på 30Kb/s, förutom när vi testade variationer i bandbredd naturligtvis. Detta för att filtrera bort variationer i nedladdningshastighet påverkade av hur många uppladdare som för tillfället fanns att tillgå. Testerna kördes strikt efter varandra och filen hade i stort sett samma antal peers (seeders och leechers) under testen. De parametrar vi varierade i NISTnet var:

- Bandbredd (bandwidth)
- Fördröjning (delay)
- Packet drop

Resultaten av olika parametrar inställda och resultaten (nedladdningstiden) de medföljde kan utläsas ur följande tabeller:

Tabell 6.1 Nedladdning med olika parametrar på Nistnet

Delay (ms)	0	100	500
Download time (s)	108	159	279
Average rate (Kb/s)	29.8571	20.3773	11.61

Bandwidth limit (kb/s)	30	20	10
Download time (s)	108	226	410
Average rate (Kb/s)	29.8571	14.3362	7.9024

Drop rate (%)	1	5	10	20
Download time (s)	143	165	228	355
Average rate (Kb/s)	22.6573	19.6363	14.2105	9,1267

Enligt vår analys spelar det mindre roll om man påverkar BitTorrent-trafiken genom att använda sig av en fördröjning eller om man begränsar bandbredden. Packet drop bör dock inte användas, eftersom detta medför att många paket måste skickas om, vilket betyder onödig extra trafik för Internetleverantören att behandla.

6.3 Test – lokal nedladdning vs. extern nedladdning

Med samma konfiguration som ovan ville vi simulera att Andre var Internetleverantör till Polly och därmed titta lite på hur Andre som Internetleverantör kan styra BitTorrent-trafiken i sitt nät. Vi använde samma torrent-fil på 3,24 Mb som tidigare. Fördröjningen slogs på innan nedladdningen börjat och slogs av efter att nedladdningen var klar. Det som betecknas som det externa nätet är utanför André. Det innebär exempelvis att extern BitTorrent-nedladdning refererar till att metainfo-filen fanns utanför det lokala nätverket; intern betyder motsatsen. Den externa ftp-nedladdningen skedde inte från samma plats som BitTorrent- nedladdningen, dvs. testfilen och dess metainfo-fil fanns inte på samma server. Vi antog att det inte skulle ha mindre betydelse för testet, då metainfo-filen länkar till en och samma tracker-fil på annan server. Testet bestod av dessa fyra moment:

1. Nedladdning av torrent-fil till Polly externt utanför Andre.
2. Nedladdning av torrent-fil innanför Andre till Polly.
3. Nedladdning av vanlig fil till Polly externt utanför Andre.
4. Nedladdning av vanlig fil innaför André till Polly.

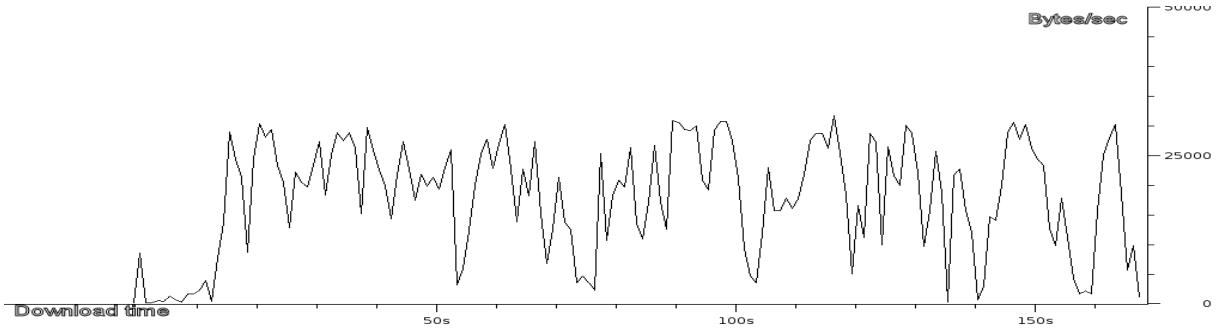
Syftet med experimentet var att kunna se hur torrent-nedladdning påverkas av att Andre manipulerar olika delays på trafiken via NISTnet. Vi ville med testet även undersöka om man kunde finna någon speciell karaktär på torrent-nedladdningen som skiljer den från annan trafik.

Resultat
Den information som framgår i tabell 6.2 visar först och främst hur trafik ter sig då filer överförs till Polly då ingen manipulation från André gjorts. Man kan utläsa att hastigheterna för nedladdning skiljer sig markant mellan de olika försöken. Den införda fördröjningen tycks ha störst effekt på nedladdning internt och för extern ftp-trafik. Att BitTorrent-trafik internt regerar starkare på delay än den som kommer externt ifrån skulle kunna visa att traffic shaping har större effekt när nedladdning sker inom Internetleverantörens eget nät. Ftp-nedladdning inom det lokala nätverket ser vi får en stor kapacitetsförsämring i paritet med den tillagda fördröjningen.

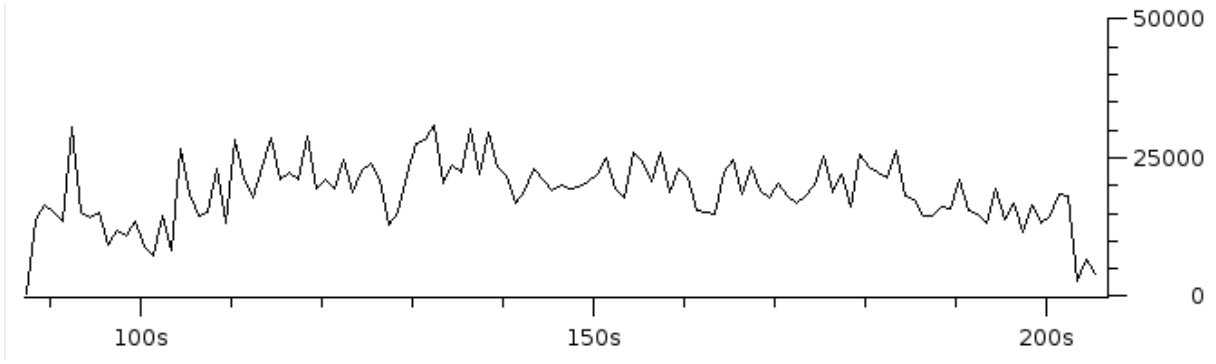
Tabell 6.2 Beskriver förhållandet mellan nedladdningstid och tillagd fördröjning i olika försök.

	download time (s)	added delay (ms)	Average rate (kb/s)
BitTorrent - externt	205	0	15,80
	249	100	13,01
	686	500	4,70
	862	1000	3,75
BitTorrent - internt	7	0	462,85
	14	100	231,42
	81	500	40,00
	167	1000	19,40
FTP - externt	4	0	810,00
	19	100	170,52
	66	500	49,09
	94	1000	34,46
FTP – internt	0,3	0	10 800,00
	7	100	462,85
	50	500	64,80
	102	1000	31,76

Vidare så tittade vi på hur nedladdningshastigheten förändrades under nedladdningsförloppet. Med en funktion i Ethereal jämförde vi graferna (figur 6.1 och figur 6.2) över torrent-nedladdningen med den över ftp-nedladdningen. Graferna är från försöket ovan men representerar avsnitt ur nedladdningarna. Det som syns är bland annat att de båda sekvenserna har medelnedladdningshastigheten kring bandbegränsingen på 30 kb/s men att torrent-trafiken har större varians. Detta tror vi beror på att torrent-trafik momentant är beroende av att det alltid finns en ”ny” torrent-piece tillgänglig. Om det helt plötsligt inte finns någon önskad piece faller nedladdningen kraftigt tills att det åter finns en peer som har en ny piece.



Figur 6.1 Illustrerar BitTorrent-nedladdning med bandbegränsningen 30kb/s av en fil på 3.24 mb

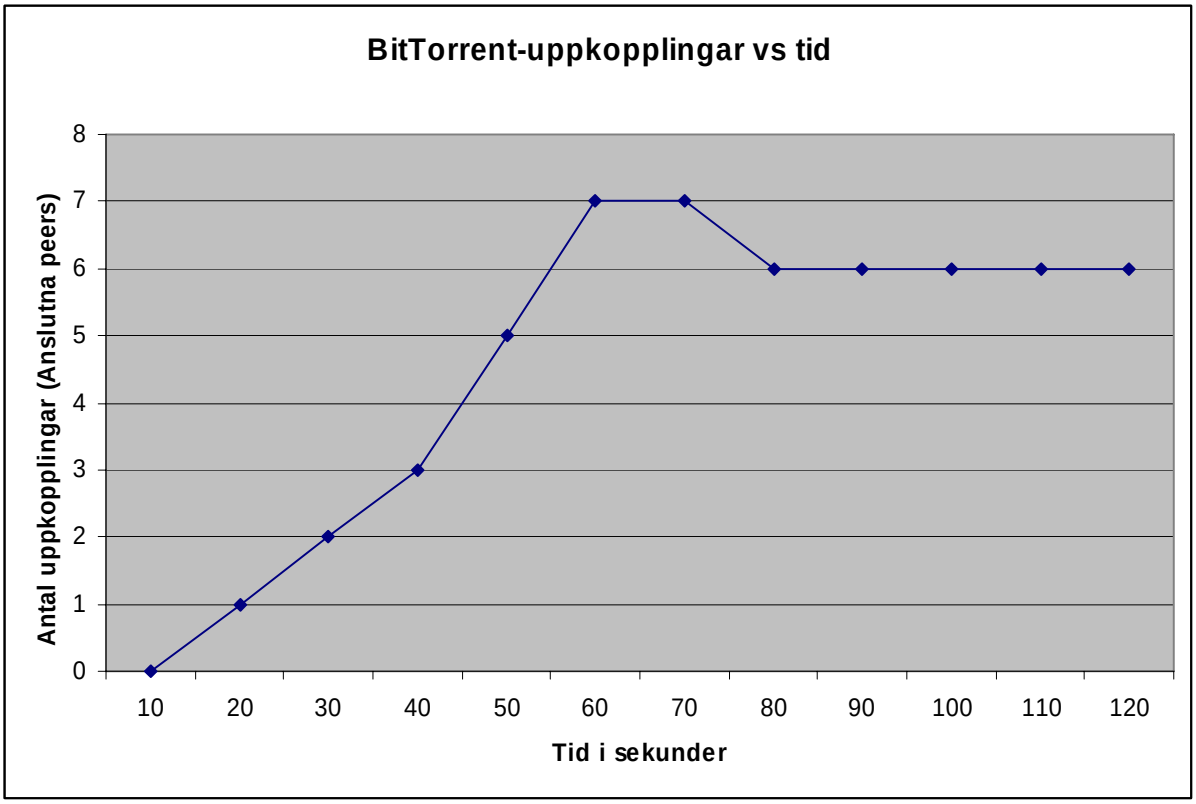


Figur 6.2 Illustrerar FTP-nedladdning med bandbegränsningen 30kb/s av en fil på 3,24mb

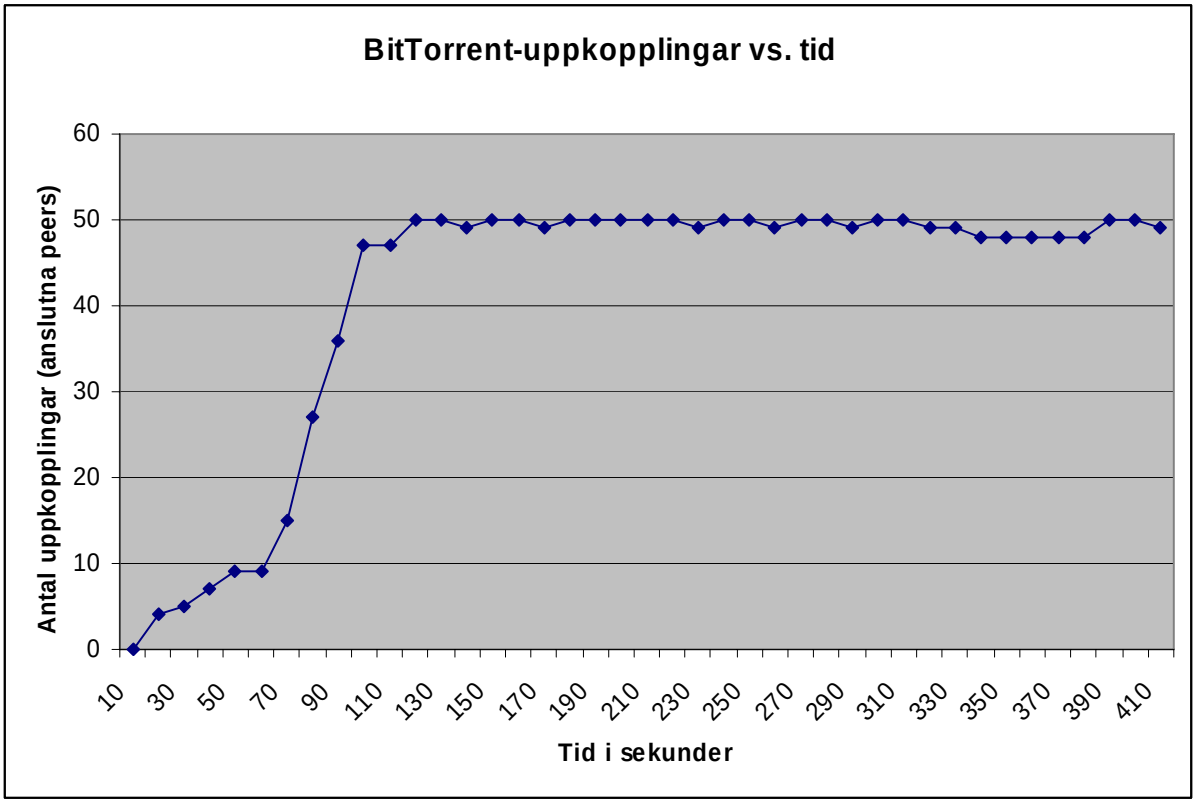
Överföringshastigheten går kraftigt ned för BitTorrent trafik i takt med större fördröjningar. Metoden att införa fördröjningar på BitTorrent trafik är därmed ett sätt att begränsa användandet av Internetleverantörernas totala bandbredd.

Vidare tittade vi på hur antalet anslutna peers, d.v.s. hur antalet BitTorrrent-uppkopplingar förändrades över tiden. För att illustrera detta plottade vi ytterligare två nedladdningsförlopp. Det ena med den fil vi hittills använt och den andra är en mycket större fil på 160 Mb.

Figur 6.3 visar att antalet anslutna peers successivt stiger under den första halvan av nedladdningsförloppet för att sedan stabilisera sig. I jämförelse med figur 6.4 ser vi ett liknande beteende i nedladdningsförloppets inledning. Det som händer är att tracker-filen letar reda på och kopplar upp sig till de tillgängliga peers, vilket kan betecknas som ”uppsökningsfasen”. Det är just denna uppsökningsfas som utgör skillnaden mellan dessa två nya tester; för den lilla filen (den på 3,24 Mb) tar uppsökningsfasen upp en mycket större del av hela nedladdningsförloppet än för den stora. Nedladdningshastigheten varierar under uppsökningsfasen och stabiliseras först när antalet anslutna peers gör det. Detta innebär att den låga nedladdningshastigheten i början för en större effekt på den lilla filen om man ser till hela nedladdningen. Vi anser att detta kan vara orsaken till att det upplevs som långsammare och mindre effektivt att ladda ner små filer med BitTorrent.



Figur 6.3. BitTorrent-uppkopplingar över tid i fallet med en ”liten” fil på 3,24 Mb.



Figur 6.4 BitTorrent-uppkopplingar i då den nedladdade filen var mycket större.

6.4 Test – variation av tidpunkt för fördröjning.

Detta test bestod i att vi laddade ner torrent-filen på 3,24 Mb till Polly via Andre. Med NISTnet på Andre använde vi två olika strategier för när vi startade fördröjningen. Vi började med att testa nedladdningen under normala omständigheter.

1. Fördröjning i nedladdningens initialskede
2. Fördröjning i nedladdningens slutskede

Med detta test ville vi försöka identifiera hur nedladdningshastighet påverkades av att lägga till fördröjningen bara i vissa skeden av nedladdningsprocessen. I början av nedladdningen sker handskakningar och avgöranden om vilka som filen ska laddas från. Senare sker själva filöverföringen. Vi ville därmed titta på skillnader och likheter mellan de två strategierna. Testets två fall utfördes direkt efter varandra med i stort sett samma peers.

Det gick inte att identifiera några förändringar i vilka peers som Polly fick filen från när vi la till fördröjningar i olika skeenden. Men när vi införde fördröjning på 1000 ms i nedladdningens början låg nedladdningshastigheten just i början väldigt lågt och stabiliserade sig runt 8 kb/s. I det andra fallet när vi slog till fördröjningen *efter* initialskedet steg slutligen nedladdningshastigheten mycket högre och varierade mer.

Tabell 6.3

Delay (ms)	Average rate (kb/s)
0	30 kb/s
1000 i startskedet	8 kb/s (varierade mellan 1 kb/s – 11 kb/s)
1000 hela filen utom i starten	12 kb/s (varierade mellan 1 kb/s- 20 kb/s)

Det är svårt att utifrån denna undersökning dra några slutsatser om när i nedladdningsförloppet som Internetleverantörerna bör införa fördröjningen av torrent-trafik eller om det ens spelar någon roll. Variationerna är så pass små mellan de två fallen. Man skulle dock kunna tänka sig att anledningen till resultatet är att en fördröjning i början innebar att ”fel” servrar valdes ut som nedladdare och att det var anledningen till att filen totalt sett laddades ner i ett långsammare tempo. I det senare fallet kunde handskakningsprocessen göras på ett normalt sätt och därför kanske fördröjningen på själva nedladdningen av filen inte hade lika stor inverkan eftersom ”rätt”(i meningen verkligt snabbast) servar valts från början.

7. Slutdiskussion

Vi tror att BitTorrent är här för att stanna. Protokollets egenskaper är utmärkta för många av dagens Internetanvändare som vill fildela, men ofta sitter på asymmetriska uppkopplingar med snabbare download än upload. Därför tror vi att det kommer bli allt viktigare för Internetleverantörerna att hantera BitTorrent-trafik på ett effektivt sätt. Konstruktiv traffic shaping, och andra nyttiga verktyg, som t.ex. CDP, är en viktig del i den utvecklingen. För att lyckas med detta krävs naturligtvis möjligheten att särskilja BitTorrent-trafik från övrig dito. Vi ser en utveckling av alltmer döljande BitTorrent-klienter, som implementerar kryptering och inte längre opererar på en och samma port. Mycket av detta, kan man tänka sig, grundar sig i ett behov att ”lura” Internetleverantörerna. Denna utveckling, om än förståelig med tanke på vissa Internetleverantörernas negativa attityd till BitTorrent, anser vi olycklig. Istället borde tillverkarna av BitTorrent-klienter jobba tillsammans med stora Internetleverantörerna för att uppnå bättre effektivitet. Samtidigt borde alla Internetleverantörer inse att många av

deras kunder använder BitTorrent, och att en kampanj mot protokollet med största sannolikhet inte är långsiktigt lönsam.

BitTorrent är som vilket annat protokoll som helst, det är inte protokollet i sig som ska motarbetas i bekämpningen av fildelning av piratkopior. Dessutom är det ett faktum att den används för legala syften. Om Internetleverantörerna vill uppnå en mer rättvis prissättning kan man erbjuda lägre prissatta uppkopplingar med begränsningar i hur mycket data man som kund får ladda upp och ned, istället för att bestraffa BitTorrent-användare med långsamma nedladdningshastigheter.

Som Internetleverantör är det också viktigt att man lever upp till sina löften. Erbjuder man 10 Mbit/s ska det också vara 10 Mbit/s och inte ”10 Mbit/s förutom när det gäller P2P-trafik”. Det är också många som tycker att det är ett löjligt agerande från Internetleverantörernas håll att erbjuda supersnabba uppkopplingar samtidigt som man begränsar vissa typer av trafik bara för att den skapar komplikationer för dem. Man behöver helt enkelt inte en 100 Mbit-lina för att kolla sin e-post och surfa på Internet.

Vi anser att det är bra att Internetleverantörerna är måna om kvalitén i deras nätverk. Så länge traffic shapingen som Internetleverantörerna ägnar sig åt har ett syfte att upprätthålla en god prestanda för så många nätverksbaserade applikationer som möjligt är det av godo. När så användandet av en typ av applikation som BitTorrent blir så spritt att det börjar gå ut över andra applikationers prestanda är Internetleverantörernas traffic shaping-handlande möjligen legitimerat. Å andra sidan om kunderna efterfrågar en så hög andel BitTorrent trafik som tidigare berättats om i denna text så blir det en nödvändighet för Internetleverantörerna att svara upp mot detta så att kunderna blir nöjda.

Den ”onda” Internetleverantören kan använda traffic shaping för att begränsa den tunga BitTorrent-trafik som försöker ansluta till datorer utanför det egna nätet och tillåter Bittorrent-trafik från andra om det är en Internetleverantör som betalar för utnyttjandet. Önskvärt vore kanske en implementering av en rättvis trafikformande algoritm som fördelar Internetleverantörens tillgängliga bandbredd så att det inte går ut över prestandan på Internetleverantörens egna abonnenter. Detta innebär att Internetleverantörerna bör fördjupa sina kunskaper om hur protokoll som BitTorrent verkar i sina nät för att maximera kundens upplevda kvalité av sin Internet-uppkoppling.

Referenser

[1] http://news.bbc.co.uk/2/hi/programmes/newsnight/4758636.stm; Författare: Adam Livingstone; Titel: ”A bit of BitTorrent bother”; Utgivare: BBC Newsnight; Senast besökt: 2006-10-26

[2] http://www.dtc.umn.edu/~odlyzko/doc/itcom.internet.growth.pdf; Andrew M. Odlyzko; “Internet traffic growth: Sources and implications”; University of Minnesota, Minneapolis, MN, USA; 2006-10-26

[3] http://www.torrentguide.net/index_sv.asp; “Hur fungerar BitTorrent?”, 2006-10-26

[4] http://www.bittorrent.org/protocol.html; BitTorrent.org; 2006-10-26

[5] http://en.wikipedia.org/wiki/Bencoding ; ”Bencode”; Wikipedia the free encyklopedia; 2006-10-26

[6] http://www.bittorrent.org/protocol.html; BitTorrent.org; 2006-10-26

[7] http://www.bittorrent.org/bittorrentecon.pdf; Bram Cohen; “Incentives Build Robustness in BitTorrent”; 2006-10-26

[8] http://www.bittorrent.org/protocol.html; BitTorrent.org; 2006-10-26

[9] http://wiki.theory.org/BitTorrentSpecification#peer_id; “Peer id”; theory.org; 2006-10-26

[10] http://en.wikipedia.org/wiki/MPAA; “Motion Picture Association of America”; wikipedia the free encyklopedia; 2006-10-26

[11] http://torrentfreak.com/encrypting-BitTorrent-to-take-out-traffic-shapers; Ernesto; “Encrypting Bittorrent to take out traffic shapers”; Torrentfreak; 2006-10-26

[12] http://en.wikipedia.org/wiki/Traffic_shaping; “Traffic shaping – Wikipedia, the free encyclopedia”; 2006-10-26

[13] http://news.bbc.co.uk/2/hi/programmes/newsnight/4758636.stm; Adam Livingstone;”A bit of BitTorrent bother”; BBC Newsnight; 2006-10-26

[14] http://en.wikipedia.org/wiki/Quality_of_service; “Quality of service – Wikipedia, the free encyclopedia”; 2006-10-26

[15] http://www.itworld.com/nl/sup_mgr/05142001; Ray Keneipp; “Traffic Shaping”; 2006-10-26

[16] http://www.azureuswiki.com/index.php/Bad_ISPs; “Bad ISPs – AzureusWiki; 2006-10-26

[17] http://www.cachelogic.com/home/pages/understanding/options.php#; “CacheLagic: Understanding the Impact of P2P”; 2006-10-26

[18] http://torrentfreak.com/cachelogic-and-bittorrent-introduce-cache-discovery-protocol; Ernesto; ”Cachelogic and BitTorrent Introduce Cache Discovery Protocol at Torrentfreak”; 2006-10-26

[19] http://en.wikipedia.org/wiki/BitTorrent_protocol_encryption; “BitTorrent protocol encryption – Wikipedia, the free encyclopedia”; 2006-10-26

[20] “Data Communications and Networking”, Behrouz A. Forouzan s. 824.

[21] http://en.wikipedia.org/wiki/BitTorrent_protocol_encryption; “BitTorrent protocol encryption – Wikipedia, the free encyclopedia”; 2006-10-26

[22] “Data Communications and Networking”, Behrouz A. Forouzan s. 825.

[23] http://en.wikipedia.org/wiki/RC4; ”RC4 – Wikipedia, the free encyclopedia”; 2006-10-26

Appendix A: Testutrustning

Pollys laptop - Compaq Presario 2500

- Processor: Pentium 4, klockfrekvens: 2.8 GHz
- Minne: 512 Mb DDR2 SDRAM
- Hårddisk: 40 Gb ATA
- Skärm 15,4 tum TFT med (64 Mb delat minne)
- Nätverkskort: RJ-45 10/100 Ethernet

OS: Microsoft Windows XP Home Edition

Andrés laptop - HP Compaq nx6310

- Processor: Intel Celeron M 410, klockfrekvens: 1.49GHz
- Minne: 512 Mb DDR2 SDRAM
- Hårddisk: 60 Gb SATA 5400 rpm
- Skärm 15.4 tum TFT med (128 Mb delat minne)
- Nätverkskort: Broadcom 10/100 Ethernet och Integrerat 802.11 a/b/g WLAN

OS: Ubuntu Linux 6.06 LTS, ”The Dapper Drake”, desktop version

Appendix B: Metainfo-filen

Enkel metainfo-fil:

Nyckel	Beskrivning
Announce	Trackerns URL
Info	En ordlista som beskriver filen
- length	Heltal. Filens längd i bytes.
- name	Sträng. Filnamnet på en sträng.
- piece length	Heltal. Antal bytes som varje del av filen innehåller.
- pieces	Binärkodad. Sträng som består av konkateneringen av filens samtliga 20 byte stora SHA 1-hashvärden, en för varje del.

Den andra typen av metainfo-fil, där torrentfilen har underkataloger, har följande struktur:

Nyckel	Beskrivning
Announce	Trackerns URL
Info	En ordlista som beskriver filen
Ofiles	En lista över ordlistor, en för varje fil. Varje ordlista I denna lista innehåller följande nycklar.
- length	Heltal. Filens längd i bytes.
- path	En lista som innehåller en eller flera strängar som tillsammans representerar en viss adress och filnamn. Varje element i listan motsvaras av antingen ett katalog namn eller ett filnamn.
Oname	Sträng. Namnet på den översta katalogen alltså den som innehåller alla filer som finns i listan.
opiece length	Heltal. Antal bytes som varje del av filen är.
Opieces	Binär kodad. Sträng som består av konkateneringen av alla 20-byte SHA 1 hashvärden, en för varje del.

Appendix C: Tracker-filen

Nyckel	Beskrivning
Infohash	Samma 20 byte SHA1-hash av informationsvärdet från metainfo-filen.
Peer id	En unik sträng på 20 byte som en användare erhåller då man börjar ladda ner.
Ip	IP-adress eller DNS namn som beskriver var peeren finns.
Port	Porten som peeren lyssnar på.
Uploaded	Total mängd som uppladdats.
Downloaded	Total mängd som nedladdats.
Left	Antal bytes som denna peer har kvar att ladda ner.
Event	Kan vara ”started”, ”completed”, eller ”stopped”. ”Started” betyder att nedladdning börjat. ”Completed” kännetecknar att nedladdningen är klar.