Contents

1.1 Abbreviation

Peer: A peer is a node in a network participating in file sharing. It can simultaneously act both as a server and a client to other nodes on the network.

Neighboring peers: Peers to which a client has an active point to point TCP connection.

Client: A client is a user agent (UA) that acts as a peer on behalf of a user.

Torrent: A torrent is the term for the file (single-file torrent) or group of files (multi-file torrent) the client is downloading.

Swarm: A network of peers that actively operate on a given torrent.

Seeder: A peer that has a complete copy of a torrent.

Tracker: A tracker is a centralized server that holds information about one or more torrents and associated swarms. It functions as a gateway for peers into a swarm.

Metainfo file: A text file that holds information about the torrent, e.g. the URL of the tracker. It usually has the extension .torrent.

Peer ID: A 20-byte string that identifies the peer. How the peer ID is obtained is outside the scope of this document, but a peer must make sure that the peer ID it uses has a very high probability of being unique in the swarm.

Info hash: A SHA1 hash that uniquely identifies the torrent. It is calculated from data in the metainfo file.

1.2 Background of BitTorrent

BitTorrent is a new and popular Peer-to-Peer (P2P) file sharing protocol. It is one of the most common protocols for transferring large files. BitTorrent network relies on Internet search engines that index files through metadata called torrents. Bit Torrent client software allows files to be downloaded and uploaded on P2P networks using a high-performance network protocol.

Whereas the conventional peer to peer system takes files from computer A to computer B BitTorrent takes data from computer A and distributes the data far more effectively. It is based on packet switching as are most programs but the packets take a different form.

Bit torrent technology takes a file and split it into a quantity of small packets or bits. As one peer makes one packet available you make a request to download that packet. The file can be downloaded independently form one another. As you leech (download) files from the users you also begin to share (seed) these files with them. This means that you may have ten to twenty people sharing the same video file and each one has certain bits of that file. As the download progresses and the parts are collected you begin to get the whole, in other words a file that you can play on the computer.

BitTorrent client is any program that implements the BitTorrent protocol. Each client is capable of preparing, requesting, and transmitting any type of computer file over a network, using the protocol. A peer is any computer running an instance of a client.
e.g., Bit Comet supports simultaneous downloads, HTTP and FTP downloading, webseeding, search for additional HTTP and FTP sources when downloading through the BitTorrent client, download queuing, selected downloads inside a torrent package, fast-resume, Mainline DHT, protocol encryption, disk cache, speed limits, port mapping, peer exchange (PEX), UDP NAT traversal, proxy, and IP filtering. Bit Comet contains an embedded Internet Explorer window for the purpose of allowing users to search for torrents within the client.

1.3 Common BitTorrent traffic handling in ISP

Hundreds of larger and smaller ISPs all over the world try to limit BitTorrent traffic on their networks by throttling/decreasing the BitTorrent download speed.

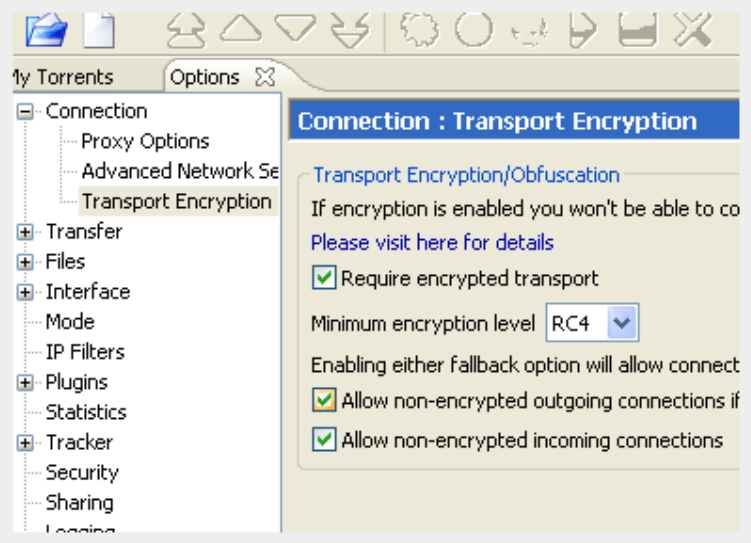
Unfortunately, most companies are not very open about their network management solutions, and according to the BBC, the 'bandwidth war' has begun.

We don't know whether our traffic is being throttled unless you have checked your ISP.

Users want to prevent the limitation by ISP, they encrypt the torrents to prevent throttling ISP's from shaping the traffic by using three most popular torrent clients i.e. **Azureus, uTorrent, and Bitcomet.**

What does encryption do?

The RC4 encryption obfuscates not only the header but also the entire stream. This means that it's very hard for your ISP to detect that the traffic you are generating comes from BitTorrent. The RC4 uses more CPU time than the plain encryption or no encryption. It is however harder to identify for traffic shaping devices.



2.1 Detecting BitTorrent blocking

2.11 Introduction

Purpose of BitTorrent blocking

In the Internet society, bandwidth is a very precious resource. ISP is competing on the market by offering cheaper and cheaper plan with unlimited bandwidth.

BitTorrent, a P2P file transfer technology, is utilizing a particular TCP port for file sharing between peers. Most of the ISP in the world didn't try to block BitTorrent traffic but some will utilize some kinds of technique to rate-limit or even block BT traffic based on their Internet traffic management policies.

In our project, we will try to explore one of the techniques in detecting BitTorrent blocking developed by Glasnost and some other technique in detecting BitTorrent blocking.

2.12 BitTorrent Traffic Blocking Technique

In the ISP world, there are basically two traffic blocking techniques being employed. The two basic traffic-blocking techniques were classified based on the location of the detection and enforcement in the link.

Inline Inspection/Enforcement technique

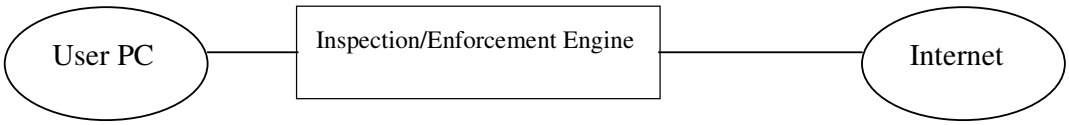


Figure 1 *– Inline Inspection/Enforcement Technique*

In the figure shown above, ISP will divert all traffic from User PC to the Inspection and enforcement engine before allowing traffic to go to the Internet. All traffic needs to goes through the policies set by the engine. If traffic is not allowed to go through, that traffic will be dropped.

Advantage:

- Easy to setup, flexible in handling the traffic shaping.

Disadvantage:

- Since the enforcement engine is placed on the path to the Internet, any failure or slow performance in the engine will cause dissatisfactory result to the user.

Non-Inline Inspection/Enforcement Technique

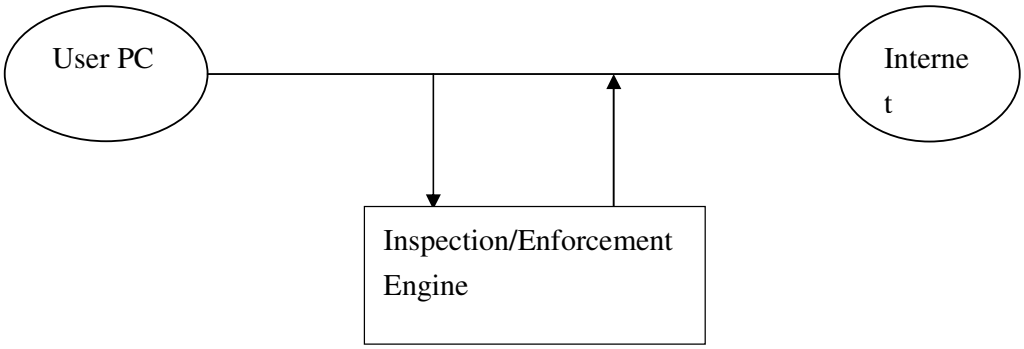


Figure 2 *– Non-inline inspection/enforcement technique*

In this kind of traffic inspection, the inspection/enforcement engine is not sitting on the traffic path between User PC and the Internet. The engine will passively monitor the traffic from the User PC to the Internet. When it detected traffic that need to be blocked or rate-limited as defined in the engine's policies, it will send TCP RST packet to the source addresses so as to torn down the TCP connections.

Advantage:

- It does not affect the traffic performance between the User PC and the Internet

Disadvantage:

- It cannot rate-limit traffic as the inline inspection/enforcement technique done.

For both the inline and non-inline inspection/enforcement technique, the inspection/enforcement engine needs to inspect the traffic and classify them. Traffic classification need to be done. In modern BitTorrent technology, since BitTorrent traffic does not necessarily utilize well-known TCP port, ISP need to use either the deep packet inspection or the traffic characteristics analysis to classify different kinds of traffic.

3.1 Different kinds of Blocking BitTorrent traffic methods

1) TCP Port

In this method, the ISP will look at the TCP port number specified in the IP header and check if it is a well-known BitTorrent port. If the TCP port number is belonging to the well-known TCP port, it will drop those IP packets and then send a TCP RST to the source address in order to torn down the TCP connections that are suspicious of BitTorrent transfer.

2) Direction

In Internet, there are two different directions of traffic, the upstream and the downstream traffic. The upstream traffic is from the User's PC uploading to the Internet while the downstream traffic is from the Internet downloading to the User's PC. For BitTorrent traffic, since it is a P2P traffic model, there will be large amount of data being found in the upstream traffic. Whenever the inspection/enforcement engine found

3) Protocol

In this kind of blocking, the ISP will utilize the Deep inspection technique; it will look into the message portion of the IP packet and check to see if it is a BitTorrent traffic packet. The protocol field of the IP packet should contain the code BTP (BitTorrent Protocol). If traffic packet were found to contain the BTP code in the protocol field, those traffic packet will be identified as BitTorrent traffic and will be dropped and a TCP RST will be send back to the source address.

3.2 Choice of tools

Prior to using the tools for testing whether or not the ISP has implemented any blocking mechanisms, it is worth to point out that there are many different ways to measure network traffics and each of them may have pros and cons.

For example, testing software measures the traffic actively by replicating fake signal. If testing in this way, the software has to be capable of executing in variety of operating systems and network environment. Otherwise, software has to be implemented under a restricted configuration for simplicity.

On the other hand, proactive testing software would mainly stick on 1 to 2 protocols or just interference by using the existing protocol that currently available.

In terms of detect the blocking, some software may try to examine the interference or any delay on packet that was sent by its own. While others may need two or more parties by talking in between each other and then comparing and interpret whether there is any blocking has been applied as a result. These multiple parties may also need to install a server in order to kick-off the testing.

In addition, software may examine the traffic on packet basis or just tracking the speed to connect to a web page without looking into detail about any specific packet. In fact, it is yet to know which approach is the best way to perform the measurement and different software may be able to apply adequately under different circumstances. The software does give us an initial start for those researchers that are interested in evaluating this information.

Following are the known ISP testing software on the market (Copy from link http://www.eff.org/testyourisp)

Tool	Active / Passive	# Participants per Test	Platform	Protocols	Notes
Gemini	Active	Bilateral	Bootable CD	Unknown	Uses pcapdiff
Glasnost	Active	1.5 sided	Java applet	BitTorrent	
ICSI Netalyzr	Active	1.5 sided	Java applet + some javascript	Firewall characteristics, HTTP proxies, DNS environment	
ICSI IDS	Passive	0 sided (on the network)	IDS	Forged RSTs	Not code users can run
Google/New America Measurement Lab	Active	2 sided	PlanetLab (server), Any (client)	Any	A server platform for others' active testing software
NDT	Active	1.5 sided	Java applet / native app	TCP performance	A sophisticated speed test
Network Neutrality Check	Active	1.5 sided	Java applet	No real tests yet	Real tests forthcoming; discussion
NNMA	Passive	Unilateral	(currently) Windows app	Any	
pcapdiff / tpcat	Either	Bilateral	Python app	Any	A tool to make manual tests easier. EFF is no longer working on pcapdiff, but development continues with the tpcat project.
Switzerland	Passive	Multilateral	Portable Python app	Any	Sneak preview release just spots forged/modified packets
Web Tripwires	Passive	1.5 sided	Javascript embed	HTTP	Must be deployed by webmasters

From the above comparison table, some software required expertise knowledge to perform the test. Or, there are some relatively easy-to-use tools that only capable to gathered insufficient testing evidence.

One of the major reasons we choose Glasnost is because this is a project cooperating with Google to establish a research project to get M-lab working, it aims to help sustain a healthy, innovative Internet."

3.3 Methodology used by BTTEST

URL Link: http://broadband.mpi-sws.org/transparency/bttest.php

1) User execute the Java Applet in the browser
2) The Java Applet will connect to the testing server on port 19980
3) Eight individual test will be run

Test No	Actions	Protocol Used	Port Used
1	Upload	BitTorrent	6881
2	Upload	TCP	6881
3	Download	BitTorrent	6881
4	Download	TCP	6881
5	Upload	BitTorrent	4711
6	Upload	TCP	4711
7	Download	BitTorrent	4711
8	Download	TCP	4711

Three kinds of Deductions can be derived from the eight test results

Failed Test No	Successful Test No	Deductions
1	2	Blocking based on protocol
1	3	Blocking based on direction
1	5	Blocking based on port

Test 1 failed but test 2 is succeeded, the only difference is the protocol used for upload, as a result, it can be deduced that the ISP is blocking based on protocol.

Test 1 failed but test 3 is succeeded, the difference is on the direction of the traffic. Test 1's is uploading while test 3 is downloading. If this kind of result is obtained, it is evidence that the ISP is trying to block the BitTorrent upstream traffic but not the downstream traffic.

Test 1 failed but test 5 succeeded, the difference is on the port being used. The ISP is blocking the well-known BitTorrent port 6881.

3.4 Signal of BitTorrent Blocking used by BTTEST

In this project, we have used the tools BTTEST to detect if the ISP is trying to block, throttle or rate-limit the BitTorrent traffic. BitTorrent traffic is being blocked when the tools encountered the following cases.

a) Both end-points (the User's PC and the servers that hosted the BTTEST) reported that the connection is reset and neither side actively closed the connection.

Scenario: Middlebox generate the TCP RST packet

Server side: Check the packet log for TCP RST packet and make sure the server didn't sent out any TCP RST packet.

Client side: Connection detected will be notified when the JavaApplet on the client side give out a "Connection reset by peer" or "An existing connection was forcibly closed by the remote host" IOException.

b) None of the control TCP traffic before and after the BitTorrent interruption was aborted.

Methodology: TCP control message was sent between the Users's PC and the server that hosted BTTEST. If control TCP traffic was allowed to goes through but not the BitTorrent traffic. This indicates that the ISP is not blocking the traffic based on source or destination IP address but the traffic patterns. In this case, we will have the strong evidence that the ISP was specifically interrupting only BitTorrent traffic.

c) BitTorrent connection is reset immediately after the exchange of BitField message.

Implications: The BitField message is a specific message used in the BTP protocol. Whenever the BitTorrent connection is immediately reset after the exchange of BitField message. This indicated that

the ISP is blocking the BitTorrent traffic.

4.1 Detecting Rate-Limiting BitTorrent Traffic

- If the slowest TCP transfer is still twice as fast as the fastest BitTorrent transfer, the ISP is then suspicious of rate-limiting the BitTorrent traffic.

4.11 Project Strategy & Implementation

1) Install Glasnost in Ubuntu Linux and collect raw data trace that act as the evidence of BT blocking and rate-limiting
- Trace different ISP's BT blocking strategy for each day of the weeks
- Trace different Hong Kong ISP's BT rate-limiting strategy for each day of the weeks

Purpose:
- To determine whether the ISPs is changing their BT blocking policies over the weeks.
- To determine whether ISP is blocking BT traffic or not
- To determine whether ISP is rate-limiting BT traffic

5.1 Network Configuration and system requirement for installing Glasnost

Operation System Used: Ubuntu Linux System (Ubuntu Linux 8.4)

Install Apache2

Purpose:
For Hosting the web server that the java program Glasnost to be run on the client's PC

Installation Method:
apt-get install apache2

Install PHP5 (for PHP script processing ability of the software Glasnost)
http://www.debianadmin.com/apache2-web-server-with-php-support-in-ubuntu.html

sudo apt-get install libapache2-mod-php4 php4-cli php4-common php4-cgi
Next we edit /etc/apache2/apache2.conf file and check the index files are correct

Configure the DNS service web server hosting
Login to www.no-ip.com
Register for a free account and add a host name under the Add Host Tab
Download the no-IP DNS updater client for Linux version to Ubuntu Server

Installation of the No-IP DNS updater
tar –xvf no-ip.tar.gz
./make

Installation of tools for Collecting TCP traces
Install libpcap (version 1.0)
sudo apt-get install flex bison m4
./configure
./make install
install libpcap-dev (for libpcap)

Installation of the Glasnost Server

Download the glasnost server pack from the website

http://broadband.mpi-sws.org/transparency/contribute.html

Uncompress the server pack with the following commands

tar –xvf glasnost-1.2.tgz /var/www/glasnost/

./make

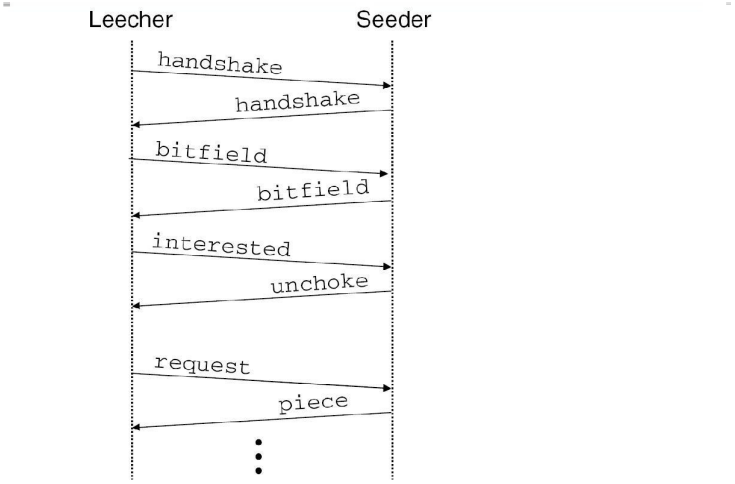
su -c chmod a+s bt_client

chmod 777 logs

The actual operation-flow of the Glasnost

5.2 Interpretation of the Sample Trace result gathered from the Tools

Packet Exchange between Leecher and Seeder



Log file result can be found in the default directory after Glasnost is installed successfully. (*Log Directory:* /var/www/glasnost/log)

For sample log file, please refer to appendix A section for more information.

6.1 Testing Results

Following sub-sections present the testing result from ISPs, they included: HGC, i-cable, PCCW, HK Boardband.

6.11 ISP Name: *HGC*

Is BitTorrent traffic on a well-known BitTorrent port (6682) throttled ?

BT Blocking Testing:
The BitTorrent upload (seeding) worked. Our tool was successful in uploading data using the BitTorrent protocol.
The BitTorrent download worked. Our tool was successful in downloading data using the BitTorrent protocol.

Rate-Limiting

Transfer	Speed TCP	Speed BitTorrent	Conclusion
Download #0	3570 Kbps	2773 Kbps	**No rate limiting**
Download #1	2035 Kbps	2582 Kbps	**No rate limiting**
Upload #0	5342 Kbps	5203 Kbps	**No rate limiting**
Upload #1	5233 Kbps	5305 Kbps	**No rate limiting**
Upload #2	5236 Kbps	5310 Kbps	**No rate limiting**

Is BitTorrent traffic on a non-standard BitTorrent port (10010) throttled?

BT Blocking test
The BitTorrent upload (seeding) worked. Our tool was successful in uploading data using the BitTorrent protocol.
The BitTorrent download worked. Our tool was successful in downloading data using the BitTorrent protocol.

Rate-Limiting

Transfer	Speed TCP	Speed BitTorrent	Conclusion
Upload #0	5455 Kbps	5136 Kbps	**No rate limiting**
Upload #1	5246 Kbps	5346 Kbps	**No rate limiting**
Upload #2	5418 Kbps	5350 Kbps	**No rate limiting**

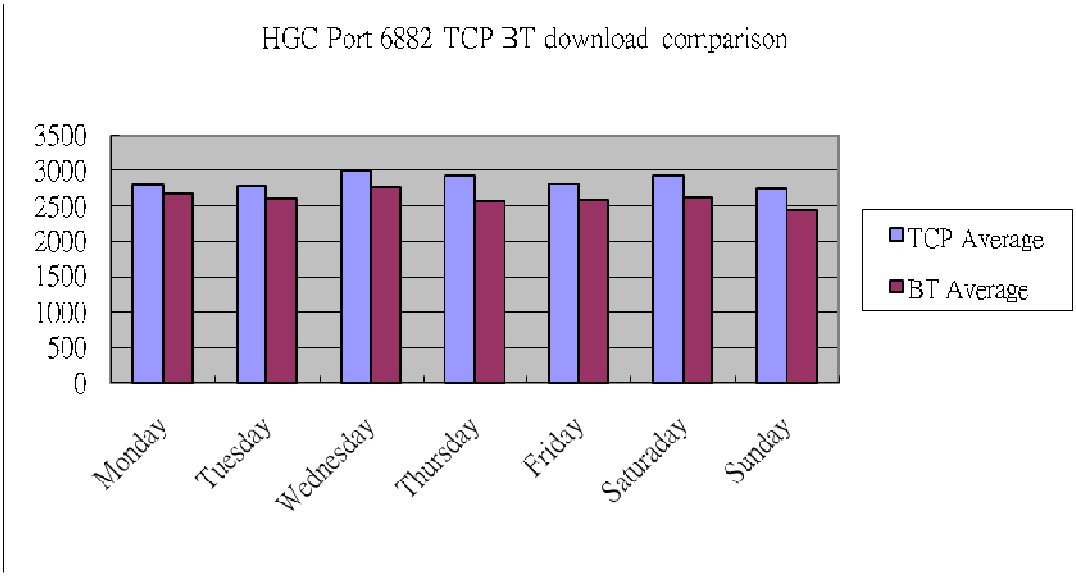
Is TCP traffic on a well-known BitTorrent port (6882) throttled?

Rate-Limiting Test Result

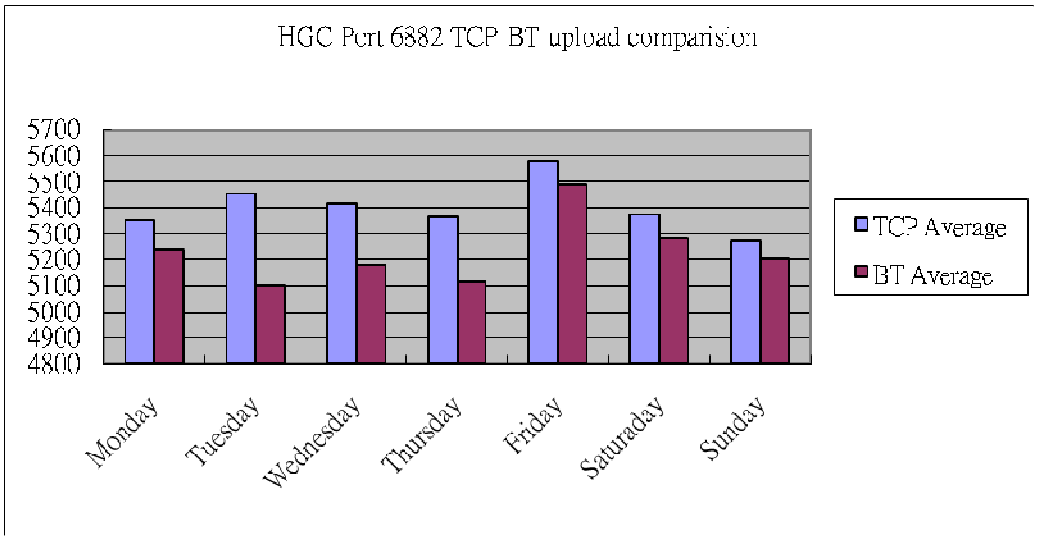
TCP Transfer	BitTorrent port	Non-BitTorrent port	Conclusion
Download #0	5342 Kbps	5455 Kbps	No rate limiting
Download #1	5233 Kbps	5246 Kbps	No rate limiting
Download #2	5236 Kbps	5418 Kbps	No rate limiting

Weekly Testing Result from ISP – HGC

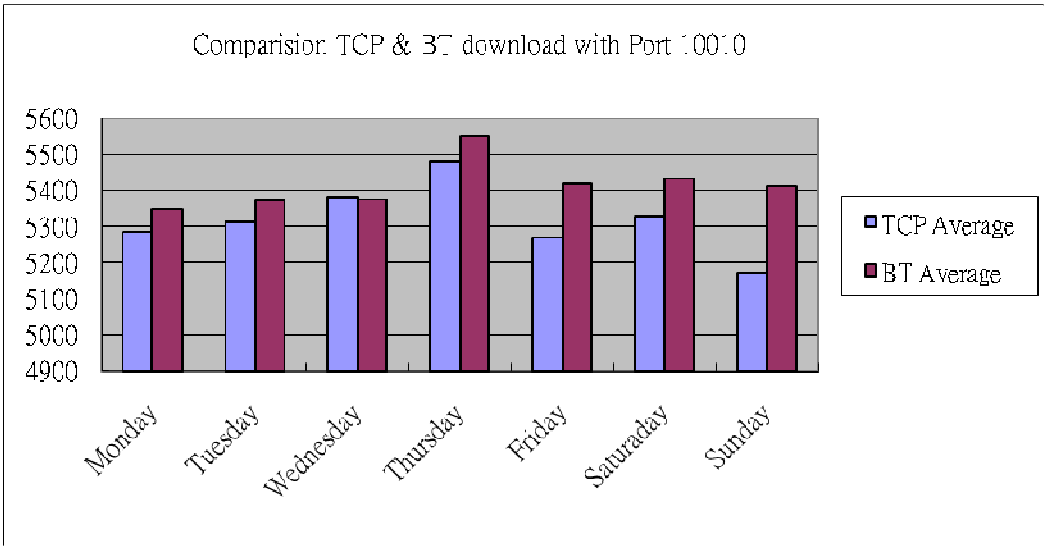
	Port 6882 Weekly test result					
	TCP Download			BT Download		
Weekday	#1	#2	TCP Average	#1	#2	BT Average
Monday	3570	2035	2802.5	2773	2582	2677.5
Tuesday	3462	2125	2793.5	2645	2597	2621
Wednesday	3685	2346	3015.5	2812	2735	2773.5
Thursday	3478	2412	2945	2601	2548	2574.5
Friday	3535	2108	2821.5	2649	2567	2608
Saturday	3524	2348	2936	2677	2579	2628
Sunday	3368	2147	2757.5	2541	2345	2443



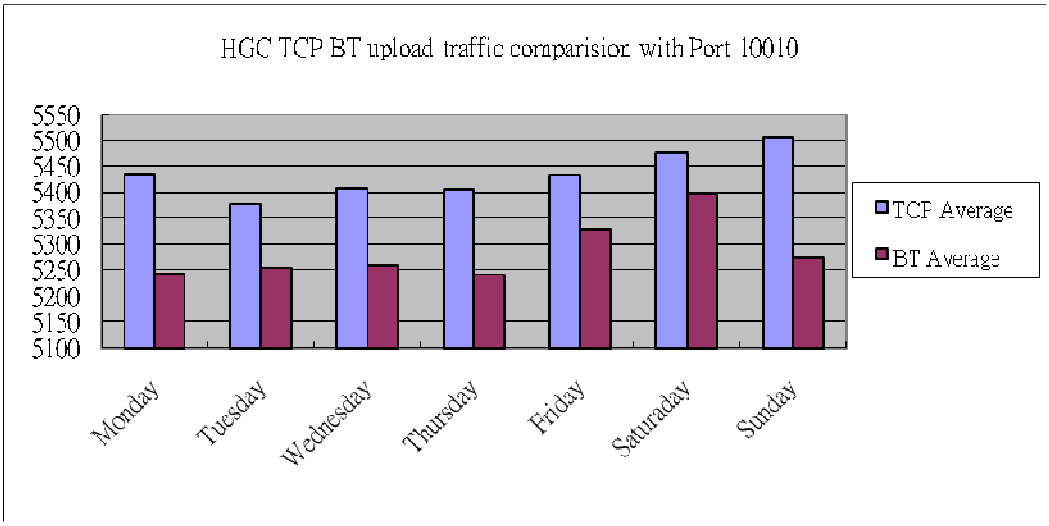
	Port 6882 Weekly test result					
	TCP Upload			BT Upload		
Weekday	#1	#2	TCP Average	#1	#2	BT Average
Monday	5455	5246	5350.5	5136	5346	5241
Tuesday	5564	5347	5455.5	5078	5123	5100.5
Wednesday	5487	5346	5416.5	5147	5215	5181
Thursday	5498	5236	5367	5096	5136	5116
Friday	5521	5642	5581.5	5421	5564	5492.5
Saturday	5327	5426	5376.5	5214	5349	5281.5
Sunday	5213	5337	5275	5168	5237	5202.5



	Port 10010 Weekly test result					
	TCP Download			BT Download		
Weekday	#1	#2	TCP Average	#1	#2	BT Average
Monday	5342	5233	5287.5	5455	5246	5350.5
Tuesday	5378	5246	5312	5532	5216	5374
Wednesday	5421	5349	5385	5412	5341	5376.5
Thursday	5531	5431	5481	5578	5527	5552.5
Friday	5326	5216	5271	5489	5348	5418.5
Saturday	5379	5278	5328.5	5478	5389	5433.5
Sunday	5210	5136	5173	5421	5405	5413

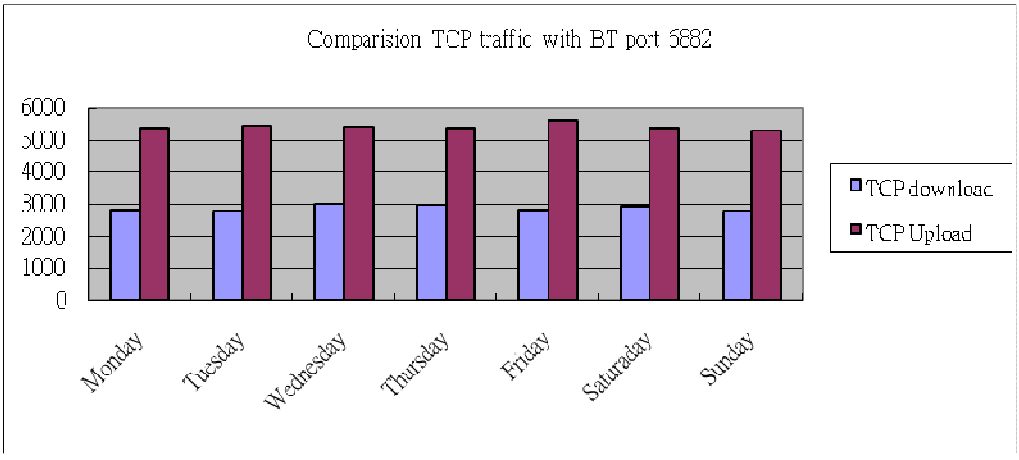


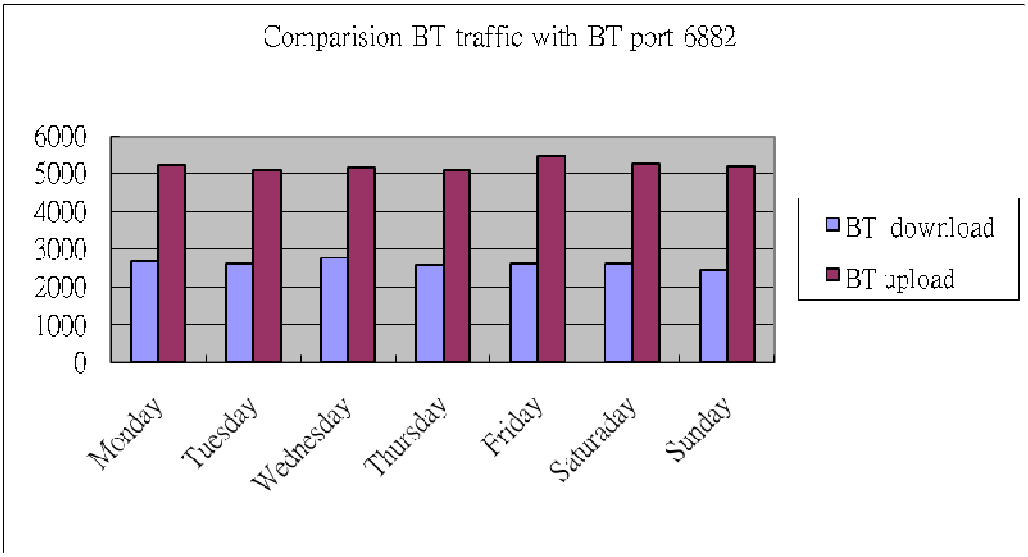
	Port 10010 Weekly test result					
	TCP Upload			BT Upload		
Weekday	#1	#2	TCP Average	#1	#2	BT Average
Monday	5455	5418	5436.5	5136	5350	5243
Tuesday	5432	5326	5379	5167	5346	5256.5
Wednesday	5467	5348	5407.5	5148	5374	5261
Thursday	5435	5376	5405.5	5164	5321	5242.5
Friday	5448	5421	5434.5	5234	5425	5329.5
Saturday	5489	5468	5478.5	5324	5468	5396
Sunday	5534	5478	5506	5218	5329	5273.5



Port 6882

Weekday	TCP download	TCP Upload	BT download	BT upload
Monday	2802.5	5350.5	2677.5	5241
Tuesday	2793.5	5455.5	2621	5100.5
Wednesday	3015.5	5416.5	2773.5	5181
Thursday	2945	5367	2574.5	5116
Friday	2821.5	5581.5	2608	5492.5
Saturday	2936	5376.5	2628	5281.5
Sunday	2757.5	5275	2443	5202.5

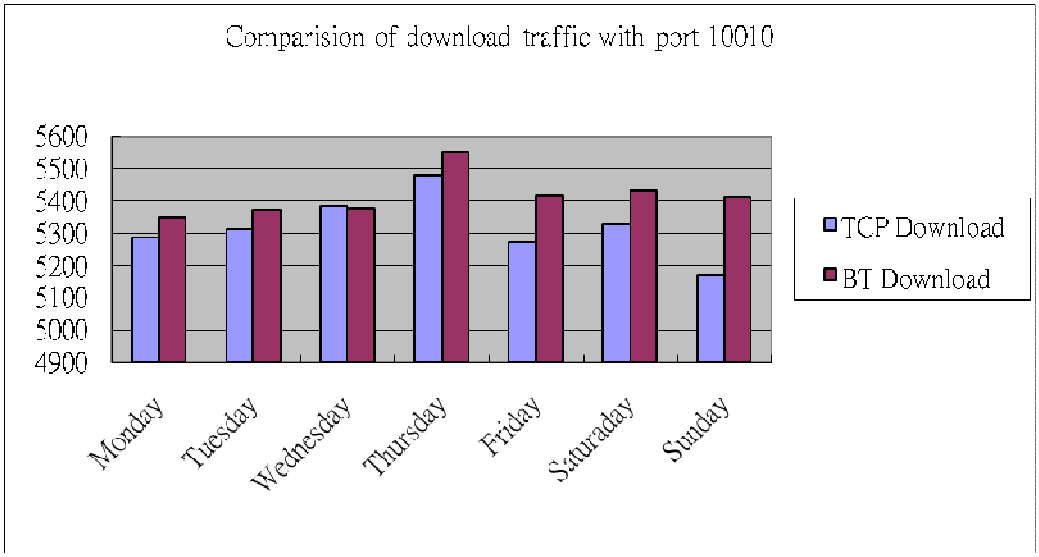


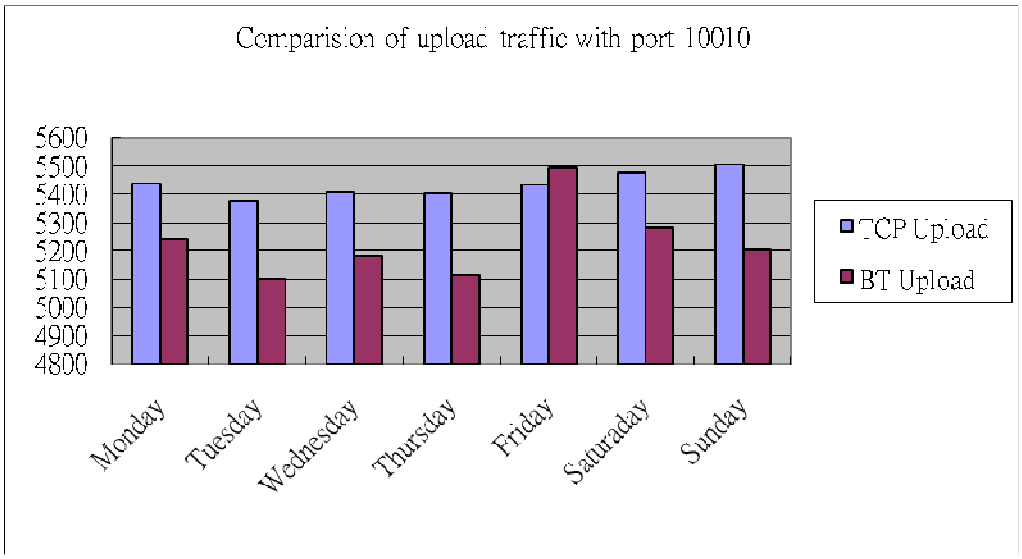


From the above figures, it can be seen that in fact HGC did allow more upload than download.

Port 10010

Weekday	TCP Download	TCP Upload	BT Download	BT Upload
Monday	5287.5	5436.5	5350.5	5241
Tuesday	5312	5379	5374	5100.5
Wednesday	5385	5407.5	5376.5	5181
Thursday	5481	5405.5	5552.5	5116
Friday	5271	5434.5	5418.5	5492.5
Saturday	5328.5	5478.5	5433.5	5281.5
Sunday	5173	5506	5413	5202.5


Comparision of download traffic with port 10010
5600
5500
5400
5300
5200
5100
5000
4900
Monday
Tuesday
Wednesday
Thursday
Friday
Saturaday
Sunday
TCP Download
BT Download


Comparision of upload traffic with port 10010
5600
5500
5400
5300
5200
5100
5000
4900
4800
Monday
Tuesday
Wednesday
Thursday
Friday
Saturaday
Sunday
TCP Upload
BT Upload

Weekday	BT Traffic Blocking (Port 6881)	BT Traffic Blocking (common TCP port 10090)	Rate-Limiting (Upload)	Rate-Limiting (Download)
Monday	No	No	No	No
Tuesday	No	No	No	No
Wednesday	No	No	No	No
Thursday	No	No	No	No
Friday	No	No	No	No
Saturday	No	No	No	No
Sunday	No	No	No	No

6.12 ISP Name: *i-cable*

Is BitTorrent traffic on a well-known BitTorrent port (6682) throttled?

BT Blocking Testing:
The BitTorrent upload (seeding) worked. Our tool was successful in uploading data using the BitTorrent protocol.

There's no indication that your ISP rate limits your BitTorrent uploads. In our tests a TCP upload achieved at least 202 Kbps while a BitTorrent upload achieved at most 262 Kbps. You can find details.

Completed BitTorrent and TCP transfers using the BitTorrent port 6881:

Transfer	Speed TCP	Speed BitTorrent	Conclusion
Download #0	237 Kbps	404 Kbps	**No rate limiting**
Download #1	336 Kbps	351 Kbps	**No rate limiting**
Download #2	357 Kbps	309 Kbps	**No rate limiting**
Upload #0	295 Kbps	186 Kbps	**No rate limiting**
Upload #1	277 Kbps	165 Kbps	**No rate limiting**
Upload #2	202 Kbps	262 Kbps	**No rate limiting**

The BitTorrent download worked. Our tool was successful in downloading data using the BitTorrent protocol.

There's no indication that your ISP rate limits your BitTorrent downloads. In our tests a TCP download achieved at least 237 Kbps while a BitTorrent download achieved at most 404 Kbps. You can find details.

Is BitTorrent traffic on a non-standard BitTorrent port (10009) throttled?

The BitTorrent upload (seeding) worked. Our tool was successful in uploading data using the BitTorrent protocol.

Your ISP possibly rate limits your BitTorrent uploads. In our tests a TCP uploads achieved at least 1144 Kbps while a BitTorrent upload achieved maximal 231 Kbps. You can find details here.

Completed BitTorrent and TCP transfers using the non-BitTorrent port 10009:

Transfer	Speed TCP	Speed BitTorrent	Conclusion
Download #0	335 Kbps	317 Kbps	**No rate limiting**
Download #1	315 Kbps	330 Kbps	**No rate limiting**
Download #2	350 Kbps	365 Kbps	**No rate limiting**
Upload #0	1144 Kbps	178 Kbps	**Potential rate limiting**
Upload #1	1147 Kbps	204 Kbps	**Potential rate limiting**
Upload #2	1147 Kbps	231 Kbps	**Potential rate limiting**

The BitTorrent download worked. Our tool was successful in downloading data using the BitTorrent protocol.

There's no indication that your ISP rate limits your BitTorrent downloads. In our tests a TCP download achieved at least 315 Kbps while a BitTorrent download achieved at most 365 Kbps. You can find details here.

Is TCP traffic on a well-known BitTorrent port (6881) throttled?

There's no indication that your ISP rate limits all downloads at port 6881. In our test, a

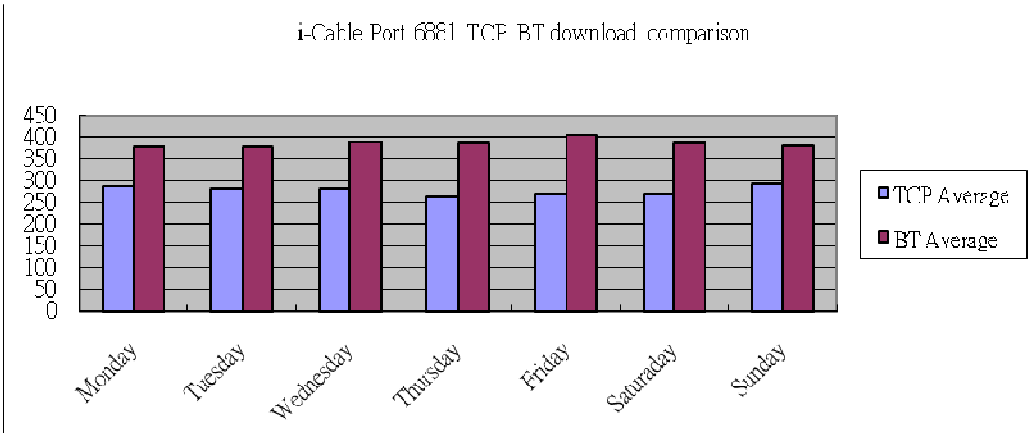
TCP download on a BitTorrent port achieved at least 357 Kbps while a TCP download on a non-BitTorrent port achieved at least 315 Kbps. You can find details here.

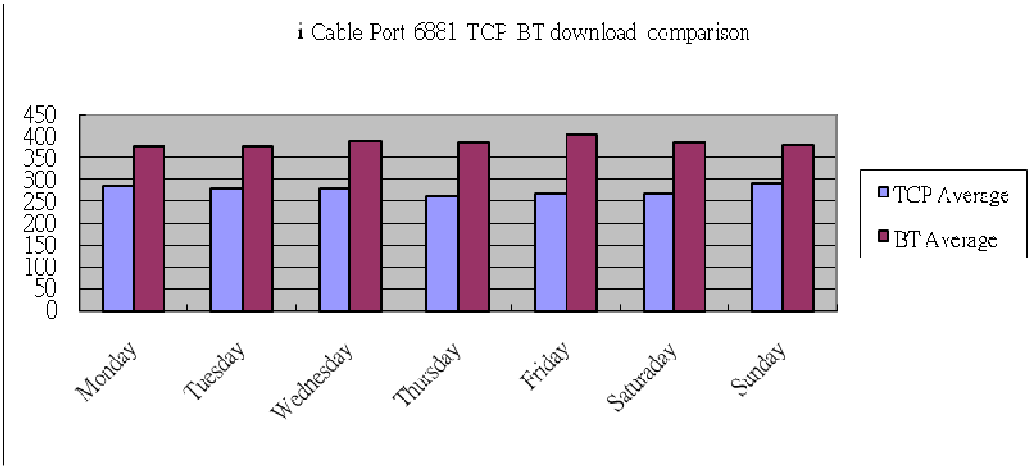
Your ISP possibly rate limits all uploads at port 6881. In our test, a TCP upload on a BitTorrent port achieved at least 314 Kbps while a TCP upload on a non-BitTorrent port achieved at least 1144 Kbps. You can find details here.

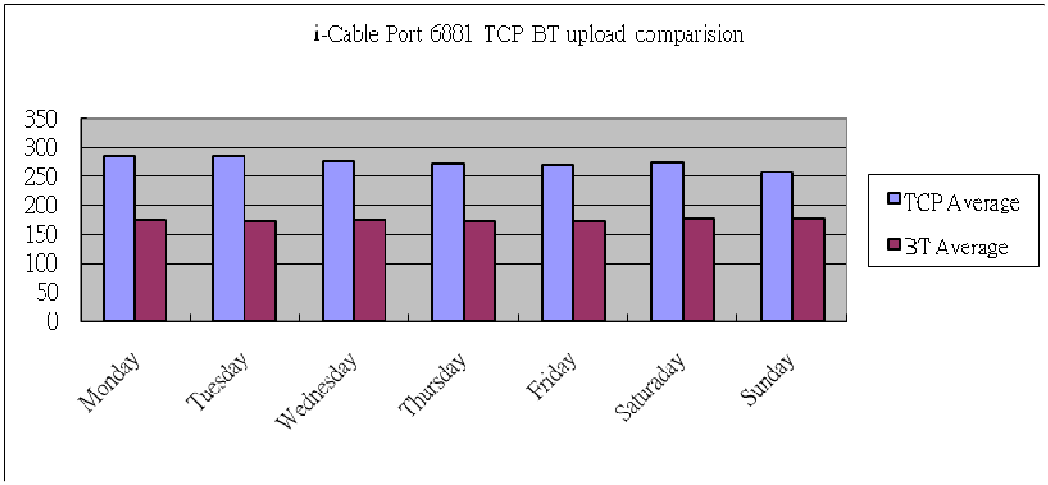
Is TCP traffic on a well-known BitTorrent port (6881) throttled?

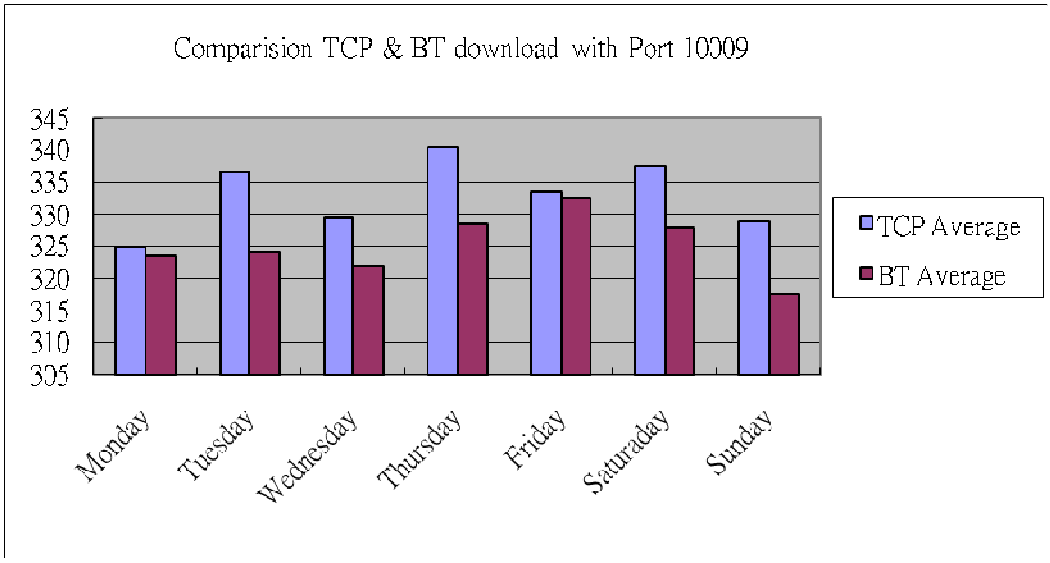
Comparing TCP transfers using the well-know BitTorrent port 6881 and the non-BitTorrent port 10009:

TCP Transfer	BitTorrent port	Non-BitTorrent port	Conclusion
Download #0	237 Kbps	335 Kbps	**No rate limiting**
Download #1	336 Kbps	315 Kbps	**No rate limiting**
Download #2	357 Kbps	350 Kbps	**No rate limiting**
Upload #0	295 Kbps	1144 Kbps	**Potential rate limiting**
Upload #1	277 Kbps	1147 Kbps	**Potential rate limiting**
Upload #2	202 Kbps	1147 Kbps	**Potential rate limiting**

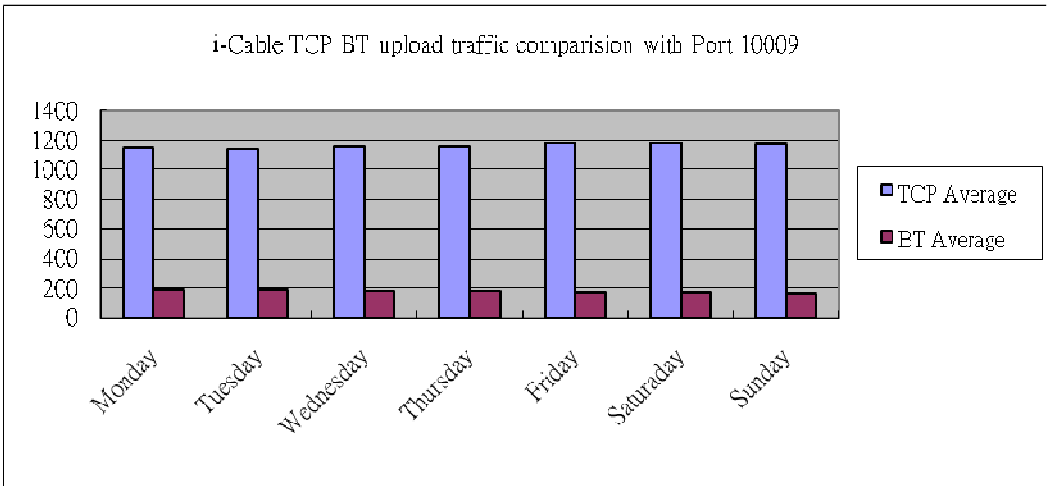

i-Cable Port 6881 TCP BT download comparison
450
400
350
300
250
200
150
100
50
0
Monday
Tuesday
Wednesday
Thursday
Friday
Saturaday
Sunday
TCP Average
BT Average


i Cable Port 6881 TCP BT download comparison
450
400
350
300
250
200
150
100
50
0
Monday
Tuesday
Wednesday
Thursday
Friday
Saturaday
Sunday
TCP Average
BT Average


i-Cable Port 6881 TCP BT upload comparision
350
300
250
200
150
100
50
0
Monday
Tuesday
Wednesday
Thursday
Friday
Saturaday
Sunday
TCP Average
BT Average

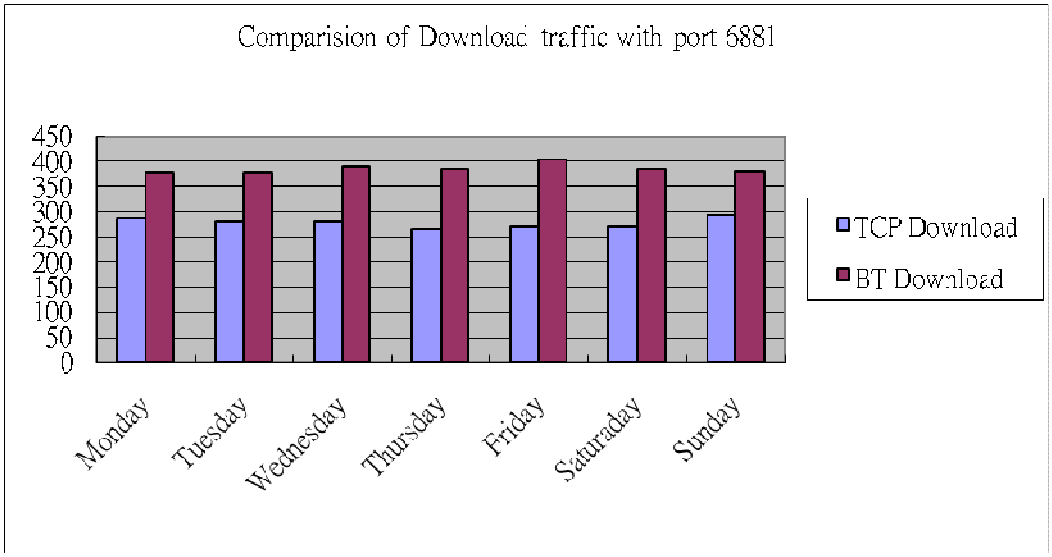

Comparision TCP & BT download with Port 10009
345
340
335
330
325
320
315
310
305
Monday
Tuesday
Wednesday
Thursday
Friday
Saturaday
Sunday
TCP Average
BT Average

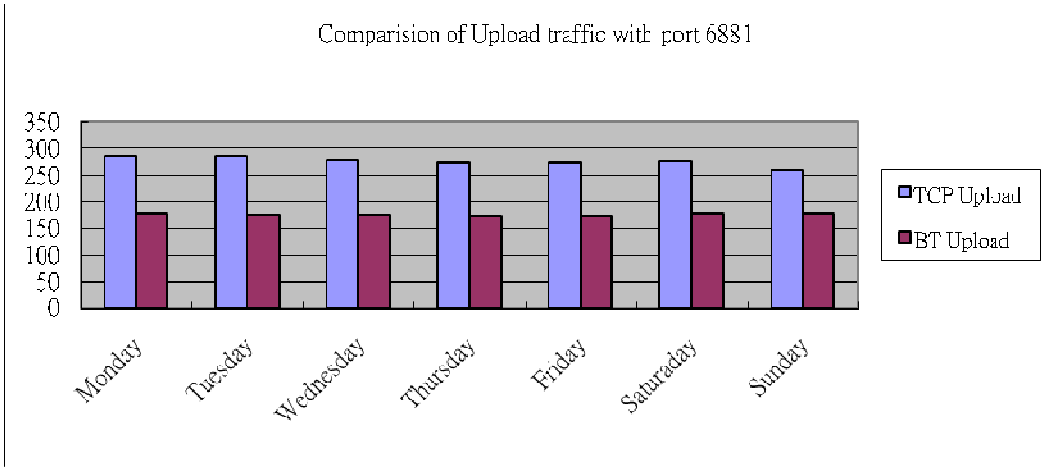
[redacted]

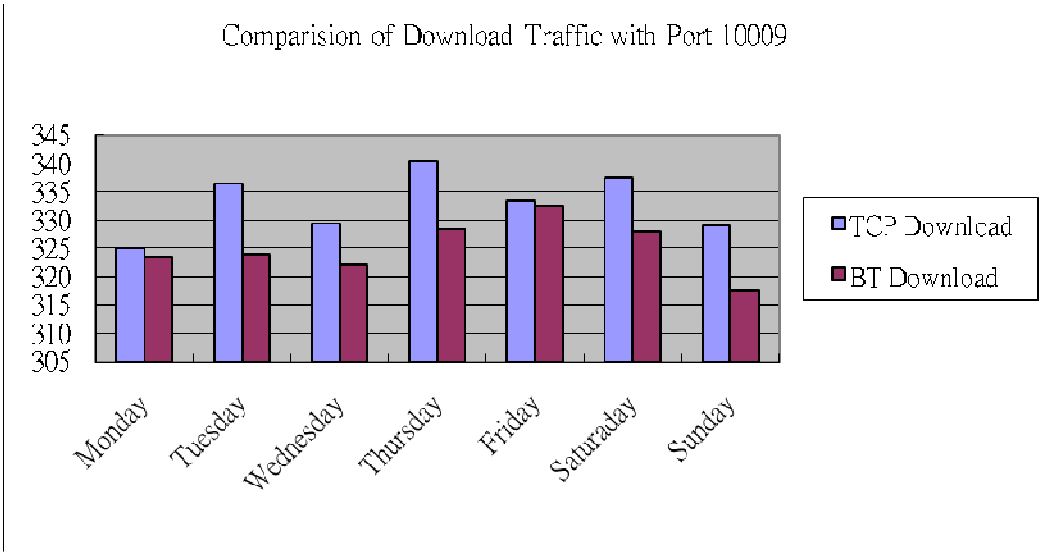


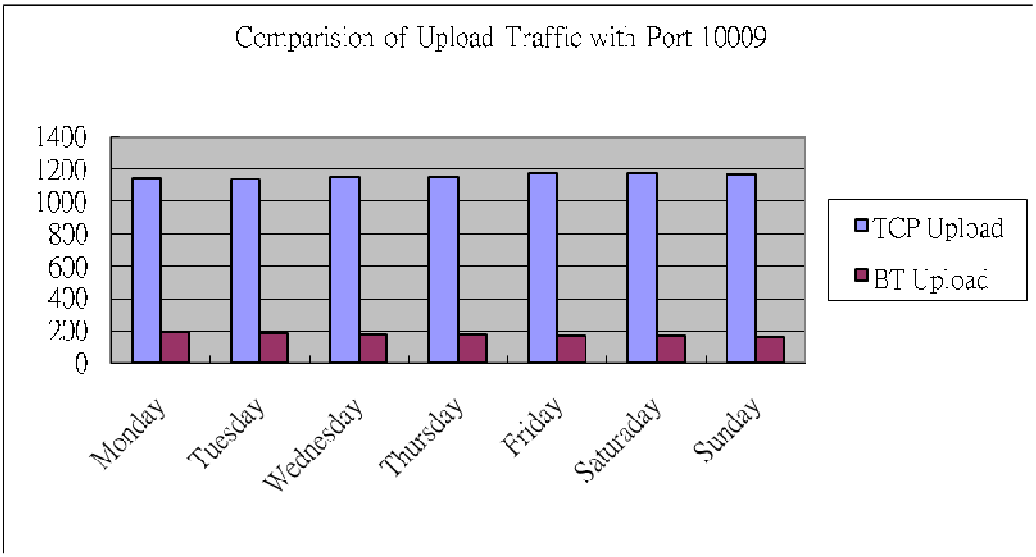
Port 6881

Weekday	TCP Download	TCP Upload	BT Download	BT Upload
Monday	286.5	286	377.5	175.5
Tuesday	280.5	285.5	378.5	173
Wednesday	280.5	277.5	390.5	175
Thursday	265	272	385.5	172
Friday	271.5	271	405	172.5
Saturaday	270.5	274	385.5	176.5
Sunday	294	259	381	176.5


Comparision of Download traffic with port 6881
450
400
350
300
250
200
150
100
50
0
Monday
Tuesday
Wednesday
Thursday
Friday
Saturaday
Sunday
TCP Download
BT Download


Comparision of Upload traffic with port 6881
350
300
250
200
150
100
50
0
Monday
Tuesday
Wednesday
Thursday
Friday
Saturaday
Sunday
TCP Upload
BT Upload


Comparision of Download Traffic with Port 10009
345
340
335
330
325
320
315
310
305
Monday
Tuesday
Wednesday
Thursday
Friday
Saturaday
Sunday
TCP Download
BT Download


Comparision of Upload Traffic with Port 10009
1400
1200
1000
800
600
400
200
0
Monday
Tuesday
Wednesday
Thursday
Friday
Saturaday
Sunday
TCP Upload
BT Upload

ISP Name: i-Cable

Weekday	BT Traffic Blocking (Port 6881)	BT Traffic Blocking (common TCP port 10090)	Rate-Limiting (Upload) (Port 6881)	Rate-Limiting (Upload) (Port 10009)	Rate-Limiting (Download) (Port 6881)	Rate-Limiting (Download) (Port 10009)
Monday	No	No	Yes	Yes	No	No
Tuesday						
Wednesday	No	No	Yes	Yes	No	No
Thursday	No	No	Yes	Yes	No	No
Friday	No	No	Yes	Yes	No	No
Saturday	No	No	Yes	Yes	No	No
Sunday	No	No	Yes	Yes	No	No

6.13 ISP Name: *Smartone*

Is BitTorrent traffic on a well-known BitTorrent port (6882) throttled?

Completed BitTorrent and TCP transfers using the BitTorrent port 6882:

Transfer	Speed TCP	Speed BitTorrent	Conclusion
Download #0	1206 Kbps	1363 Kbps	**No rate limiting**
Download #1	1157 Kbps	1080 Kbps	**No rate limiting**
Download #2	1112 Kbps	1167 Kbps	**No rate limiting**
Upload #0	74 Kbps	86 Kbps	**No rate limiting**
Upload #1	80 Kbps	92 Kbps	**No rate limiting**
Upload #2	92 Kbps	92 Kbps	**No rate limiting**

Is BitTorrent traffic on a non-standard BitTorrent port (10010) throttled?

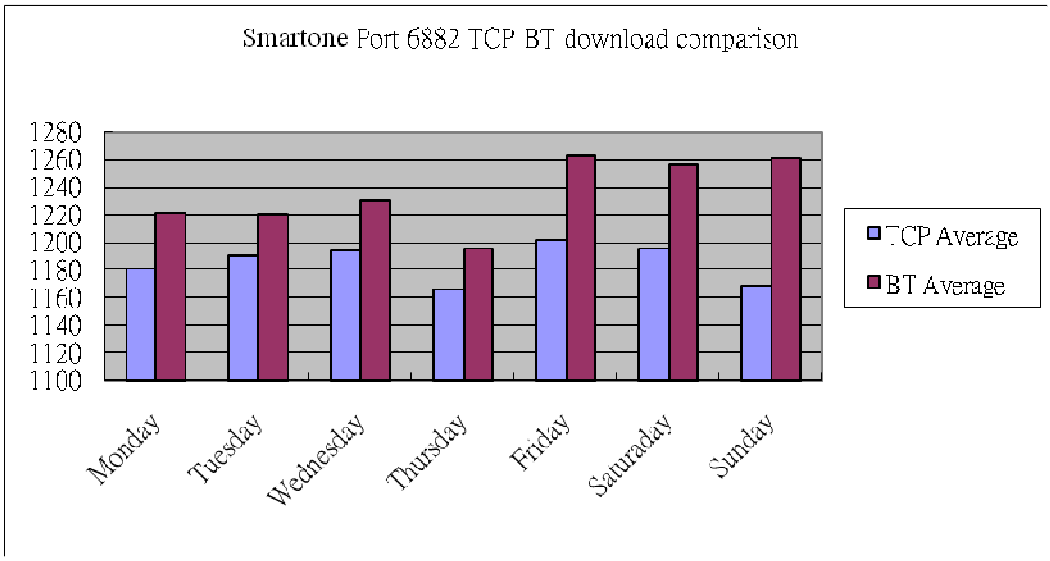
Completed BitTorrent and TCP transfers using the non-BitTorrent port 10010:

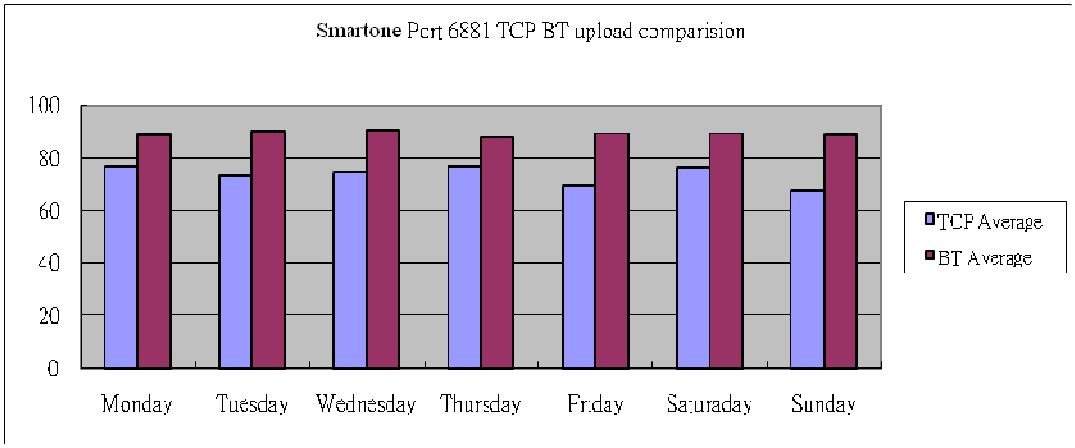
Transfer	Speed TCP	Speed BitTorrent	Conclusion
Download #0	1054 Kbps	948 Kbps	**No rate limiting**
Download #1	1048 Kbps	1158 Kbps	**No rate limiting**
Download #2	1190 Kbps	1153 Kbps	**No rate limiting**
Upload #0	75 Kbps	90 Kbps	**No rate limiting**
Upload #1	81 Kbps	89 Kbps	**No rate limiting**
Upload #2	90 Kbps	80 Kbps	**No rate limiting**

Is TCP traffic on a well-known BitTorrent port (6882) throttled?

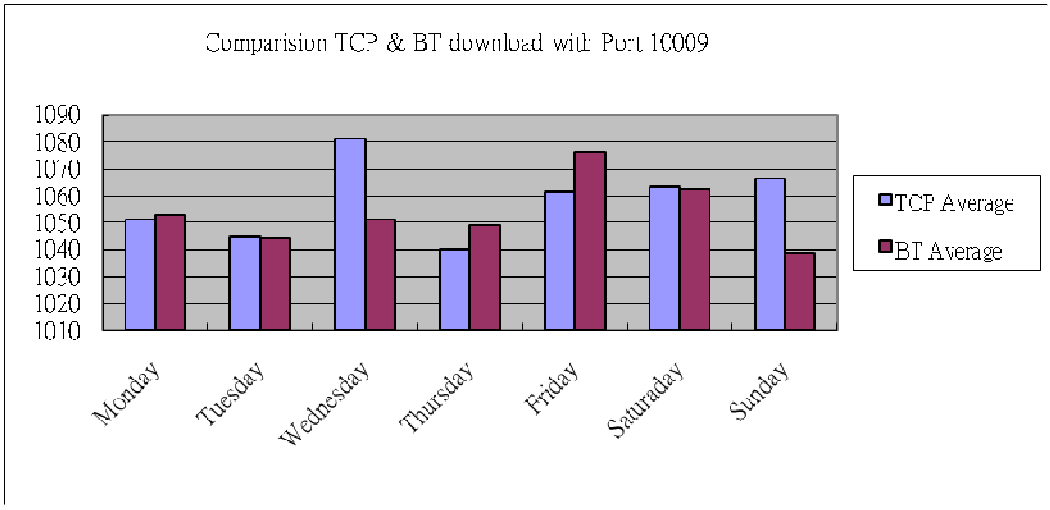
Comparing TCP transfers using the well-know BitTorrent port 6882 and the non-BitTorrent port 10010:

TCP Transfer	BitTorrent port	Non-BitTorrent port	Conclusion
Download #0	1206 Kbps	1054 Kbps	**No rate limiting**
Download #1	1157 Kbps	1048 Kbps	**No rate limiting**
Download #2	1112 Kbps	1190 Kbps	**No rate limiting**
Upload #0	74 Kbps	75 Kbps	**No rate limiting**
Upload #1	80 Kbps	81 Kbps	**No rate limiting**
Upload #2	92 Kbps	90 Kbps	**No rate limiting**

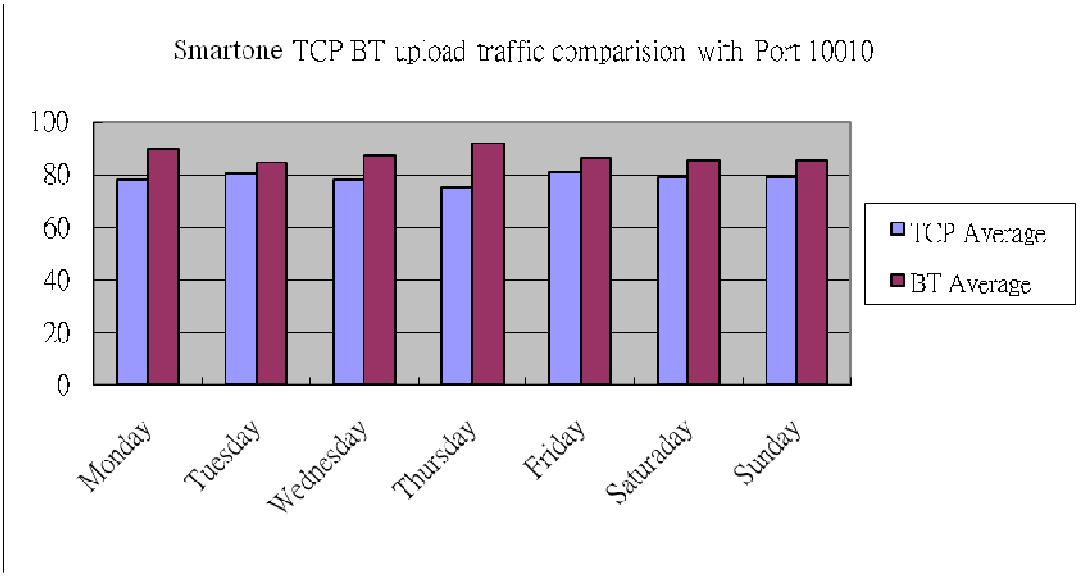

Smartone Port 6882 TCP BT download comparison
1280
1260
1240
1220
1200
1180
1160
1140
1120
1100
Monday
Tuesday
Wednesday
Thursday
Friday
Saturaday
Sunday
TCP Average
BT Average


Smartone Port 6881 TCP BT upload comparision
100
80
60
40
20
0
Monday
Tuesday
Wednesday
Thursday
Friday
Saturaday
Sunday
TCP Average
BT Average

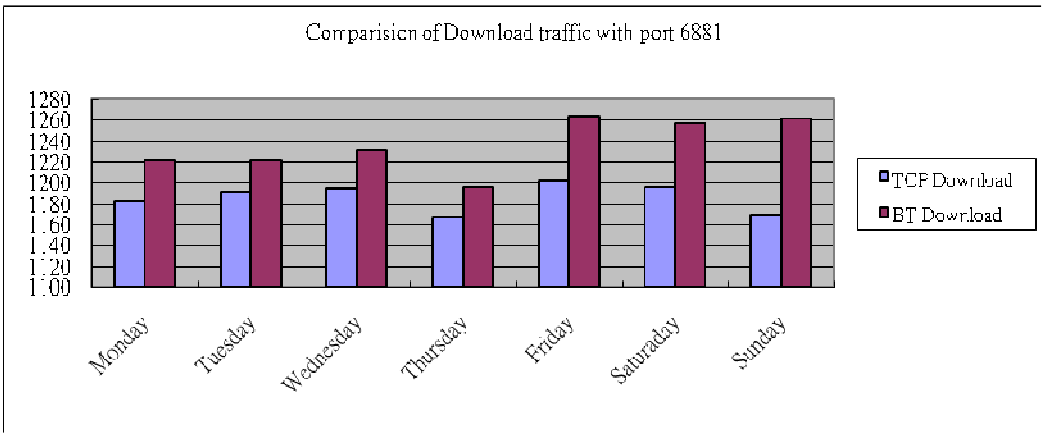
[redacted]

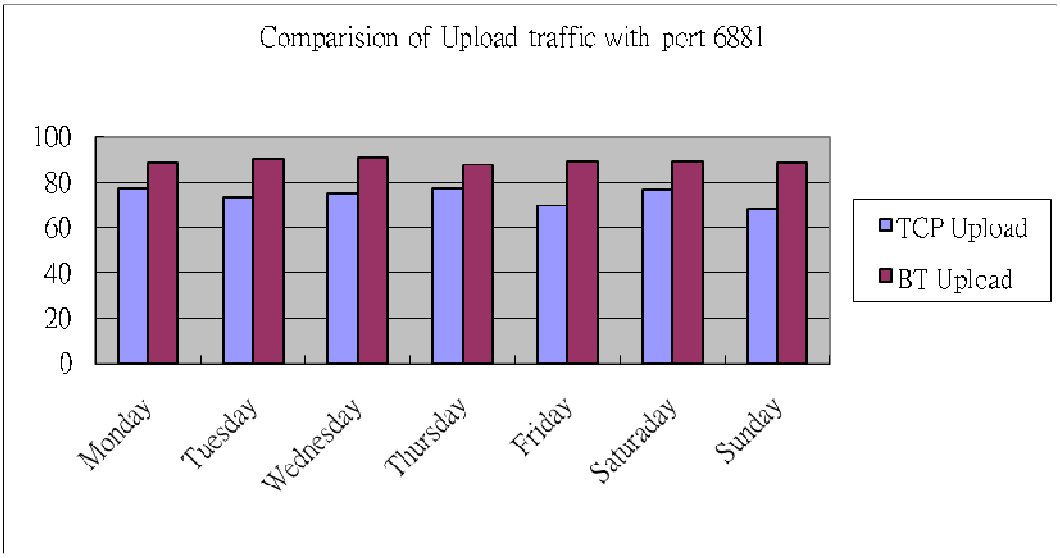

Comparision TCP & BT download with Port 10009
1090
1080
1070
1060
1050
1040
1030
1020
1010
Monday
Tuesday
Wednesday
Thursday
Friday
Saturaday
Sunday
TCP Average
BT Average

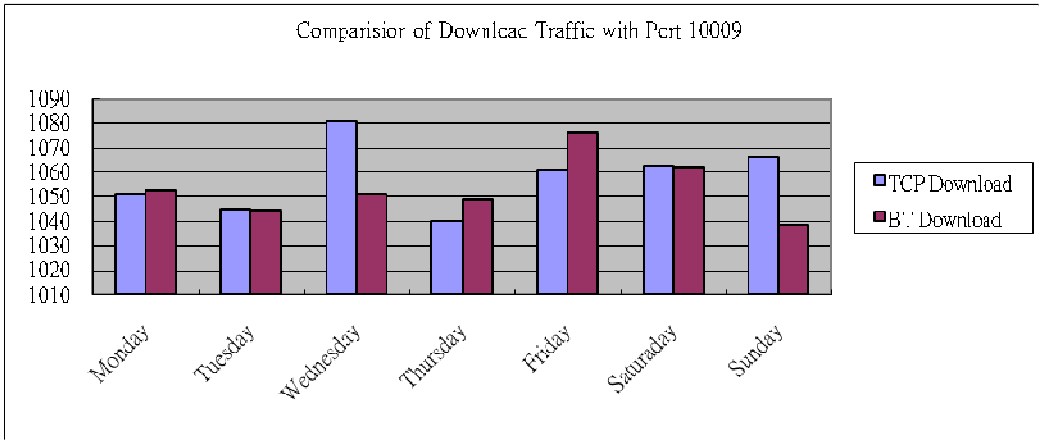
[redacted]

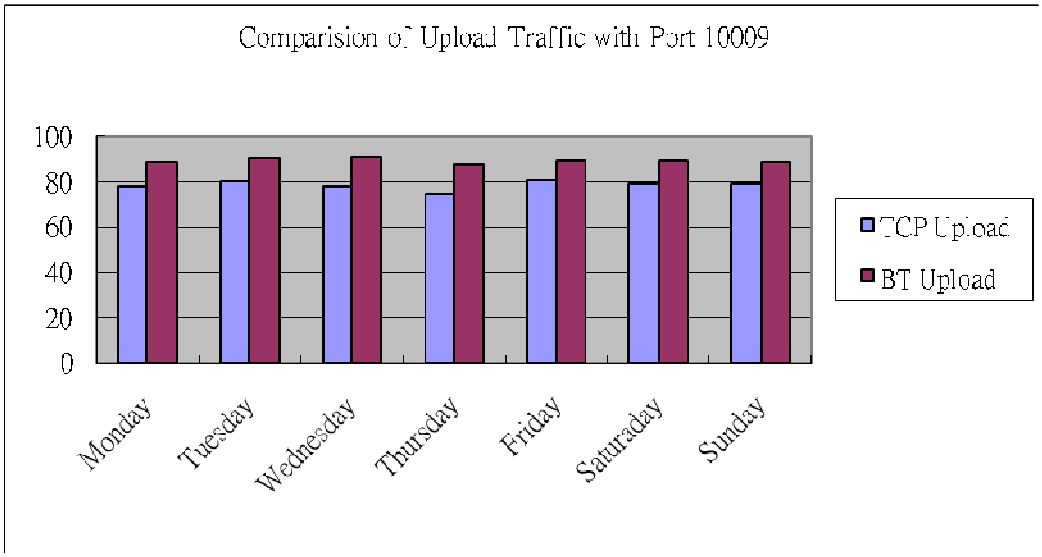

Smartone TCP BT upload traffic comparision with Port 10010
100
80
60
40
20
0
Monday
Tuesday
Wednesday
Thursday
Friday
Saturaday
Sunday
TCP Average
BT Average

[redacted]


Comparision of Download traffic with port 6881
1280
1260
1240
1220
1200
1180
1160
1140
1120
1100
Monday
Tuesday
Wednesday
Thursday
Friday
Saturaday
Sunday
TCP Download
BT Download


Comparision of Upload traffic with port 6881
100
80
60
40
20
0
Monday
Tuesday
Wednesday
Thursday
Friday
Saturaday
Sunday
TCP Upload
BT Upload


Comparision of Download Traffic with Port 10009
1090
1080
1070
1060
1050
1040
1030
1020
1010
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
TCP Download
BT Download


Comparision of Upload Traffic with Port 10009
100
80
60
40
20
0
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
TCP Upload
BT Upload

ISP Name: Smartone

Weekday	BT Traffic Blocking (Port 6881)	BT Traffic Blocking (common TCP port 10090)	Rate-Limiting (Upload) (Port 6881)	Rate-Limiting (Upload) (Port 10009)	Rate-Limiting (Download) (Port 6881)	Rate-Limiting (Download) (Port 10009)
Monday	No	No	No	No	No	No
Tuesday	No	No	No	No	No	No
Wednesday	No	No	No	No	No	No
Thursday	No	No	No	No	No	No
Friday	No	No	No	No	No	No
Saturday	No	No	No	No	No	No
Sunday	No	No	No	No	No	No

6.14 ISP Name: *HK Broadband*

Is BitTorrent traffic on a well-known BitTorrent port (6881) throttled?

- There was a problem with the BitTorrent upload (seeding). No connection on port 6881 was ever possible. It is likely that a traffic shaper or a local firewall blocks this port for all connections.

- There was a problem with the BitTorrent download. No connection on port 6881 was ever possible. It is likely that a traffic shaper or a local firewall blocks this port for all connections.

Is BitTorrent traffic on a non-standard BitTorrent port (10009) throttled?

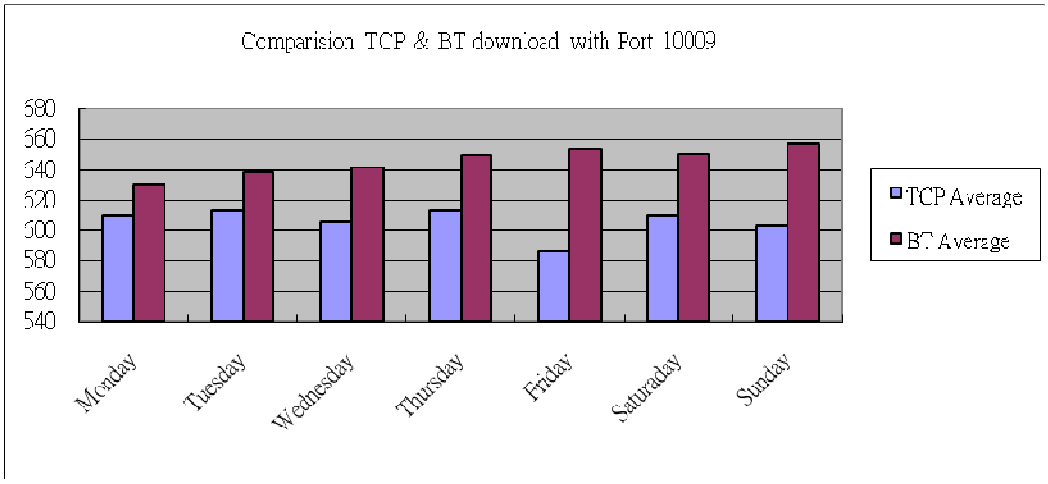
The BitTorrent upload (seeding) worked. Our tool was successful in uploading data using the BitTorrent protocol.

- There's no indication that your ISP rate limits your BitTorrent uploads. In our tests a TCP upload achieved at least 1049 Kbps while a BitTorrent download

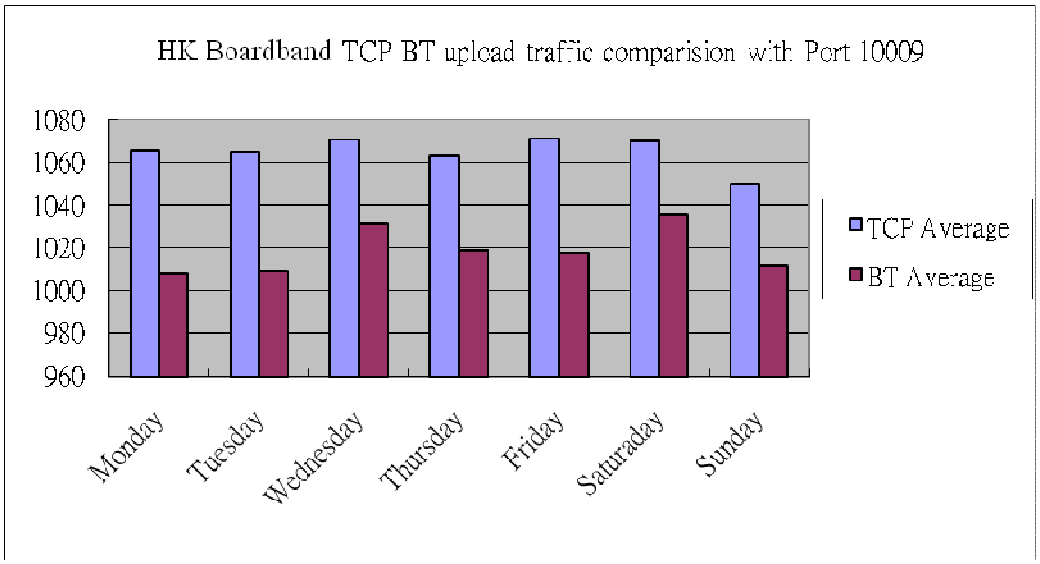
achieved at most 1076 Kbps. You can find details here.

Transfer Speed	TCP Speed	BitTorrent Speed	Conclusion
Download #0	629Kbps	637Kbps	**No rate limiting**
Download #1	590Kbps	623Kbps	**No rate limiting**
Download #2	472Kbps	623Kbps	**No rate limiting**
Upload #0	1082Kbps	941Kbps	**No rate limiting**
Upload #1	1049Kbps	1076Kbps	**No rate limiting**
Upload#2	1078Kbps	1073Kbps	**No rate limiting**

- The BitTorrent download worked. Our tool was successful in downloading data using the BitTorrent protocol.
- There's no indication that your ISP rate limits your BitTorrent downloads. In our tests a TCP download achieved at least 472 Kbps while a BitTorrent download achieved at most 637 Kbps. You can find details here.

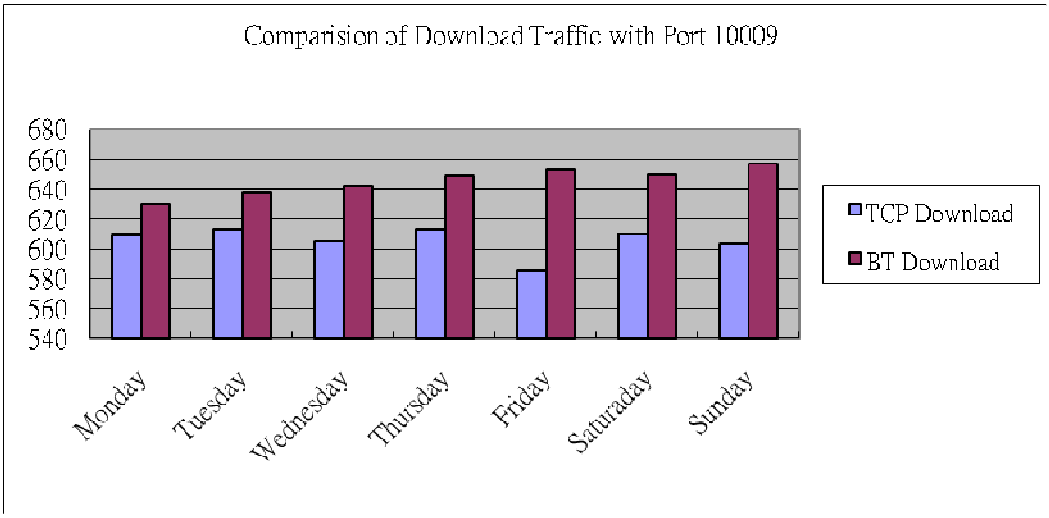


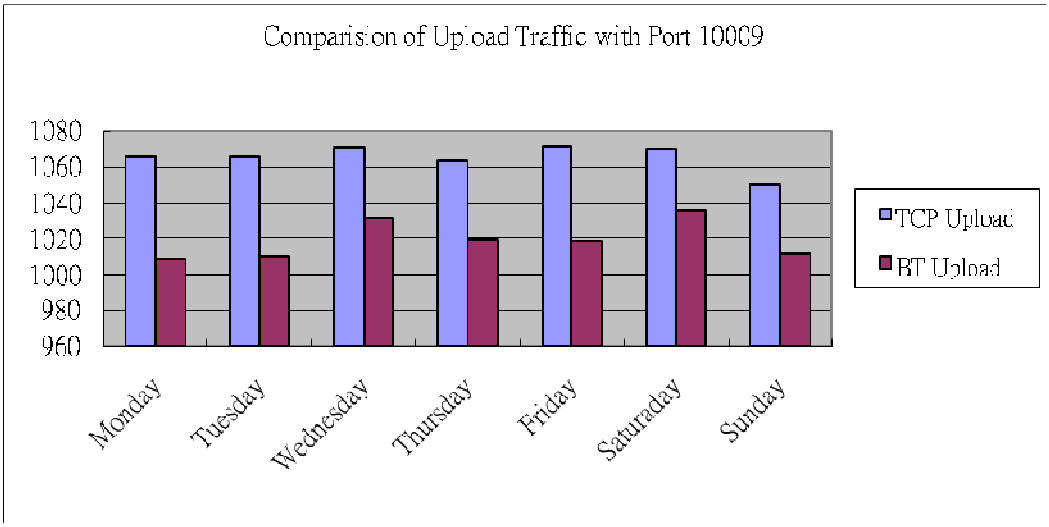
	Port 10009 Weekly test result					
	TCP Upload			BT Upload		
Weekday	#1	#2	TCP Average	#1	#2	BT Average
Monday	1082	1049	1065.5	941	1076	1008.5
Tuesday	1076	1054	1065	945	1074	1009.5
Wednesday	1085	1056	1070.5	989	1073	1031
Thursday	1068	1058	1063	962	1076	1019
Friday	1079	1063	1071	964	1072	1018
Saturaday	1064	1076	1070	973	1098	1035.5
Sunday	1073	1026	1049.5	981	1042	1011.5



Port 10009

Weekday	TCP Download	TCP Upload	BT Download	BT Upload
Monday	609.5	1065.5	630	1008.5
Tuesday	613	1065	638	1009.5
Wednesday	605.5	1070.5	641.5	1031
Thursday	613	1063	649	1019
Friday	585.5	1071	653	1018
Saturaday	610	1070	650	1035.5
Sunday	603.5	1049.5	657	1011.5


Comparision of Download Traffic with Port 10009
680
660
640
620
600
580
560
540
Monday
Tuesday
Wednesday
Thursday
Friday
Saturaday
Sunday
TCP Download
BT Download


Comparision of Upload Traffic with Port 10009
1080
1060
1040
1020
1000
980
960
Monday
Tuesday
Wednesday
Thursday
Friday
Saturaday
Sunday
TCP Upload
BT Upload

Conclusion drawn for ISP HK-Broadband

Weekday	BT Traffic Blocking (Port 6881)	BT Traffic Blocking (common TCP port 10090)	Rate-Limiting (Upload) (Port 6881)	Rate-Limiting (Upload) (Port 10009)	Rate-Limiting (Download) (Port 6881)	Rate-Limiting (Download) (Port 10009)
Monday	Yes	No	N/A	No	N/A	No
Tuesday	Yes	No	N/A	No	N/A	No
Wednesday	Yes	No	N/A	No	N/A	No
Thursday	Yes	No	N/A	No	N/A	No
Friday	Yes	No	N/A	No	N/A	No
Saturday	Yes	No	N/A	No	N/A	No
Sunday	Yes	No	N/A	No	N/A	No

6.15 ISP Name: *PCCW*

Is BitTorrent traffic on a well-known BitTorrent port (6881) throttled?

Completed BitTorrent and TCP transfers using the BitTorrent port 6881:

Transfer Speed	TCP Speed	BitTorrent Speed	Conclusion
Download #0	38Kbps	18Kbps	**Potential Rate limiting**
Download #1	45Kbps	27Kbps	**No rate limiting**
Download #2	45Kbps	52Kbps	**No rate limiting**
Upload #0	100Kbps	195Kbps	**No rate limiting**
Upload #1	161Kbps	145Kbps	**No rate limiting**
Upload#2	179Kbps	191Kbps	**No rate limiting**

Is BitTorrent traffic on a non-standard BitTorrent port (10009) throttled?

Completed BitTorrent and TCP transfers using the non-BitTorrent port 10009:

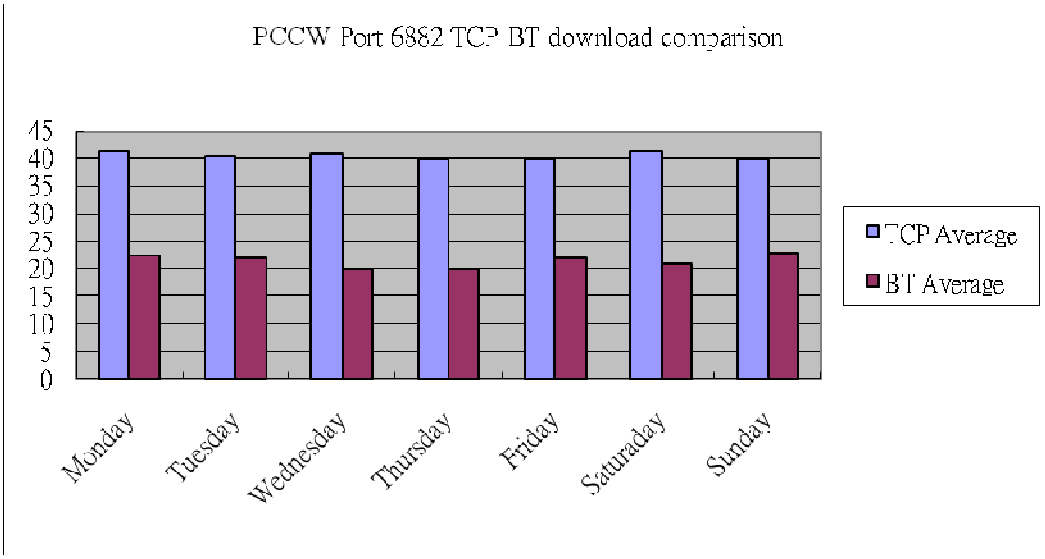
Transfer Speed	TCP Speed	BitTorrent Speed	Conclusion
Download #0	435Kbps	65Kbps	**Potential rate limiting**
Download #1	419Kbps	58Kbps	**Potential rate limiting**
Download #2	432Kbps	81Kbps	**Potential rate limiting**
Upload #0	386Kbps	175Kbps	**Potential rate limiting**
Upload #1	392Kbps	164Kbps	**Potential rate limiting**
Upload#2	253Kbps	172Kbps	**No rate limiting**

Is TCP traffic on a well-known BitTorrent port (6881) throttled?

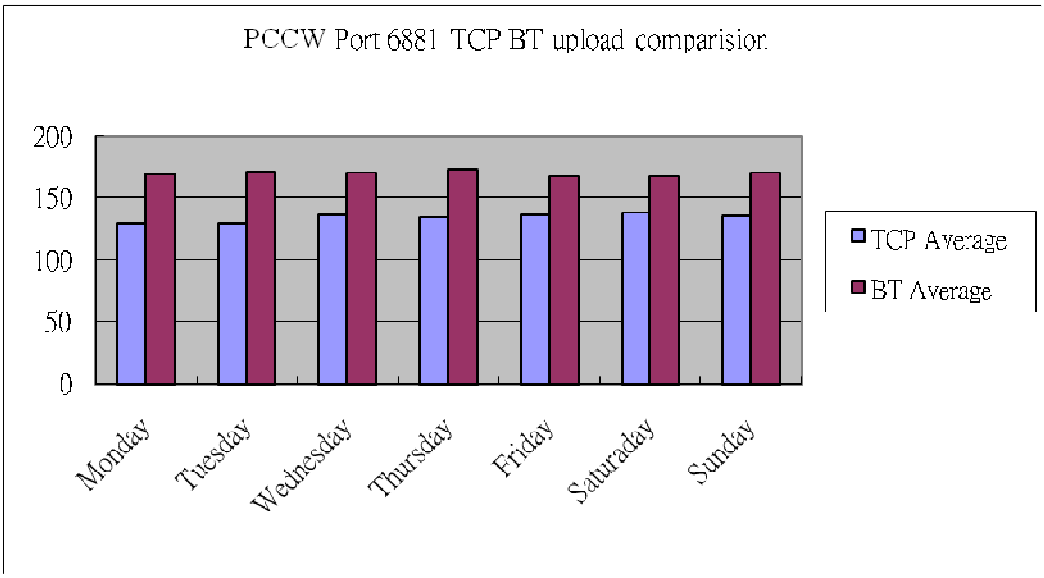
Comparing TCP transfers using the well-know BitTorrent port 6881 and the non-BitTorrent port 10009:

Transfer Speed	BitTorrent Port	Non BitTorrent Port	Conclusion
Download #0	38Kbps	435Kbps	Potential rate limiting
Download #1	45Kbps	419Kbps	Potential rate limiting
Download #2	45Kbps	432Kbps	Potential rate limiting
Upload #0	100Kbps	386Kbps	Potential rate limiting
Upload #1	161Kbps	392Kbps	Potential rate limiting
Upload#2	179Kbps	253Kbps	No rate limiting

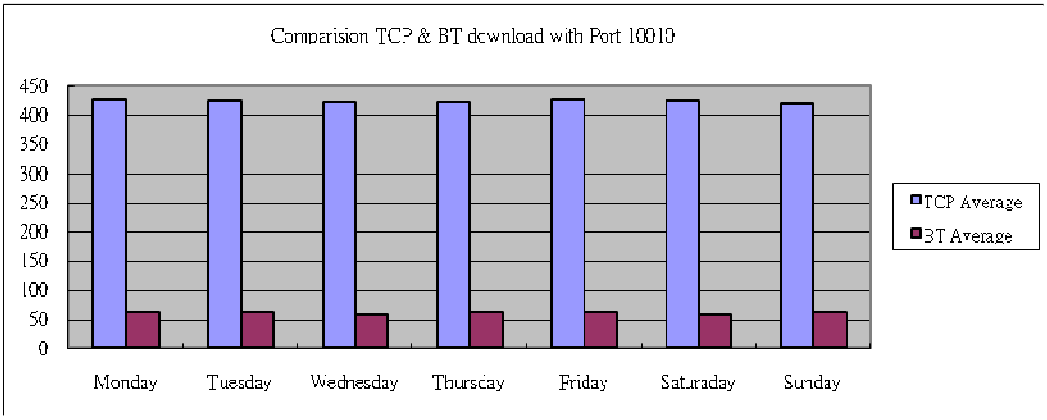
	Port 6881 Weekly test result					
	TCP Download			BT Download		
Weekday	#1	#2	TCP Average	#1	#2	BT Average
Monday	38	45	41.5	18	27	22.5
Tuesday	34	47	40.5	18	26	22
Wednesday	36	46	41	16	24	20
Thursday	38	42	40	17	23	20
Friday	37	43	40	16	28	22
Saturaday	39	44	41.5	17	25	21
Sunday	32	48	40	19	27	23



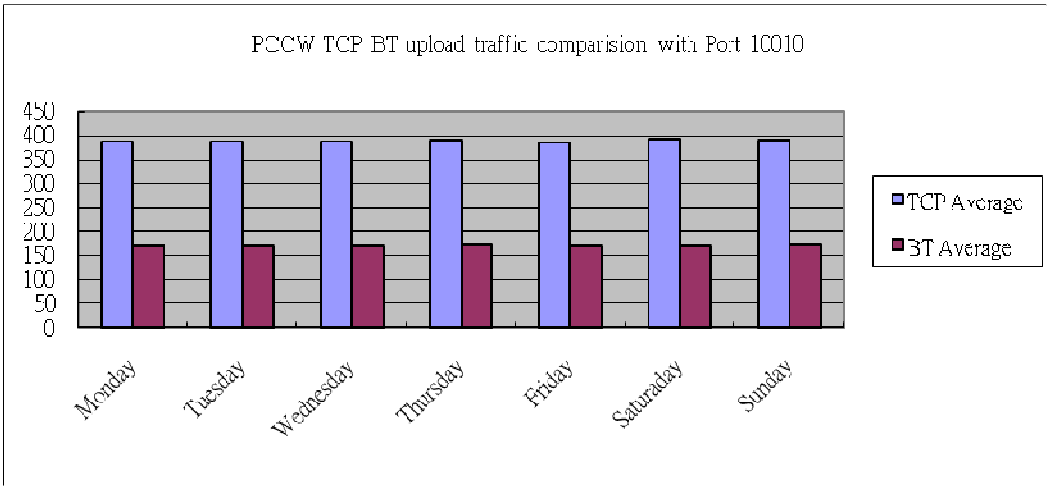
	Port 6882 Weekly test result					
	TCP Upload			BT Upload		
Weekday	#1	#2	TCP Average	#1	#2	BT Average
Monday	100	161	130.5	195	145	170
Tuesday	98	162	130	197	147	172
Wednesday	112	161	136.5	196	146	171
Thursday	105	164	134.5	198	149	173.5
Friday	107	167	137	194	142	168
Saturaday	108	169	138.5	192	143	167.5
Sunday	109	162	135.5	193	148	170.5



	Port 10009 Weekly test result					
	TCP Download			BT Download		
Weekday	#1	#2	TCP Average	#1	#2	BT Average
Monday	435	419	427	65	58	61.5
Tuesday	432	419	425.5	67	57	62
Wednesday	431	416	423.5	64	52	58
Thursday	428	417	422.5	68	54	61
Friday	436	418	427	69	56	62.5
Saturaday	437	413	425	62	53	57.5
Sunday	428	415	421.5	64	58	61

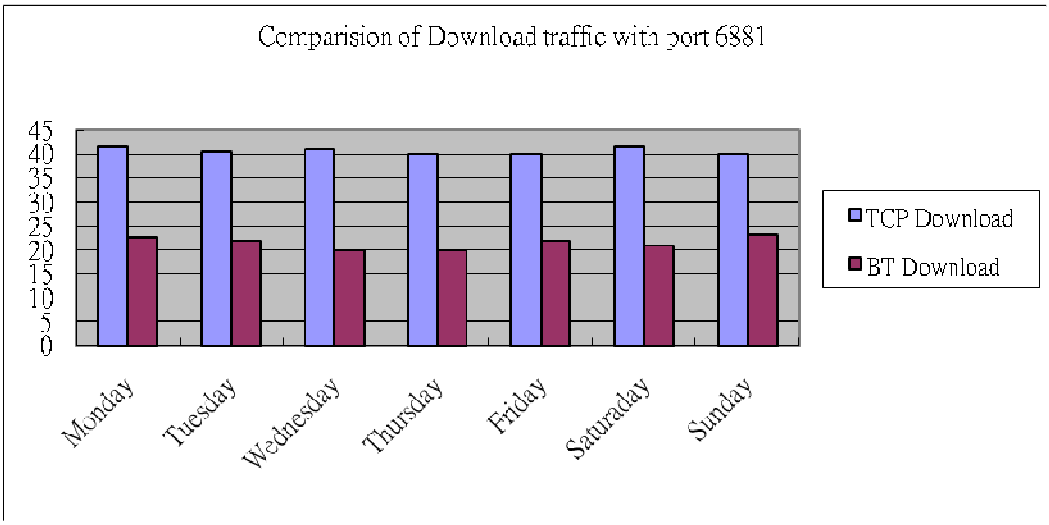


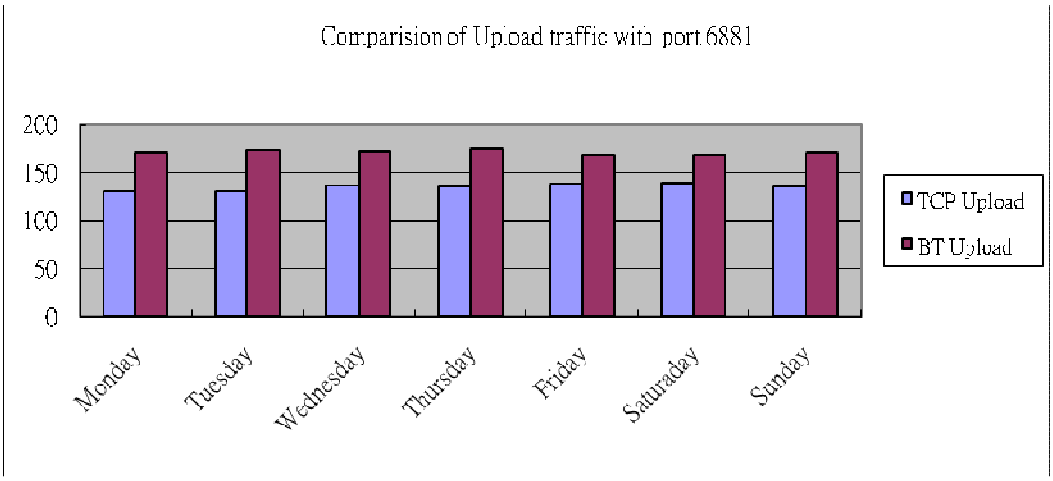
	Port 10009 Weekly test result					
	TCP Upload			BT Upload		
Weekday	#1	#2	TCP Average	#1	#2	BT Average
Monday	384	392	388	175	164	169.5
Tuesday	384	391	387.5	174	164	169
Wednesday	382	392	387	176	162	169
Thursday	386	394	390	178	165	171.5
Friday	375	395	385	173	168	170.5
Saturaday	386	397	391.5	175	167	171
Sunday	384	396	390	179	167	173



Port 6881

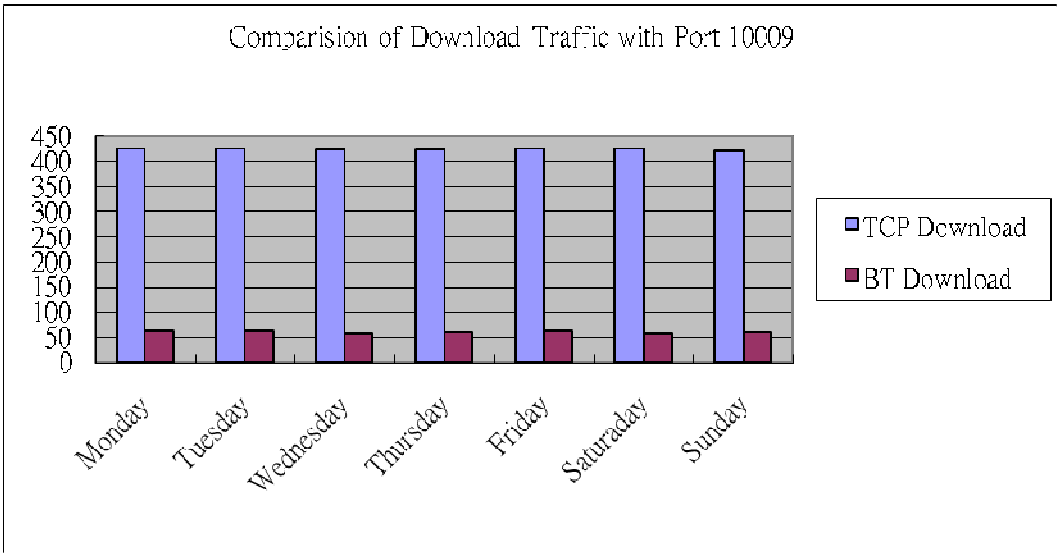
Weekday	TCP Download	TCP Upload	BT Download	BT Upload
Monday	41.5	130.5	22.5	170
Tuesday	40.5	130	22	172
Wednesday	41	136.5	20	171
Thursday	40	134.5	20	173.5
Friday	40	137	22	168
Saturaday	41.5	138.5	21	167.5
Sunday	40	135.5	23	170.5

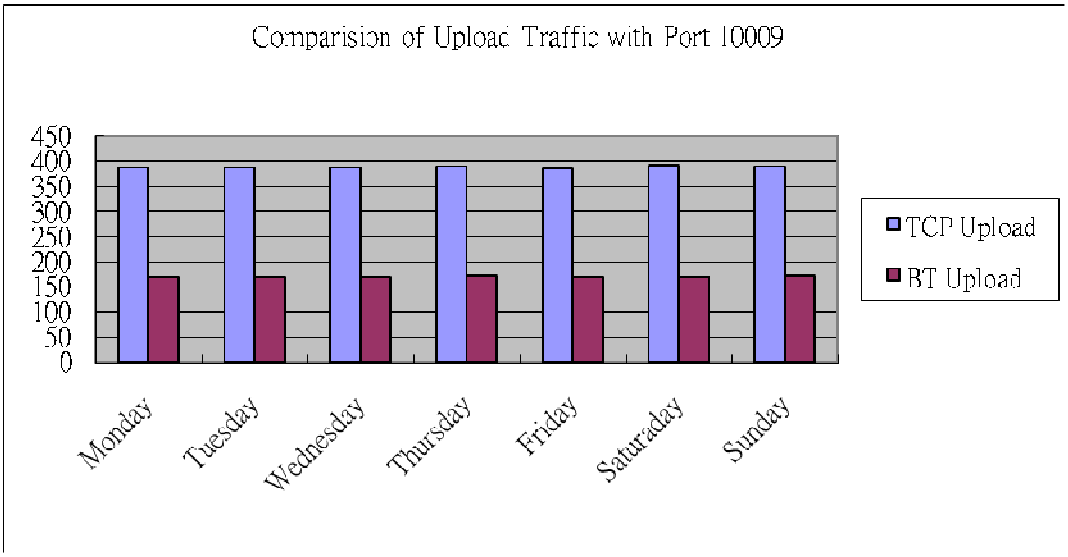




Port 10009

Weekday	TCP Download	TCP Upload	BT Download	BT Upload
Monday	427	388	61.5	169.5
Tuesday	425.5	387.5	62	169
Wednesday	423.5	387	58	169
Thursday	422.5	390	61	171.5
Friday	427	385	62.5	170.5
Saturaday	425	391.5	57.5	171
Sunday	421.5	390	61	173


Comparision of Download Traffic with Port 10009
450
400
350
300
250
200
150
100
50
0
Monday
Tuesday
Wednesday
Thursday
Friday
Saturaday
Sunday
TCP Download
BT Download


Comparision of Upload Traffic with Port 10009
450
400
350
300
250
200
150
100
50
0
Monday
Tuesday
Wednesday
Thursday
Friday
Saturaday
Sunday
TCP Upload
BT Upload

Conclusion drawn for ISP: PCCW

Weekday	BT Traffic Blocking (Port 6881)	BT Traffic Blocking (common TCP port 10090)	Rate-Limiting (Upload) (Port 6881)	Rate-Limiting (Upload) (Port 10009)	Rate-Limiting (Download) (Port 6881)	Rate-Limiting (Download) (Port 10009)
Monday	No	No	Yes	Yes	Yes	Yes
Tuesday	No	No	Yes	Yes	Yes	Yes
Wednesday	No	No	Yes	Yes	Yes	Yes
Thursday	No	No	Yes	Yes	Yes	Yes
Friday	No	No	Yes	Yes	Yes	Yes
Saturday	No	No	Yes	Yes	Yes	Yes
Sunday	No	No	Yes	Yes	Yes	Yes

7.1 Difficulties Encountered

In using the Glasnost tools, there are several problems encountered that may affect the reliability of testing results.

- Tracing of the BT messages from foreign testing servers has passed through quite a lot of countries and different countries may impose rate-limiting policies. As a result, the rate-limiting results gathered through foreign servers as in Germany may not be correct.
- As a result of that, we have implemented the testing server in Hong Kong and the same kind of results were obtained when compared with the result obtained through the Germany server. So it can be concluded that the results gathered is consistent
- There are difficulties in performing web hosting, as some ISP didn't allow the virtual web hosting. Network address translation need to be implemented in Linux system and port forwarding need to be configured in router.
- Since Java client were used for detecting BT and TCP traffic packets, some of the security settings in browsers and operating system may affect the performance of the measurement. Accurate measurement of rate limiting is not easily attainable.
- For the traffic rate measurement, since there may be different reasons that affect the traffic rate, for example, the machine that run the client is busy downloading other from the web, the operating system is busy running some other programs, this all will affect the measurement results.
- ISP may utilize traffic shaping in a way that just slightly lower the traffic rate of BT when they alerted the existence of measurement tools in the market. Because of the degree of lowering the traffic rate is not that high, the program may incorrectly interpret that the ISP implements no rate limiting.

8.1 Summary of Results

ISP Name	Block BT (port 6881)	Block BT (other ports)	Slow down BT traffic upload (port 6881)	Slow Down BT Traffic Upload (port 10009)	Slow Down BT Traffic Download (Port 6881)	Slow down BT traffic download (other ports)
HGC	No	No	No	No	No	No
Smartone	No	No	No	No	No	No
PCCW	No	No	Yes	Yes	Yes	Yes
i-Cable	No	No	No	No	No	No
HK BB	Yes	No	N/A	No	N/A	No

9.1 Conclusion

From the testing result of the BT-blocking strategies and rate-limiting traffic handling of BT traffic, it can be concluded from the testing result the followings:

- ISP didn't really change their BT traffic handling policies over the weeks.
- ISP HK Broadband is blocking BT and TCP traffic on BT port 6881 but no other blocking and rate-limiting on non-BT port
- ISP PCCW is rate-limiting both upload and download BT traffic regardless of the TCP port used

10.1 Future Challenges

Because ISP is noticing the existence of tools that are available in the market to measure the BT performance and customer is mostly concerned that ISP is controlling their traffic, ISP is using some-kinds of white-listing technique to avoid being others to detect the ISP traffic handling behavior. As a result, researchers may need to develop more tricky technique in order to avoid being detected by ISP that we are trying to measure their performance.

11.1 Reference

1. Notes on P2P blocking and evasion (http://www.educatedguesswork.org/p2pi.pdf)
2. BitTorrent Protocol—BTP/1.0 (http://jonas.nitro.dk/bittorrent/bittorrent-rfc.html)
3. Detecting BitTorrent Blocking (http://broadband.mpi-sws.org/transparency/results/detecting.html)
4. http://en.wikipedia.org/wiki/BitTorrent

Appendix A

Sample Packet Trace result gathered from the Tools

The following results showed that No BT blocking and no BT rate-limiting for the ISP being tested.

1257080089509 Starting BT client v2.2.1

1257080089509 Configuration: -p 19980 -h 255.255.255.255 -i eth0 -o

Client frankyeung-bt.no-ip.org 192.168.1.100 connected

Saving trace to bt_192.168.1.100_frankyeung-bt.no-ip.org_1257080094.dump

Received: setup downstream port 6881 duration 6 □

Start of Testing BT download with the well-known BT port 6881

Setting up downstream transfer.

1257080099655 Closing transfer socket

1257080099655 Closing listen socket

1257080099655 Both sockets closed successfully

Client: Linux,i386,2.6.28-11-generic,Sun Microsystems Inc.,1.6.0_16

Received: bt downstream port 6881 duration 11

Running BT downstream transfer port 6881 .

1257080103656 Socket 19 set to nonblocking

1257080103656 Sending handshake message□

Exchange of handshake message for setup of TCP transfer session

1257080103656 Waiting for handshake message

1257080103658 Received handshake message

1257080103658 Sending bitfield message□*Exchange of BT transfer message*

1257080103658 Waiting for bitfield message

1257080103658 Received bitfield message

1257080103658 Sending interested message

1257080103658 Waiting for unchoke message

1257080103659 Received unchoke message

1257080103659 Sending request message

1257080113658 Connection closed (read, errno=0, state=24)

1257080113658 End of transfer; 1567174546 bytes and 5978 pieces transferred

Resets: 0 ; Resets sent: 0

Transfered 1567174546 bytes in 10.002 seconds: 1253489440 bps

1257080115660 Closing transfer socket

1257080115660 Closing listen socket

1257080115660 Both sockets closed successfully

Client: Transfered 1.567096832E9 bytes in 10.001 seconds: 1.253552110388961E9 bps□*Finish testing the BT download with port 6881*

Received: tcp downstream port 6881 duration 11□*Start testing TCP download with port 6881*

Running TCP downstream transfer port 6881 .

1257080117659 Socket 19 set to nonblocking

1257080117659 Sending handshake message

1257080117659 Waiting for handshake message

1257080117660 Received handshake message

1257080117660 Sending bitfield message

1257080117660 Waiting for bitfield message

1257080117660 Received bitfield message

1257080117660 Sending interested message

1257080117660 Waiting for unchoke message

1257080117661 Received unchoke message

1257080117661 Sending request message

1257080127660 Read failed (errno=104, state=24)

1257080127660 End of transfer; 1575825727 bytes and 6011 pieces transferred

Resets: 0 ; Resets sent: 0

Transfered 1575825727 bytes in 10.001 seconds: 1260576753 bps

1257080129662 Closing transfer socket

1257080129662 Closing listen socket

1257080129662 Both sockets closed successfully

Client: Transfered 1.575747584E9 bytes in 10.0 seconds: 1.2605980672E9 bps□*Finish the TCP download with port 6881*

Received: bt downstream port 6881 duration 11□*Start the BT download with port 6881(Testing for rate-limiting)*

Running BT downstream transfer port 6881 .

1257080131661 Socket 19 set to nonblocking

1257080131661 Sending handshake message

1257080131661 Waiting for handshake message

1257080131662 Received handshake message

1257080131662 Sending bitfield message

1257080131662 Waiting for bitfield message

1257080131662 Received bitfield message

1257080131662 Sending interested message

1257080131662 Waiting for unchoke message

1257080131663 Received unchoke message

1257080131663 Sending request message

1257080141661 Connection closed (read, errno=104, state=24)

1257080141661 End of transfer; 1613576335 bytes and 6155 pieces transferred

Resets: 0 ; Resets sent: 0

Transfered 1613576335 bytes in 10.001 seconds: 1290772908 bps

1257080143663 Closing transfer socket

1257080143663 Closing listen socket

1257080143663 Both sockets closed successfully

Client: Transfered 1.61349632E9 bytes in 10.0 seconds: 1.290797056E9 bps □*Finish the BT download with port 6881 (Testing for rate-limiting)*

Received: tcp downstream port 6881 duration 11□*Test the tcp download with port 6881*

Running TCP downstream transfer port 6881 .

1257080145662 Socket 19 set to nonblocking

1257080145662 Sending handshake message

1257080145662 Waiting for handshake message

1257080145663 Received handshake message

1257080145663 Sending bitfield message

1257080145663 Waiting for bitfield message

1257080145663 Received bitfield message

1257080145663 Sending interested message

1257080145663 Waiting for unchoke message

1257080145664 Received unchoke message

1257080145664 Sending request message

1257080155663 Connection closed (read, errno=104, state=24)

1257080155663 End of transfer; 1619868103 bytes and 6179 pieces transferred

Resets: 0 ; Resets sent: 0

Transfered 1619868103 bytes in 10.001 seconds: 1295781361 bps

1257080157665 Closing transfer socket

1257080157665 Closing listen socket

1257080157665 Both sockets closed successfully

Client: Transfered 1.619787776E9 bytes in 10.0 seconds: 1.2958302208E9 bps□*Finish testing tcp download with port 6881*

Received: setup bt downstream port 4711 duration 6 □*Start bt downstream with port 4711*

Setting up downstream transfer.
1257080164664 Closing transfer socket
1257080164664 Closing listen socket
1257080164664 Both sockets closed successfully
Received: bt downstream port 4711 duration 11
Running BT downstream transfer port 4711 .
1257080168665 Socket 19 set to nonblocking
1257080168665 Sending handshake message
1257080168665 Waiting for handshake message
1257080168665 Received handshake message
1257080168665 Sending bitfield message
1257080168665 Waiting for bitfield message
1257080168666 Received bitfield message
1257080168666 Sending interested message
1257080168666 Waiting for unchoke message
1257080168666 Received unchoke message
1257080168666 Sending request message
1257080178666 Read failed (errno=104, state=24)
1257080178666 End of transfer; 1631665168 bytes and 6224 pieces transferred
Resets: 0 ; Resets sent: 0
Transfered 1631665168 bytes in 10.001 seconds: 1305189869 bps
1257080180668 Closing transfer socket
1257080180668 Closing listen socket
1257080180668 Both sockets closed successfully
Client: Transfered 1.631584256E9 bytes in 10.0 seconds: 1.3052674048E9 bps
□*Finish testing bt download with port 4711*
Received: tcp downstream port 4711 duration 11□*Test tcp download with port 4711*
Running TCP downstream transfer port 4711 .
1257080182667 Socket 19 set to nonblocking
1257080182667 Sending handshake message
1257080182667 Waiting for handshake message
1257080182667 Received handshake message
1257080182667 Sending bitfield message
1257080182667 Waiting for bitfield message
1257080182668 Received bitfield message
1257080182668 Sending interested message

1257080182668 Waiting for unchoke message
1257080182668 Received unchoke message
1257080182668 Sending request message
1257080192667 Connection closed (read, errno=104, state=24)
1257080192667 End of transfer; 1624324772 bytes and 6196 pieces transferred
Resets: 0 ; Resets sent: 0
Transfered 1624324772 bytes in 10.001 seconds: 1299385103 bps
1257080194669 Closing transfer socket
1257080194669 Closing listen socket
1257080194669 Both sockets closed successfully
Client: Transfered 1.624244224E9 bytes in 10.0 seconds: 1.2993953792E9 bps
□*Finish testing tcp download with port 4711*
Received: bt downstream port 4711 duration 11
□*Test BT download with port 4711*
Running BT downstream transfer port 4711 .
1257080196668 Socket 19 set to nonblocking
1257080196668 Sending handshake message
1257080196668 Waiting for handshake message
1257080196669 Received handshake message
1257080196669 Sending bitfield message
1257080196669 Waiting for bitfield message
1257080196669 Received bitfield message
1257080196669 Sending interested message
1257080196669 Waiting for unchoke message
1257080196669 Received unchoke message
1257080196669 Sending request message
1257080206669 Connection closed (read, errno=104, state=24)
1257080206669 End of transfer; 1618033004 bytes and 6172 pieces transferred
Resets: 0 ; Resets sent: 0
Transfered 1618033004 bytes in 10.001 seconds: 1294324281 bps
1257080208671 Closing transfer socket
1257080208671 Closing listen socket
1257080208671 Both sockets closed successfully
Client: Transfered 1.617952768E9 bytes in 10.0 seconds: 1.2943622144E9 bps
□*Finish BT download with port 4711*
Received: tcp downstream port 4711 duration 11□*Start testing of tcp download with port*

4711

Running TCP downstream transfer port 4711 .

1257080210673 Socket 19 set to nonblocking

1257080210673 Sending handshake message

1257080210673 Waiting for handshake message

1257080210673 Received handshake message

1257080210673 Sending bitfield message

1257080210673 Waiting for bitfield message

1257080210673 Received bitfield message

1257080210673 Sending interested message

1257080210673 Waiting for unchoke message

1257080210674 Received unchoke message

1257080210674 Sending request message

1257080220670 Connection closed (read, errno=104, state=24)

1257080220670 End of transfer; 1592341618 bytes and 6074 pieces transferred

Resets: 0 ; Resets sent: 0

Transfered 1592341618 bytes in 9.997 seconds: 1274218863 bps

1257080222672 Closing transfer socket

1257080222672 Closing listen socket

1257080222672 Both sockets closed successfully

Client: Transfered 1.592262656E9 bytes in 10.0 seconds: 1.2738101248E9 bps□

Finish testing tcp download with port 4711

Received: setup upstream port 6881 duration 5□*Start testing BT upload with port 6881*

Setting up upstream transfer.

1257080229671 Closing transfer socket

1257080229671 Closing listen socket

1257080229671 Both sockets closed successfully

Received: bt upstream port 6881 duration 10

Running BT upstream transfer port 6881 .

1257080233672 Socket 19 set to nonblocking

1257080233673 Received handshake message

1257080233673 Sending handshake message

1257080233673 Waiting for bitfield message

1257080233673 Received bitfield message

1257080233673 Sending bitfield message

1257080233673 Waiting for interested message

1257080233674 Received interested message

1257080233674 Sending unchoke message

1257080243673 End of transfer; 558366720 bytes and 2130 pieces transferred

Resets: 0 ; Resets sent: 0

Transfered 558366720 bytes in 10.001 seconds: 446660814 bps

1257080245673 Closing transfer socket

1257080245673 Closing listen socket

1257080245673 Both sockets closed successfully

#Client:Transfered 5.5839441E8 bytes in 10.001 seconds: 4.466708609139086E8 bps□

Finish BT upload testing with port 6881

Received: tcp upstream port 6881 duration 10□*Start tcp upload with port 6881*

Running TCP upstream transfer port 6881 .

1257080247674 Socket 19 set to nonblocking

1257080247675 Received handshake message

1257080247675 Sending handshake message

1257080247675 Waiting for bitfield message

1257080247675 Received bitfield message

1257080247675 Sending bitfield message

1257080247675 Waiting for interested message

1257080247678 Received interested message

1257080247678 Sending unchoke message

1257080257679 End of transfer; 556269568 bytes and 2122 pieces transferred

Resets: 0 ; Resets sent: 0

Transfered 556269568 bytes in 10.004 seconds: 444820734 bps

1257080259681 Closing transfer socket

1257080259681 Closing listen socket

1257080259681 Both sockets closed successfully

Client: Transfered 5.56297154E8 bytes in 10.0 seconds: 4.450377232E8 bps□

Finish tcp upload with port 6881

Received: bt upstream port 6881 duration 10□*Start BT upload with port 6881*

Running BT upstream transfer port 6881 .

1257080261675 Socket 19 set to nonblocking

1257080261681 Received handshake message

1257080261681 Sending handshake message

1257080261681 Waiting for bitfield message

1257080261681 Received bitfield message

1257080261681 Sending bitfield message

1257080261681 Waiting for interested message

1257080261681 Received interested message

1257080261681 Sending unchoke message

1257080271677 End of transfer; 543948800 bytes and 2075 pieces transferred

Resets: 0 ; Resets sent: 0

Transfered 543948800 bytes in 10.002 seconds: 435078637 bps

1257080273679 Closing transfer socket

1257080273679 Closing listen socket

1257080273679 Both sockets closed successfully

#Client:Transfered 5.43975775E8 bytes in 10.002 seconds: 4.35093601279744E8 bps

Finish BT upload with port 6881

Received: tcp upstream port 6881 duration 10*Start tcp upload with port 6881*

Running TCP upstream transfer port 6881 .

1257080275678 Socket 19 set to nonblocking

1257080275679 Received handshake message

1257080275679 Sending handshake message

1257080275679 Waiting for bitfield message

1257080275679 Received bitfield message

1257080275679 Sending bitfield message

1257080275679 Waiting for interested message

1257080275680 Received interested message

1257080275680 Sending unchoke message

1257080285679 End of transfer; 597426176 bytes and 2279 pieces transferred

Resets: 0 ; Resets sent: 0

Transfered 597426176 bytes in 10.001 seconds: 477907392 bps

1257080287679 Closing transfer socket

1257080287679 Closing listen socket

1257080287679 Both sockets closed successfully

Client: Transfered 5.97455803E8 bytes in 10.0 seconds: 4.779646424E8 bps*End tcp upload with port 6881*

Received: setup upstream port 4711 duration 5

Setting up upstream transfer.

1257080294681 Closing transfer socket

1257080294681 Closing listen socket

1257080294681 Both sockets closed successfully

Received: bt upstream port 4711 duration 10

Running BT upstream transfer port 4711 . *Start BT upload with port 4711*

1257080299684 Socket 19 set to nonblocking

1257080299685 Received handshake message

1257080299685 Sending handshake message

1257080299685 Waiting for bitfield message

1257080299685 Received bitfield message

1257080299685 Sending bitfield message

1257080299685 Waiting for interested message

1257080299686 Received interested message

1257080299686 Sending unchoke message

1257080309685 End of transfer; 738197504 bytes and 2816 pieces transferred

Resets: 0 ; Resets sent: 0

Transfered 738197504 bytes in 10.000 seconds: 590553574 bps

1257080311687 Closing transfer socket

1257080311687 Closing listen socket

1257080311687 Both sockets closed successfully

Client: Transfered 7.38234112E8 bytes in 10.0 seconds: 5.905872896E8 bps *End BT upload with port 4711*

Received: tcp upstream port 4711 duration 10 *Start tcp upload with port 4711*

Running TCP upstream transfer port 4711 .

1257080313686 Socket 19 set to nonblocking

1257080313686 Received handshake message

1257080313686 Sending handshake message

1257080313686 Waiting for bitfield message

1257080313686 Received bitfield message

1257080313686 Sending bitfield message

1257080313686 Waiting for interested message

1257080313687 Received interested message

1257080313687 Sending unchoke message

1257080323688 End of transfer; 553123840 bytes and 2110 pieces transferred

Resets: 0 ; Resets sent: 0

Transfered 553123840 bytes in 10.002 seconds: 442393827 bps

1257080325690 Closing transfer socket

1257080325690 Closing listen socket

1257080325691 Both sockets closed successfully

Client: Transfered 5.5315127E8 bytes in 10.0 seconds: 4.42521016E8 bps□*End tcp upload with port 4711*

Received: bt upstream port 4711 duration 10□*Start BT upload with port 4711*

Running BT upstream transfer port 4711 .

1257080327687 Socket 19 set to nonblocking

1257080327687 Received handshake message

1257080327687 Sending handshake message

1257080327687 Waiting for bitfield message

1257080327687 Received bitfield message

1257080327687 Sending bitfield message

1257080327688 Waiting for interested message

1257080327688 Received interested message

1257080327688 Sending unchoke message

1257080337689 End of transfer; 573571072 bytes and 2188 pieces transferred

Resets: 0 ; Resets sent: 0

Transfered 573571072 bytes in 10.002 seconds: 458759692 bps

1257080339691 Closing transfer socket

1257080339691 Closing listen socket

1257080339691 Both sockets closed successfully

#Client:Transfered 5.73599516E8 bytes in 10.002seconds:4.587878552289542E8 bps

End BT upload with port 4711

Received: tcp upstream port 4711 duration 10□*Start tcp upload with port 4711*

Running TCP upstream transfer port 4711 .

1257080341693 Socket 19 set to nonblocking

1257080341696 Received handshake message

1257080341696 Sending handshake message

1257080341696 Waiting for bitfield message

1257080341713 Received bitfield message

1257080341713 Sending bitfield message

1257080341713 Waiting for interested message

1257080341713 Received interested message

1257080341713 Sending unchoke message

1257080351695 End of transfer; 554696704 bytes and 2116 pieces transferred

Resets: 0 ; Resets sent: 0

Transfered 554696704 bytes in 10.002 seconds: 443666811 bps

1257080353697 Closing transfer socket

1257080353698 Closing listen socket

1257080353698 Both sockets closed successfully

Client: Transfered 5.54724212E8 bytes in 10.0 seconds: 4.437793696E8 bps□

End tcp upload with port 4711