Project Proxy/Cache: Eliminating Redundancy - Internet's Number One Enemy

Ivan Klimek, Tomáš Korenko, Marián Keltika*

P2P Workgroup, Computer Networks Laboratory
Faculty of Electrical Engineering and Informatics
Letná 9, 042 00 Košice, Slovakia

Ivan.Klimek@cnl.sk, Korenko.Tomas@gmail.com,
Marian.Keltika@cnl.sk

Abstract. Peer-to-Peer (P2P) currently represents the single largest portion of Internet's traffic. It is well known that the redundancy rate in P2P networks is very high; in fact redundancy is the main concept behind P2P. This paper will show how much traffic P2P generates and how much of it is redundant. The popularity distribution in P2P networks will be studied. Thru it, it will be shown that caching of P2P data is viable. Based on these calculations we present how much disk space is required for what hit percentage. A working proof-of-concept P2P Proxy/Cache will be presented that was shown to be able to serve a segment of several thousand users on commodity PC hardware. The second biggest portion of Internet traffic is streamed video, mostly also because of redundancy. Therefore we developed a truly generic Flash caching mechanism that will be also presented in this paper.

1 Introduction

Peer-to-Peer networks currently produce about 40 percent of the total Internet traffic. This is a massive drop when comparing with the values in 2005/2006 where it represented 60 to 80 percent. The cause of this change is aggressive shaping by Internet Service Providers needed to guarantee QoS for other services. Still, P2P represents about 71 percent of uplink bandwidth. [5] This paper will present a solution that eliminates all P2P client uploads and greatly minimizes downloads thru avoiding redundancy. This is possible thru utilizing a found attack vector in the most widely used P2P protocol - BitTorrent. There are many P2P protocols currently available; however the most popular is BitTorrent with 60-90 percent of total P2P traffic. [6, 7] An automated lightweight Man-in-the-Middle attack on the BitTorrent protocol will be

* Master degree study programme in field: Informatics
Supervisor: Assoc. Professor Ing. František Jakab, PhD., Computer Networks Laboratory, Faculty of Electrical Engineering and Informatics, Technical University Košice

IIT.SRC 2010, Bratislava, April 21, 2010, pp. 1–8.

presented. Thru this fully transparent "attack" it is possible to gain control over the whole protocol and all client downloads in the given network segment without the knowledge of the client. All BitTorrent clients are "vulnerable" to this attack as it is not a fault, but a protocol design weakness. [4]

The only similar project that the authors are aware of - pCache [2] does not seem to be active anymore and their implementation is also not scalable and therefore not usable outside of lab environment. Proxy/Cache aims at maximum effectivity and price/performance ratio.

The question is how could be something like this done? P2P generates tens of Petabytes of traffic, how much disk space would be needed to effectively cache it? To answer this question the popularity distribution in P2P networks will be studied and the required disk space determined. The immense amount of P2P traffic opens not only the question of disk space but also of the required processing power. The performance parameters and used optimization techniques will be therefore also presented.

Because of the mentioned shaping of P2P traffic, streamed video is gaining popularity as an alternative mean of on-demand content access. Streamed video represents great new challenges as ordinary caching techniques don't cope with the dynamic URLs used in most streaming services.

2 P2P Popularity Distribution

To collect the necessary data we wrote a spider that went thru the most popular BitTorrent search engine – ThePirateBay.org. Even if the site claims to have more than 2,5 million torrents indexed only 1 066 000 were found of which only 524 288 were active. Where active means at least one seeder and one leecher. The active BitTorrent content represents about 700 TB.

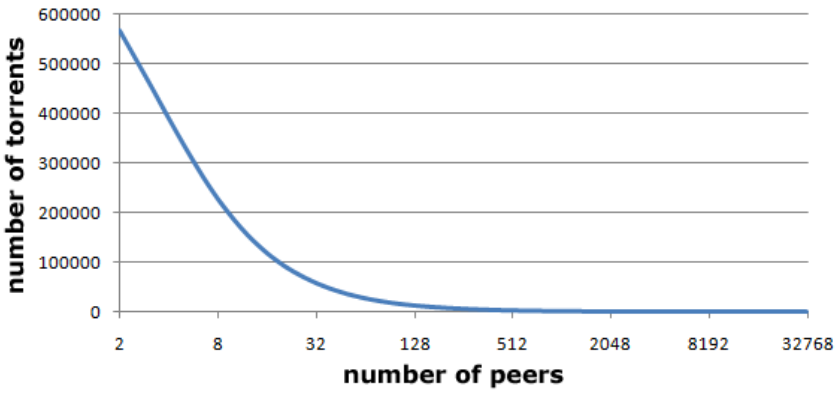

Figure 1. Distribution of torrents based on number of peers.

First we measured distribution of torrents based on number of peers (see Figure 1). It is clear that only a fraction of torrents has more than 200 peers.

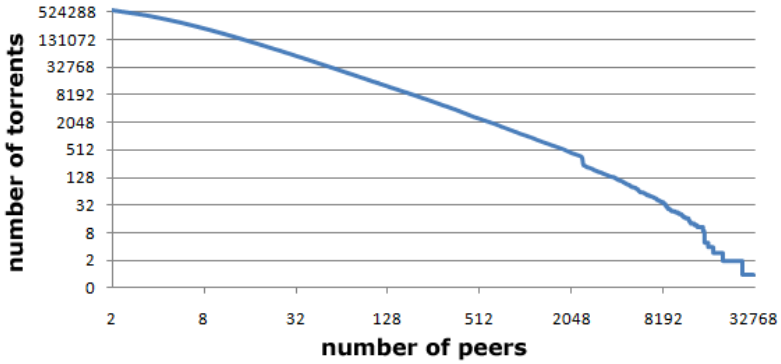

Figure 2. Detailed distribution with logarithmic vertical axis.

In the Figure 2, detailed graph of distribution, vertical axis represents number of torrents with more than x peers.

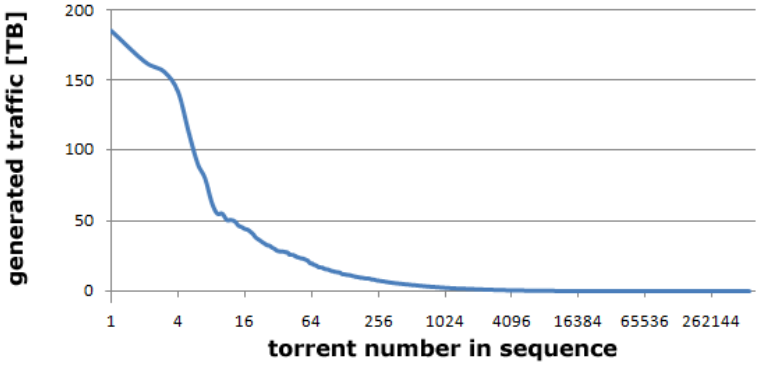

Figure 3. Traffic generated by torrent.

The horizontal axis represents the torrents sorted by their traffic while the vertical axis represents the corresponding traffic generated by the torrent. Figure 3 shows that just a few torrents are responsible for majority of the BitTorrent traffic.

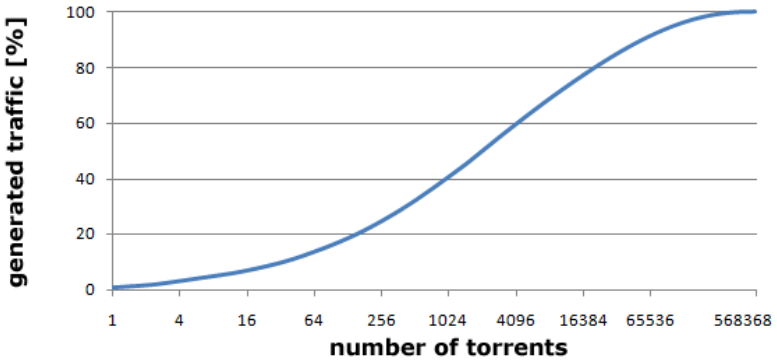

Figure 4. Total traffic generated by torrents.

Figure 4 demonstrates that just by caching of the 150 most popular torrents it is possible to save up to 20% of the BitTorrent traffic. This means that a Proxy/Cache with only approximately 500 GB of storage can save up to 8% of the ISP's downlink.

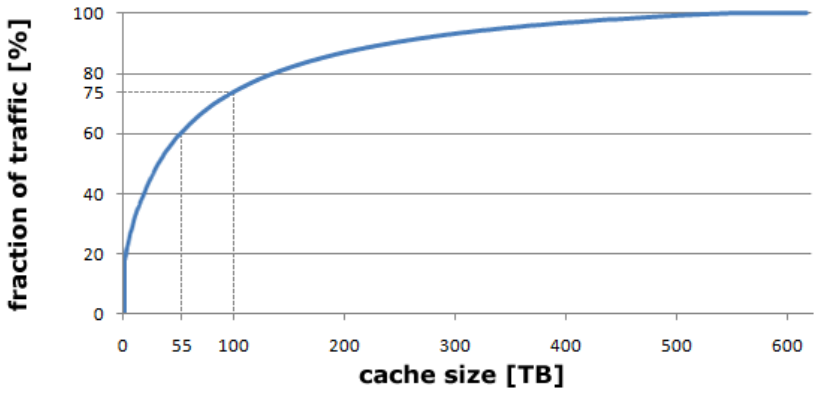

Figure 5. Cache size.

The cache disk space can be represented by the Pareto principle also called the 80-20 rule. By caching 20 percent of the active BitTorrent content up to 80 percent of hit probability can be achieved. According to our observations, optimal cache size for active torrents is between 55 TB and 100 TB (see Figure 5).

3 Proxy/Cache

The primary logic is based on listening to the network traffic and detecting "interesting" packets, in case of BitTorrent - Tracker requests. Proxy/Cache is a pro-active cache that means it caches every content requested. Therefore all Tracker requests are hijacked thru a MiTM TCP/UDP session hijack and redirected to the Proxy/Cache which then handles the downloads. Such approach minimizes the overhead connected with keeping the MiTM session only for the client redirection. Pro-active caching also benefits the client download speed even if the content is not yet cached. This is possible thru what we call Proxy/Cache assisted download, immediately after a new content request is detected the download is started. In parallel to downloading it, content is also provided to the client. Because of the superior link speed and the Downloader effectiveness Proxy/Cache assisted downloads are always faster than normal ones. Already cached content is being provided directly from the Proxy/Cache thus the download speed is limited only by the network/system load.

Tracker requests are not the only attack vector used, similar automated MiTM attacks are being fine tuned just now for DHT/PEX and Local Peer Discovery also.

As the Proxy/Cache is the only peer the client sees, there is absolutely no possibility of last-mile uplink. To keep the content available globally, the content is being seeded to the community based on available system resources and/or administratively set limits.

Proxy/Cache can be placed either directly in the operator's network or it can use port mirroring of in/out ports (see Figure 6).

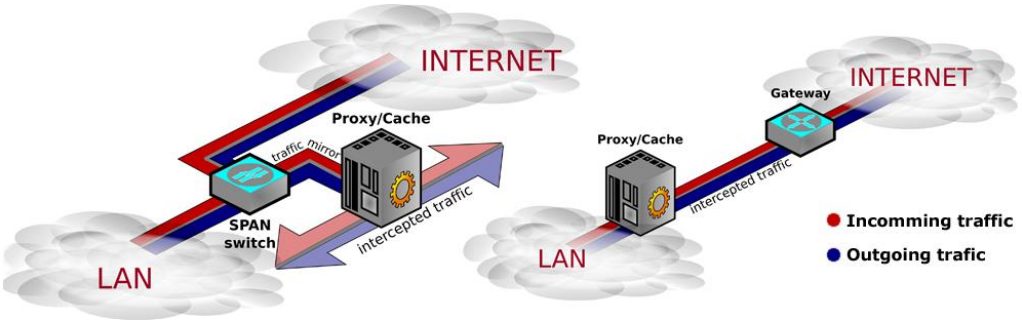

Figure 6. Port mirroring setup. Proxy/Cache can be placed either before or after the gateway.

Proxy/Cache runs on OpenSolaris and consists of the following components (see Figure 7):

- Interceptor – kernel level packet capturing/injection (our own dedicated driver)
- Downloader – downloads and provides content, based on LibTorrent [8]
- Database – minimalistic single purpose database
- Coordinator – application logic for the Downloader
- Analyser – Interceptor decision logic
- WebGUI – statistical/analytical user interface

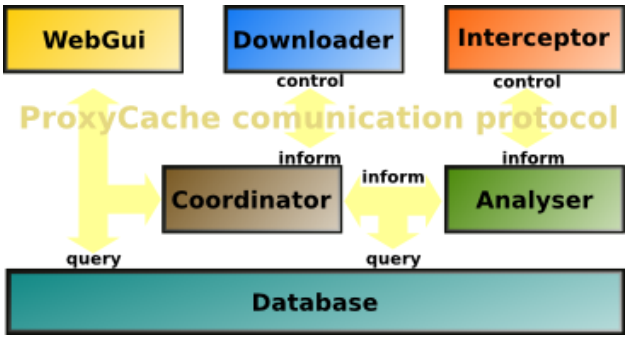

Figure 7. Process Responsibility hierarchy.

3.1 Optimization

To avoid any unnecessary overhead we decided not to use pcap, nor any other existing packet capture SW, but instead to develop our own highly optimized single purpose kernel module (see Figure 8). This module does a elementary packet selection and then forward the data to user space for further analysis without almost any performance loses. The analysis itself is offloaded to a GPGPU for parallel string matching.

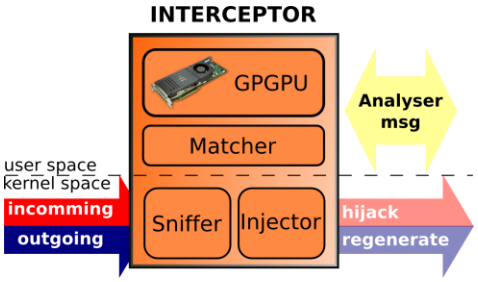

Figure 8. Interceptor model.

Further optimization is achieved by fully utilizing Intel IOAT technology which greatly reduces the CPU load on network traffic processing (see Figure 9).

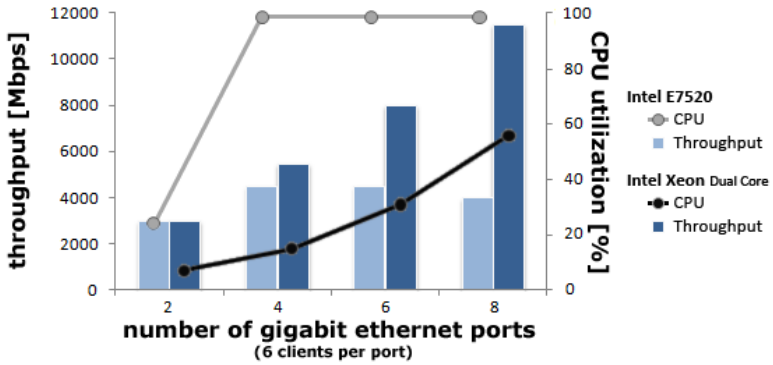

Figure 9. Performance difference between a IOAT enabled and disabled system. [3]

One of the most important features of OpenSolaris is the ZFS file system which can use SSD disks as fast read/write cache to boost I/O performance by a factor of 20 and more (see Figure 10). This way a hybrid storage subsystem can be created combining the capacity of classic HDDs with the I/O performance of SSDs.

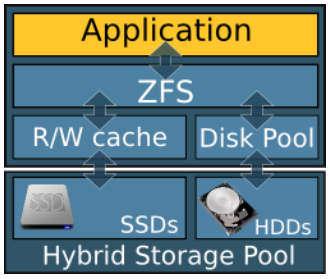

Figure 10. ZFS Hybrid storage pool.

Further we developed our own minimalistic database engine that keeps all the necessary data always in RAM to maximize the I/O performance. The downloader has been optimized for server usage so except of other tweaks it downloads first and then puts the downloaded blocks in the right order. This way it is by 65 percent faster than the most commonly used BitTorrent client uTorrent.

4 Generic HTTP/Flash Caching

Universal Resource Locators (URL) are because of various methods like session IDs becoming more unique than universal. Current caching techniques identify content directly by the URL or its part [9]. The best what classic caching can do is the ability of Squid to cache YouTube videos by using the video ID. But what about all the other sites? What about dynamic URLs? The possibilities of Squid end here. Generic HTTP/Flash caching therefore uses the server´s reply instead, because it can clearly identify all content all the time. The Content-type and Content-length parameters, together with hashing random blocks are used as a universal identifier. This mechanism allows us not only identify redundancy across any content completely ignoring its URL, but also to detect parallel live streams of the same content. Currently we are experimenting with an approach that we internally call Live stream proxying. If multiple streams of the same content are detected, only one is kept alive and the content is being transparently multiplexed to the local users from the proxy/cache.

Caching of Flash videos can be very efficient since storing only a small set of objects can produce high hit ratios. That is, by storing only 10% of long-term popular videos, a cache can serve almost 80% of requests (see Figure 11). [1]

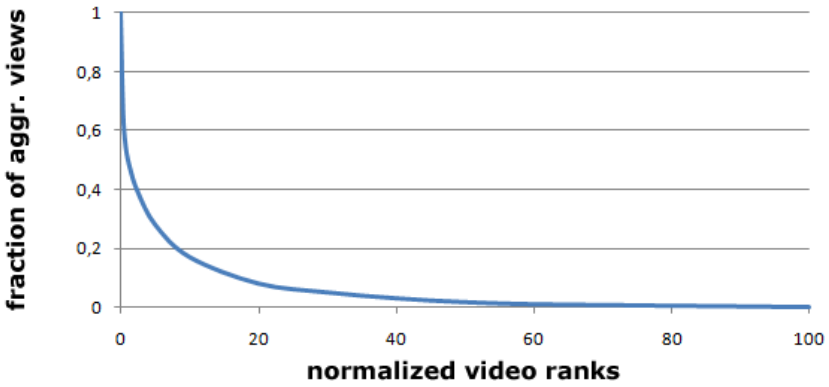

Figure 11. Youtube aggregated video views.

5 Conclusions

The presented solution provides effective means of P2P traffic optimization with no known negative side effects. It's true that the usage of automated hacking techniques makes it controversial but the authors believe that the end justifies the means even

more if there are no negative effects on the clients nor on the network. Most data redundancy outside of the served segment is eliminated; redundantly requested content is served from the Proxy/Cache. For example with 100TB storage it could save up to 80 percent of P2P traffic. No client uplink, not because of shaping but because of the full protocol control. Client uplink is simply not required anymore. This means a change from 71 percent to 0 in P2P uplink usage in the local segment, without any network degradation. The network segment size doesn't affect the cache size but just the I/O performance requirements. The whole solution is completely transparent and no part of the currently used technology has to be changed. Because content is provided from the Proxy/Cache the client's download speeds are maximized. No blocking or shaping of P2P is required anymore. Proxy/Cache doesn't require any special HW; it can run on low cost commodity PC hardware. This paper also presented a novel high efficiency approach to generic http/flash caching that except of allowing caching all the session protected/dynamic URL content, that until now couldn't be cached, also opens new possibilities like live stream proxying.

References

[1] Cha, M., Kwak, H., Rodriguez, P., Ahn, Y., Moon, S.: 2009. Analyzing the video popularity characteristics of large-scale user generated content systems. *IEEE/ACM Trans. Netw.* 17, 5 (Oct. 2009), 1357-1370.

[2] Hefeeda, M., Hsu, C., Mokhtarian, K.: Design and Evaluation of a Proxy Cache for Peer-to-Peer Traffic. School of Computing Science, Simon Fraser University, July 2008.

[3] Intel Corporation: "Accelerating High-Speed Networking with Intel® I/O Acceleration Technology", 2009, [Online; accessed 24-February-2010], [Online], Available: http://download.intel.com/technology/comms/perfnet/download/ 98856.pdf

[4] Klimek, I.: P2P Proxy/Cache. *In Proc. of of the Third International Conference on Internet Technologies and Applications* (ITA 09), Wrexham, UK, September 2009. [Online; accessed 24-February-2010], [Online], Available: http://stargate.cnl.tuke.sk/~klimek/ita09.pdf

[5] Pries, R., Wamser, F., Staehle, D., Heck, K., Tran-Gia, P.: "On Traffic Characteristics of a Broadband Wireless Internet Access," *Next Generation Internet Networks, 2009. NGI '09* , vol., no., pp.1-7, 1-3 July 2009.

[6] Schulze, H., Mochalski, K.: Internet Study 2007. ipoque, November 2007.

[7] Schulze, H., Mochalski, K.: Internet Study 2008/2009. ipoque, November 2009.

[8] http://www.rasterbar.com/products/libtorrent/

[9] http://www.squid-cache.org/