CONTENT SERVER SATURATION DETECTION AND LOAD BALANCING BASED ON FINANCIAL TECHNIQUES

Radu A. BADEA[1], Eugen BORCOCI[2]

The steady demand for performance in Media Content dissemination has forced both Network and Service Providers acting in (real time) content streaming to innovate new techniques and extend existing ones as to improve Quality of Services (QoS). In many situations Content Service Providers (CSP) and Network Providers (NP) operate as distinct entities, making it often difficult for CSPs to acquire detailed network related information. This context introduced the necessity of new emerging technologies, able to resolve this issue as far as it is possible under these restricted conditions. In our paper we address (partially) the mentioned points, by proposing a strategy for dynamic Content Server overload detection and selection, based solely on measurements done at either server location and/or client premises, utilizing a method derived from Financial Technical Analysis.

Keywords: Bollinger Bands, Financial Technical Analysis, Server Overload

1. Introduction

The Content Server (CS) selection and overload detection problem is often found as a central optimization issue in content delivery systems. Various solutions were proposed, taking into consideration both general performance criteria and more specialized ones, regarding only some subclasses of applications. However this is still an open research issue especially in heterogeneous network environments and business models involving different actors.

In DONAR [1], a server selection procedure is presented that "outsource" the replica decision process to a third party authority such as the Cloud Provider where the Content is stored, permitting at the same time client customization through API calls. These clients are not burdened to decide which server to choose, and CSPs can design their core business without the need to run DNS machines and IP mapping and coordination software for request distribution among datacenters. In [2], a survey of different approaches for VoD (video on demand) Systems is done. Among the existing solutions, video stream multicast delivery for clients with closely-spaced requests for the same video is mentioned

[1] Dept.of Telecommunications, University POLITEHNICA of Bucharest, Romania, e-mail: rbadea@elcom.pub.ro

[2] Dept.of Telecommunications, University POLITEHNICA of Bucharest, Romania

and dynamic system state information is used to improve server selection performance. A more recent study [3] investigates in detail the YouTube CDN with the intention to understand the mechanisms and policies set behind user video download. The results show that besides Round Trip Time (RTT) between user and server, many other factors are relevant for the server selection process, depending on various, real-time conditions and other issues like day/night request variations, DNS latency, load-balancing capabilities etc.

Since the area of server overload detection and CS selection is a broad one, despite existing solutions is beneficial to propose alternative approaches using techniques successfully deployed in other domains, like Financial Markets.

Technical Analysis (TA) is a methodology for forecasting the stock price direction by evaluation of past market data, especially price and volume [4][5]. By introducing statistical estimations and calculating mathematical correlations for important market parameters, price trend detection and future market valuation is desired as output. While these techniques are widely used by stock market traders, their usefulness for other scientific areas is beginning to be recognized as well.

For example, in [6] we can see how Bollinger Trading Bands [7] and Keltner Channels are used for Network Telescope Datasets. The authors try to determine present and future malevolent internet activity (such as viruses and distributed denial of service attacks) by regarding past traffic data trough the aforementioned technical assessment. Overall conclusion is that some of these techniques originating in the financial field are also applicable to network data analysis.

The paper is organized as follows: Financial Technical Parameters in Chapter 2, the Paper Objectives in Chapter 3, Bollinger Bands in Chapter 4, the Test System Architecture in Chapter 5 and finally, Conclusions in Chapter 6.

2. Financial Technical Parameters Definitions

The current section will shortly introduce TA parameters like Moving Average (MA) and Bollinger Bands (BB) [7], setting as central research objective CS saturation detection and the development of a decision strategy for systems where saturation occurs.

The first signal that will be used is a simple Moving Average that is defined as the normal average over the last N values of the current signal, shifted at each iteration step with one position forward.

The Standard Deviation (SD) is representing the dispersion in a data set. A low SD (close to zero) denotes data values in the vicinity of the mean value (Expected Value) of the data set. Similarly, higher SD shows a value spreading over a wider range around the Expected Value. The SD can be written as:

$$\sigma = \sqrt{\frac{1}{N} * \sum_{i=1}^{N} (x_i - \mu)^2} \; where\; \mu = \frac{1}{N} * \sum_{i=1}^{N} x_i \qquad (1)$$

Bollinger Bands evolved from the concept of trading bands (i.e. a "channel" for stock price values), permitting a definition of relative "highs" and "lows", compared to previous trades. BBs are used also as a volatility indicator, signaling spikes and other anomalies. Technically, they are built from three different signals as mentioned next:

- A N-period Moving Average (MA).
- "Upper" limit (band) at "alpha" times the standard deviation (N-period) above the MA (MA + α*σ).
- "Lower" limit at "alpha" times the standard deviation below the MA (MA - α*σ), "alpha" being a parameter.

The interpretation of signals generated by BB observation varies among users in the financial world, but for a networking and server performance related study, the main benefits brought are the detection of "instability" regions in performance parameters like server network traffic and CPU utilization when approaching saturation.

3. Description of Objectives and Initial Requirements

The current section describes the functionality of a content chunking data transfer design. A functional system will be designed and implemented, enabling realistic result evaluations. The main points, around which its architecture was built, are:

- The use of techniques employed frequently in the financial industry, such as Bollinger Bands and Moving Averages. These methods have been extended to become suitable for our specific purpose of detecting various server saturation conditions.
- The target file type is MPEG4 (.mp4) encapsulated audio/video content.
- A “chunking” of content into smaller self-contained pieces (i.e., a larger video is split to several segments that can be regarded as mini-videos and played back separately).
- A parallel, multi-server load balanced chunk download, done by the client application but considering overload condition information, received from the CSs.

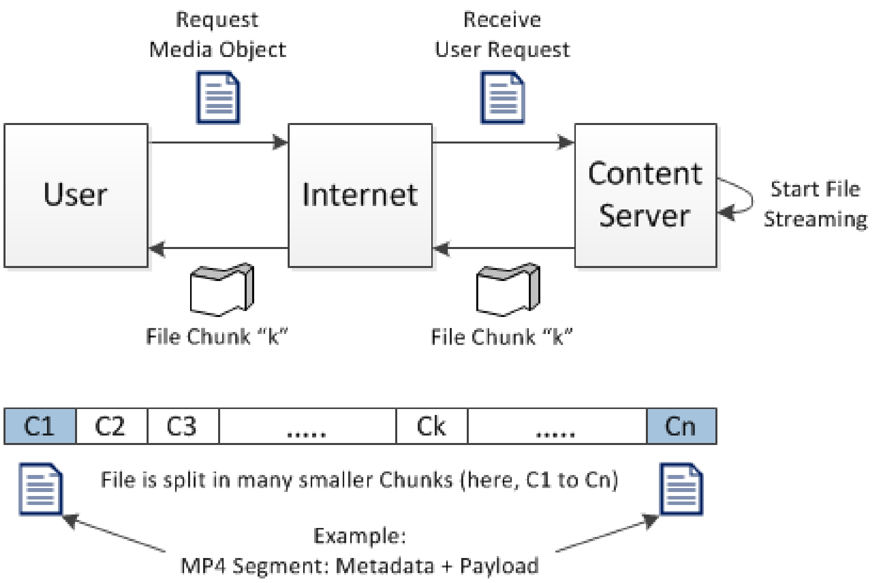


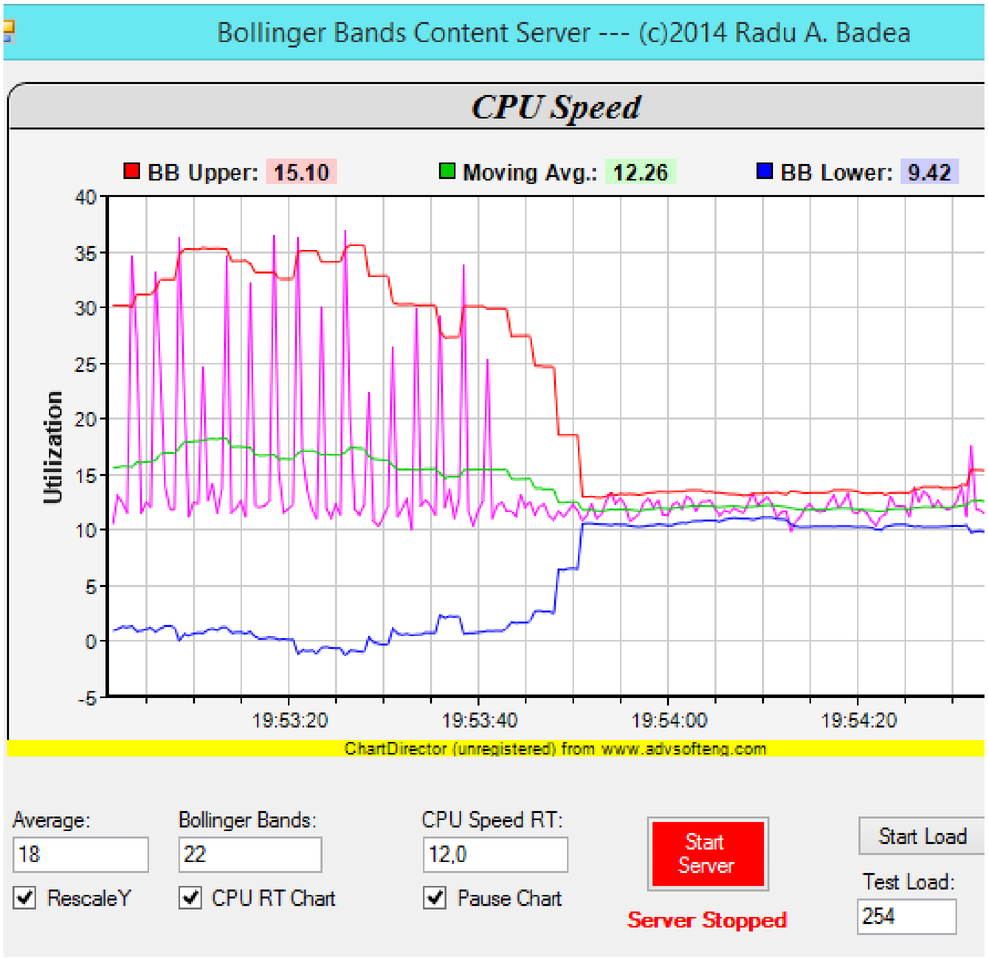
Fig. 1 Content Chunking for Transmission

For the experimental part we will need some test videos that should be transferred between the application peers. The easiest way to achieve movie chunks is to split a larger video into segments by a binary encoder/decoder (*Fig. 1*). Because of MP4 file format requirements, an efficient and reliable splitting can be achieved by using some third party open source software tools, like the *GPAC - MP4Box* tools [8]. The GPAC Project provides an Open source multimedia framework composed of several tools aimed to enable custom media file manipulation, including splitting, recomposing, trans-coding etc. of media content. The applications are cross-platform and designed to be used by a wide audience, including researcher, content creators and even industrial companies. For more details, please see [8].

The individual chunks are transferred as binary encoded data and whenever a client will download a chunk, it is guaranteed that it is possible to play it back without waiting for the others, since format boundaries (when splitting the initial MP4 file) are respected.

4. Bollinger Bands - Contextual Application

For overload detection, the BB property used is that these bands are widening themselves each time a higher volatility of the reference signal occurs, permitting thus a facile recognition of situations like that.



Average: 18

Bollinger Bands: 22

CPU Speed RT: 12,0

☑ RescaleY

☑ CPU RT Chart

☑ Pause Chart

Server Stopped

Test Load: 254

Fig. 2 Bollinger Bands - CPU Load Chart

As threshold on the signal level we will use a chart area value obtained by integrating over a defined time period the BB difference (upper – lower BB, see *Fig. 2*). This has as advantage the generation of a “smoothened” decision parameter, where singular random load spikes (false alarms) are filtered out, increasing the confidence in the detection process.

Fig. 2 is showing the transition from a near overload situation (the left side of the image) towards a stable phase, where the CPU is very lightly loaded (right image side). It can be seen how the BBs are gradually approaching when the CPU load (magenta chart) is decreasing. The BB gap reduction is gradual although the CPU reduces its activity quite sharply already at 19:53:42 (see timeframe on chart). This is designed like this in order to smoothen singular CPU spikes that could trigger false load alarms. In the test situation found in this example, the CPU Load MA (green) is between 10% and 15% for the "stable" phase and between 15% and 20% for the load/saturation time. The magenta spikes represent the intervals when intense CPU activity is simulated by the application and that trigger the widening of the BBs.

5. Test System Architecture and Functional Description

For the time being, a first application version was implemented on both Client and Server ends aimed at detecting and solving overload conditions of the server CPU(s). The normal working scenario consists of a client, connected at the same time to several content servers and running parallel chunk download sessions, dividing the requests among these servers following a specialized algorithm. Each client will get a download “quota” from the server, i.e., a certain amount of processing resources.

Please note that although the current implementation is focused towards CPU load, the architecture can be used for many other system performance parameters, like network bandwidth, disk access etc.

The server application is built so that when an overload situation occurs, a signaling protocol is initiated by the server with current connected (or downloading) clients, transmitting them the new internal server state together with a modified (reduced) quota. The clients then have to decide a strategy to overcome this limitation, by reallocating chunk requests to other, lesser crowded servers. A high-level representation is depicted on *Fig 3*.

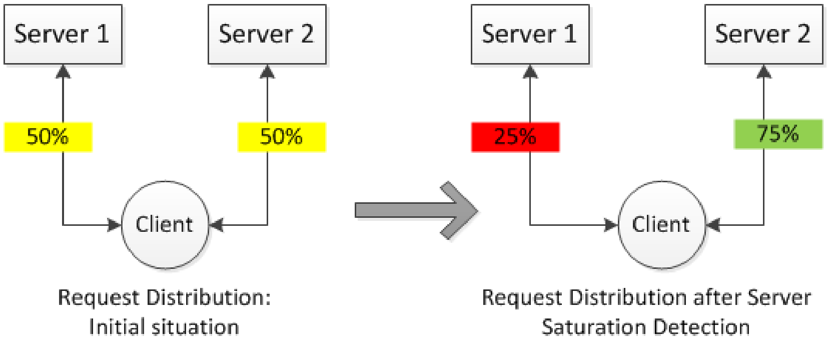


Fig. 3 Content Server Request Distribution - before and after Saturation Detection

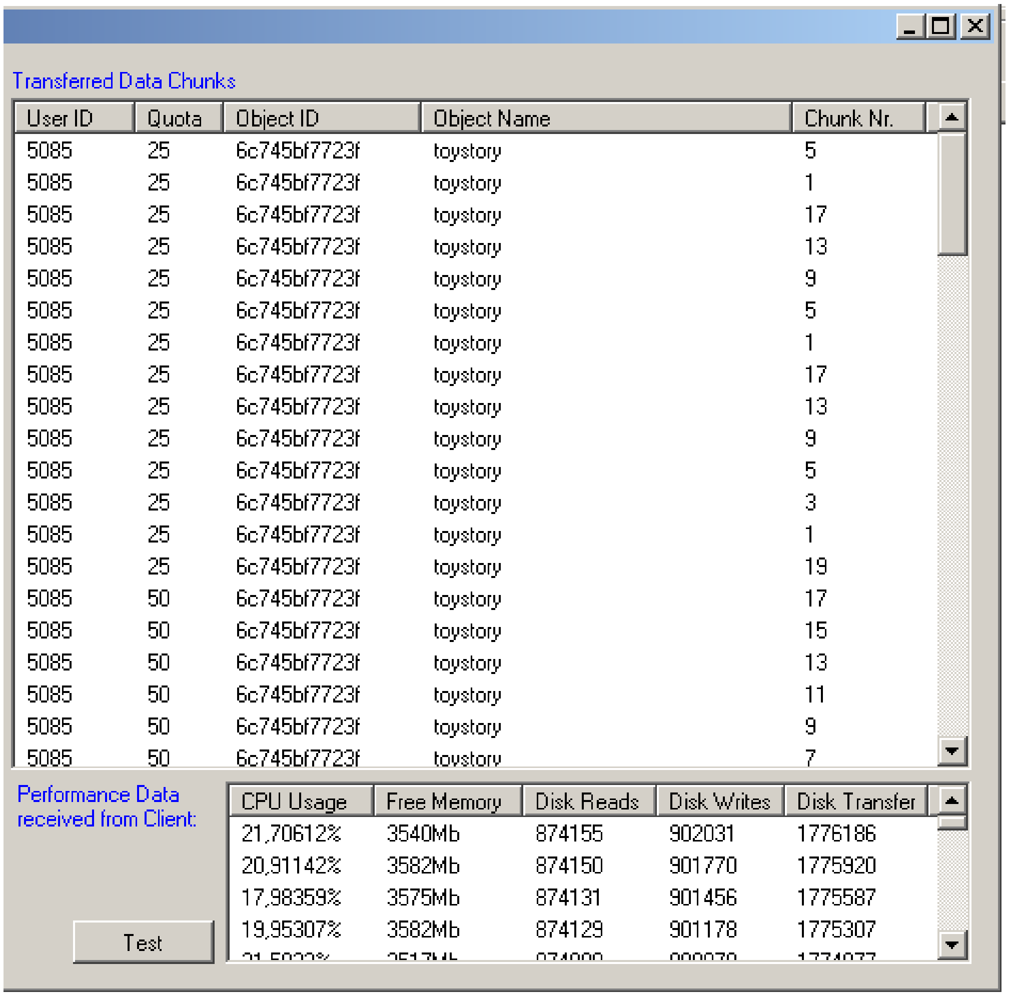
In the basic scenario implemented now, a client will connect to only two servers, having requests split evenly across them. The server applications have the possibility to “simulate” a large CPU load (starting background threads where long running computations are performed). The overload detection is done by evaluating an area contained between the two BBs surrounding the CPU load signal. (*Fig. 2*). The testbed contains chunk servers located at IP: "141.85.43.135" for "Server 1" and "141.85.58.64" for "Server 2". The client application is residing on a third, different machine. Port 8012 is used for communication.

When a saturation condition is met the server will reduce the client download quota from 50% to 25 % and the client will, at his turn, increase the number of requests routed towards the second, free server, which will serve now up to 75% of all client requests. This is shown on *Fig. 4*, in the “quota” column (Server 1). The corresponding “Chunk Nr.” columns show the actual chunk numbers requested and that are transferred from the respective server to the client. From a total of 19 chunks (the test video is split to 19 smaller videos with the help of the MPEG splitter found in the MP4Box application), numbered from 1 to 19, we can see how the client request reallocation actually works. The "User ID", "Object ID / Name" fields identify the connected users and requested files. In our test evaluation, to provide easy understandable and unambiguous contextual descriptions, only one user is connected at a time, requesting a movie called "toystory".

At Server 1, where the quota is transitioning from 50% to 25%, chunk sequence numbers are: 1, 3, 5 and then 9, 13, 17 etc. At the same time, the other server is rising its quota to 75% for this client, thus the chunk sequence becomes: 2, 4, 6 and then 7, 8, 10, 11, 12, 14 etc. These two latter distributions have to be compared with the normal situation (50% for both servers), when one is delivering even sequence numbers and the other, odd ones.

The switching is done by the client application, but based on information received from the connected servers. The dynamic load balancing is chunk based, so that at each time moment, the download process is performed from several sources concurrently. The client has the responsibility to reorder the chunks so that playback is done correctly.

Fig. 5 is showing the client application, with corresponding saved chunks, for each one displaying also the source (originating server) and chunk number. The red marked area is a full movie download (chunks 1 to 19) and, starting with chunk 5, the server switching process can be identified, this time from the client perspective. The normal server succession S1-S2 is replaced by S1-S2-S2-S2, denoting the modified download quota from Server 1 (S1) and Server 2 (S2). The filenames contain the chunk number and also a timestamp. On the bottom left image corner we can see the two servers (IPs), where our client is connected to.



Transferred Data Chunks

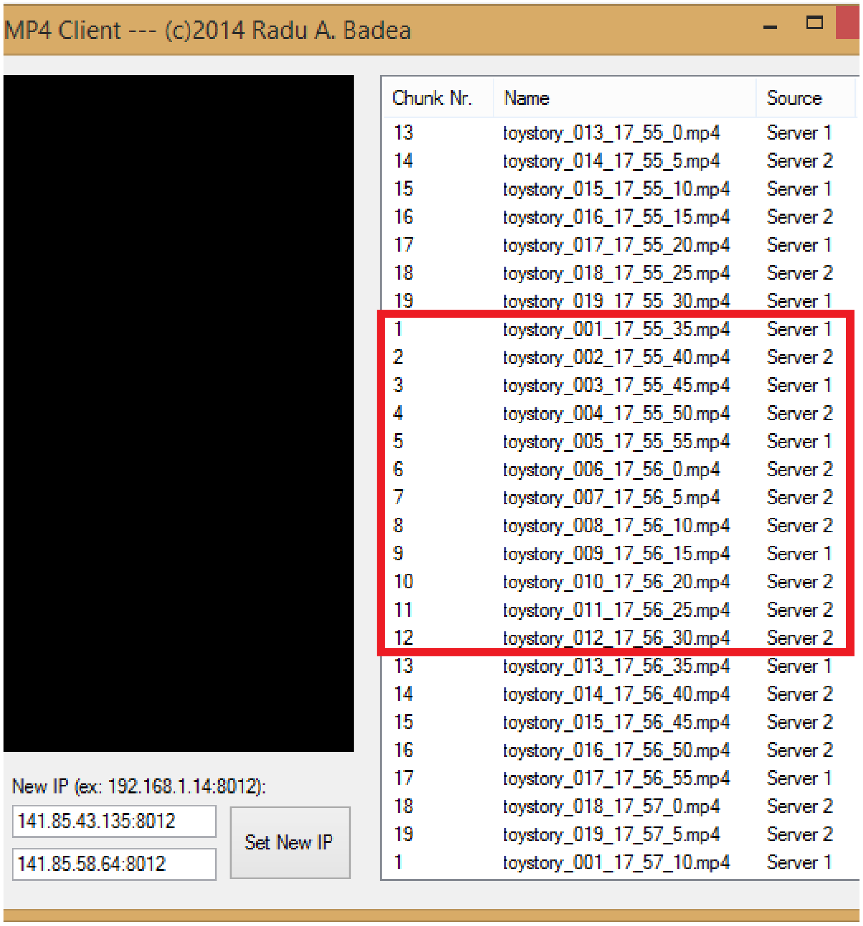
User ID	Quota	Object ID	Object Name	Chunk Nr.
5085	25	6c745bf7723f	toystory	5
5085	25	6c745bf7723f	toystory	1
5085	25	6c745bf7723f	toystory	17
5085	25	6c745bf7723f	toystory	13
5085	25	6c745bf7723f	toystory	9
5085	25	6c745bf7723f	toystory	5
5085	25	6c745bf7723f	toystory	1
5085	25	6c745bf7723f	toystory	17
5085	25	6c745bf7723f	toystory	13
5085	25	6c745bf7723f	toystory	9
5085	25	6c745bf7723f	toystory	5
5085	25	6c745bf7723f	toystory	3
5085	25	6c745bf7723f	toystory	1
5085	25	6c745bf7723f	toystory	19
5085	50	6c745bf7723f	toystory	17
5085	50	6c745bf7723f	toystory	15
5085	50	6c745bf7723f	toystory	13
5085	50	6c745bf7723f	toystory	11
5085	50	6c745bf7723f	toystory	9
5085	50	6c745bf7723f	toystory	7

Performance Data received from Client:

CPU Usage	Free Memory	Disk Reads	Disk Writes	Disk Transfer
21,70612%	3540Mb	874155	902031	1776186
20,91142%	3582Mb	874150	901770	1775920
17,98359%	3575Mb	874131	901456	1775587
19,95307%	3582Mb	874129	901178	1775307
[illegible]	3517Mb	[illegible]	[illegible]	[illegible]

Fig. 4 Server 1 – 141.85.43.135 – Chunk Transfer and Client Performance Data

Server 1 has reduced quota value for User 5085 from 50% to 25% due to CPU overload. The delivered Chunk Sequence Numbers on the right show this reduction. Below, CPU Usage, Free RAM Memory, Disk Access etc. are received from the client machine and displayed.



New IP (ex: 192.168.1.14:8012):

141.85.43.135:8012

141.85.58.64:8012

Chunk Nr.	Name	Source
13	toystory_013_17_55_0.mp4	Server 1
14	toystory_014_17_55_5.mp4	Server 2
15	toystory_015_17_55_10.mp4	Server 1
16	toystory_016_17_55_15.mp4	Server 2
17	toystory_017_17_55_20.mp4	Server 1
18	toystory_018_17_55_25.mp4	Server 2
19	toystory_019_17_55_30.mp4	Server 1
1	toystory_001_17_55_35.mp4	Server 1
2	toystory_002_17_55_40.mp4	Server 2
3	toystory_003_17_55_45.mp4	Server 1
4	toystory_004_17_55_50.mp4	Server 2
5	toystory_005_17_55_55.mp4	Server 1
6	toystory_006_17_56_0.mp4	Server 2
7	toystory_007_17_56_5.mp4	Server 2
8	toystory_008_17_56_10.mp4	Server 2
9	toystory_009_17_56_15.mp4	Server 1
10	toystory_010_17_56_20.mp4	Server 2
11	toystory_011_17_56_25.mp4	Server 2
12	toystory_012_17_56_30.mp4	Server 2
13	toystory_013_17_56_35.mp4	Server 1
14	toystory_014_17_56_40.mp4	Server 2
15	toystory_015_17_56_45.mp4	Server 2
16	toystory_016_17_56_50.mp4	Server 2
17	toystory_017_17_56_55.mp4	Server 1
18	toystory_018_17_57_0.mp4	Server 2
19	toystory_019_17_57_5.mp4	Server 2
1	toystory_001_17_57_10.mp4	Server 1

Fig. 5 Client Application - Receiving Chunks from Servers

The Client Application receives Chunks from both Server 1 and Server 2. When a saturation situation occurs at any server, it redirects requests to the other machine to balance out this limitation. The red box is marking such a situation with Server 1 entering overload.

6. Conclusions

This paper has presented results obtained by applying Financial Techniques methods for network related measurements, especially considering Bollinger Bands and their capacity to detect "instability" regions for various types of signals, including those generated by server CPU Load and Network Traffic. The proposed solution is based on a distributed approach, using several servers at the same time for separate content chunk download.

The main benefit, compared to the solution where the server would be switched entirely, resides in the possibility to fine-tune the whole downloading process. So, instead of renouncing completely at Server 1 for Server 2, and maybe creating an *underload* situation on Server 1, the "quota" chunk method allows for a better load balance between clients and involved CSs. Simultaneously loading servers are another possible situation, that will be analyzed in a future paper.

Also, horizontal server scalability can be implemented more easily, since theoretically a client can connect to a lot of CSs at once, lowering the burden on each individual server in real world scenarios where we also have numerous concurrent clients competing for these servers.

The main innovative contributions here are related to the server overload detection with the help of Bollinger Bands architecture.

Acknowledgement

This work has been partially supported by the Research Project DISEDAN, No.3-CHIST-ERA C3N, 2014- 2015

REFERENCES

[1] *P. Wendell, J. W. Jiang, M. J. Freedman, and J. Rexford*, “DONAR: Decentralized Server Selection for Cloud Services” Department of CS, Princeton University, SIGCOMM 2010 https://www.cs.princeton.edu/~jrex/papers/donar-sigcomm10.pdf

[2] *N. Carlsson and D. L. Eager*, “Server Selection in Large-scale Video-on-Demand Systems”, ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 2010 http://www.cs.usask.ca/grads/nic169/papers/tomccap08.pdf

[3] *R. Torres, A. Finamore, J. R. Kim, M. Mellia, M. Munaf, S. Rao* “Dissecting Video Server Selection Strategies in the YouTube CDN”, Proceeding ICDCS 2011, pp. 248-257 https://engineering.purdue.edu/~isl/ICDCS11.pdf

[4] *Kirkpatrick and Dahlquist*. “Technical Analysis: The Complete Resource for Financial Market Technicians”. Financial Times Press, 2006, page 3.

[5] *Murphy, John* “Technical Analysis of the Financial Markets”. New York Institute of Finance, 1999, pp. 1-5, 24-31.

[6] *B. Cowie and B. Irwin* “An Evaluation of Trading Bands as Indicators for Network Telescope Datasets”, SATNAC 2011 http://www.satnac.org.za/proceedings/2011/papers/Network_Management_and_OSS/149.pdf

[7] Bollinger Bands: http://www.bollingerbands.com/

[8] GPAC - MP4Box Application: http://gpac.wp.mines-telecom.fr/mp4box/