Building CDN Federation[1]

Piotr STAPP, Piotr ZGADZAJ

Wydział Matematyki i Nauk Informacyjnych PW,
ul. Koszykowa 75, 00-662 Warszawa
p.stapp@mini.pw.edu.pl, p.zgadzaj@mini.pw.edu.pl

ABSTRACT: The following document has two parts. The first describes an example of well-balanced CDN on PlanetLab environment. In the second part we present ideas how well-balanced CDN can be extended to CDN federation.

KEYWORDS: Content Distribution Network, CDN, PlanetLab, Federation.

1. Introduction

In 2012 Poland and Ukraine hold the UEFA European Cup in soccer. Using historical data, we know that in 1998 official Soccer World Cup Website had 1,35 billion request over 3 months, with peaks 73 million request per day and 12 million request per hour [1]. These numbers were exceeded during Summer Olympic Games in 2004 and 2008. We do not have data for 2012, but for sure these numbers were exceeded several times. We knew that 8 million people watch live Felix Baumgartner jump on YouTube [7].

Nowadays Content Delivery Network (CDN) is a solution for the above problem. But many servers give us only one thing: possibility of user distribution. The Wikipedia entry for CDN states: "A content delivery network or content distribution network (CDN) is a system of computers networked together across the Internet that cooperate transparently to deliver content to end users, most often for the purpose of improving performance, scalability, and cost efficiency." But an important question is how to improve those? We will try to find a proper answer.

[1] The paper is partially founded by EU grant FP7-ICT-224263 OneLab2: *An Open Federated Laboratory Supporting Network Research for the Future Internet.*

The first part is an example of how to build well-balanced CDN. These ideas were presented during FedCSIS 2012 in article *Building well-balanced CDN* [11].

In the second part (§ 6) we extend the above idea to worldwide scale.

2. CDN Architecture

2.1. Notation

First of all we need to define a notation. We decided to use the same as in [6], depicted below:

– Web Server (WS) – is a container of content;

– Service Registry (SR) – discovers and stores resources and policy information in a local domain.

2.2. Architecture (CDN definition)

Using the above notation we can define: "CDN is built with one or more Web Servers (WS'S) and one Service Registry (SR)". In the simplest CDN definition SR works as "first line" for user requests. It can also be responsible for resource discovery and policy in local domains. WS works as a container for content available for user. Architecture of CDN is presented in figure1 below:

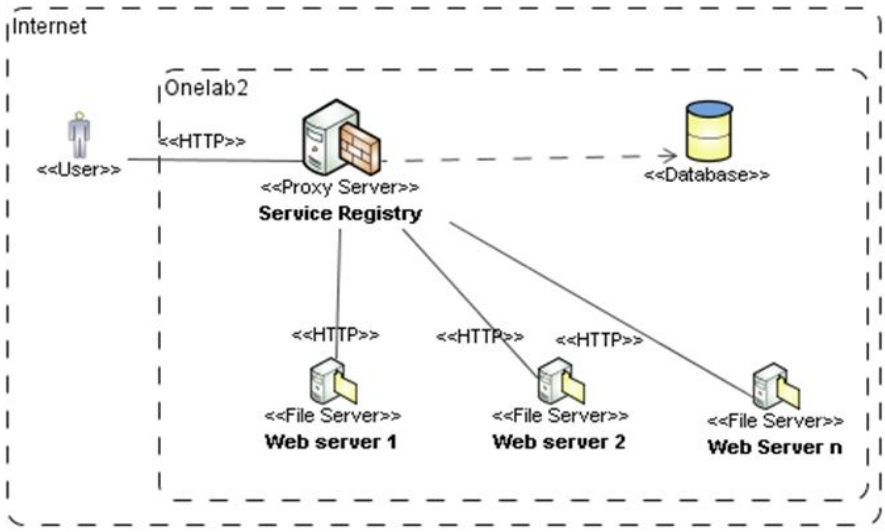

Figure 1. CDN architecture

The main idea is as follows (presented on figure 2 below):

- End user sends a request for some content to the Service Registry (SR);
- SR finds "the best" Web Server (WS) for this user;
- SR redirects the user to "the best" WS;
- The WS receives the request;
- The user downloads the requested content from WS.

The figure 2 presents sequence diagram:

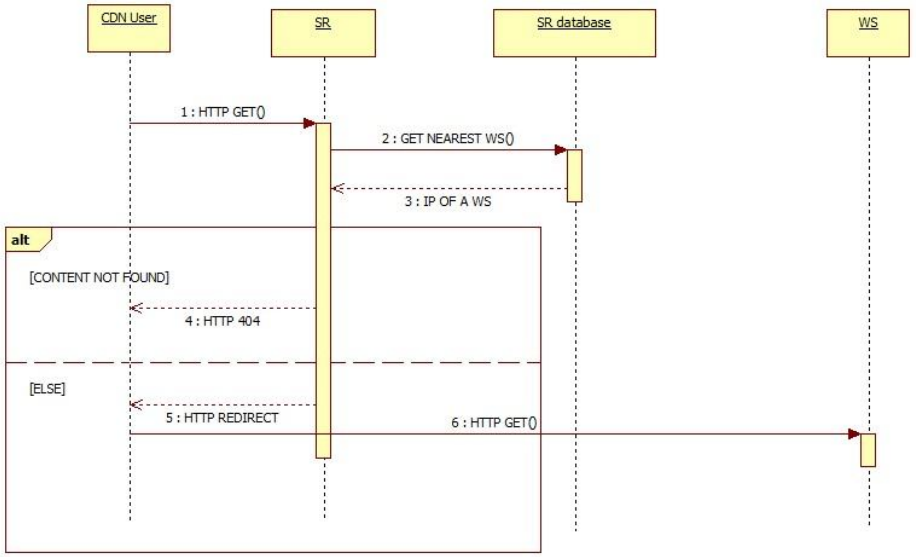

Figure 2. CDN sequence diagram

It is important to notice that WS is described as "the best" depends on the policy in the SR. For example it may depend on GEO-IP combined with WS's load and speed. There is a large number of metrics, but only those based on Quality of Service-related parameters have matured to a level that allows the delivery of comparable results.

2.3. PlanetLab

The Wikipedia entry for PlanetLab states: "PlanetLab is a group of computers available as a testbed for computer networking and distributed systems research. It was established in 2002 by Larry L. Peterson from Princeton, and as of June 2010 was composed of 1090 nodes at 507 sites worldwide. Each research project has 'a slice', or virtual machine access to a subset of the nodes."

We can define PlanetLab as an Internet simulator. Unfortunately it has the biggest disadvantage of Internet – it is neither repeatable nor isolated. In other words: every experiment is unique and other experiments performed at the same time have influence on our experiment. To obtain dependable results one must conduct several experiments and observe the average.

3. Environment Set-up

As we have described earlier, we created our CDN on PlanetLab network, which uses Linux-based OS's. Our environment consists of two main parts, i.e. Service Registry SR and Web Server WS.

1. On Service Registry we have installed some additional software:
    a. For handling user HTTP requests and redirections we have used Apache based WWW server (Lighttpd [4]), with enabled FAST-CGI and PHP support (to enable database access) – we installed following packages:
        i. lighttpd;
        ii. lighttpd-fastcgi;
        iii. php-cli.
    b. As a storage for information about network metrics and topology we used SQL database: PostgreSQL 8.2.11 [8]. To facilitate operations on storage we created a special api consisting of several SQL-based stored procedures. Database api is described in details in § 4 (Internal Architecture).
    c. Database was periodically updated by shell scripts configured in CronTab [2], which gathers data from Web Servers (WS) about current workload.
2. On Web Server (WS) we have installed the following software:
    a. For handling user HTTP requests we have used Apache-based WWW server (Lighttpd [4]) with enabled FAST-CGI support – hence the following packages were installed:
        i. lighttpd;
        ii. lighttpd-fastcgi.
3. We also used some general-purpose tools to facilitate performing tests:
    a. For running shell scripts we used PSSH [10]. This tool is similar to standard SSH client, the main difference is that it allows to run shell scripts in parallel (on multiple nodes simultaneously).

b. For transferring binary resources (files) we use PSCP [10]. This tool is an extend of SCP and similar to PSSH, as it allows to transfer one file to multiple nodes simultaneously.
c. For simulating user requests (requests for content) we use WGET [12].

4. Internal Architecture

In §3 (Environment set-up) we have described that CDN workload data is stored in SQL database installed on Service Registry (SR). The database is periodically updated with the data gathered from Web Servers (WS) by a shell script. The shell script loads data into database through special stored procedure. Shell script is scheduled as cron task.

User requests on Service Registry (SR) invoke stored procedure, which extracts from database location of the best Web Server (WS).

The main components of database api are as follows:

– FUNCTION add_new_weight_value(character varying, character varying, character varying) – SQL function (used by shell script) which adds new rate value for specified Web Server;
– FUNCTION recount_agregate_weight(character varying) – SQL function (used by shell script) which recounts weight of specified Web Server after adding new rate value;
– FUNCTION get_nearest_cdn(character varying) RETURNS character varying – SQL function (used by PHP script) which finds "the best" Web Server for the specific users.

5. Experiment

5.1. Service Registry

Our implementation was shell script which collects data from Web Servers. We collected TX rates extracted from Web Servers network interfaces.

5.2. Web Server

We created shell script which retrieves TX rate from ETH0 interface of Web Server. Data generated from this script serves as simple HTML page by PHP script. This PHP script is used by Service Registry to extract TX rate from Web Server.

5.3. Database

We need redirect implementation, which should ensure that probability that *n'th* Web Server will handle user request is higher for those Web Servers which have lower weight (have lower workload). Moreover, dependency between probability that *n'th* Web Server is chosen for client and weight should not be linear, it should be rather similar to *1/x*.

To implement such logic we used the following solution.

1. For each Web Server we calculate following value:

$$x_i = \frac{\sum_{k=1}^{n} w_k}{w_i} \quad . \tag{1}$$

2. We ordered ascent values computed in previous step:

$$x_j : j = 1..n \quad \forall k,l : \left(k \le l \wedge l \le n \wedge 1 \le k \Rightarrow x_k \le x_l\right) . \tag{2}$$

3. Based on previously calculated values we evaluated:

$$z_j = \frac{\sum_{m=1}^{j} x_m}{\sum_{k=1}^{n} x_k} \quad . \tag{3}$$

Definition of these values shows that following statement is true:

$$\max\left(z_j : j = 1...n\right) = 1 . \tag{4}$$

To calculated values, we add additional one:

$$z_0 = 0 \quad . \tag{5}$$

4. Having performed the above operations, we have *n+1* weights which all are in range [0, 1]. Moreover it can be proof that:

$$\forall k,l : (0 < k,l \le n \wedge x_k \le x_l \Rightarrow z_k - z_{k-1} \ge z_l - z_{l-1}) . \tag{6}$$

5. In the last step, we randomized a number from range [0,1) and looked for minimal value of z_j, which is greater than randomized number. Random number is needed to make better distribution between recalculations.

5.4. Experiment Results

We need to find metric to compare results. Our first idea was to use throughput. But the question was which one: the whole server throughput or just generated by our clients. Calculating metric using the whole server TX rate does not work, because it is not comparable on virtual environment. Especially that we do not know infrastructure behind it. Unfortunately calculating throughput generated only by our clients is also incorrect, because lot of throughput is generated on other virtual machines.

We decided to check how requests were distributed to servers. Having recalculated every weight, we ordered servers by weight and calculated how many requests went to the servers from the one with the least weight up to this with the most weight. The following table (table 1) presents result series by average, where *Weight i + 1 > Weight i* in the moment of redirection:

Tab. 1. Results of the experiment

Server weight	No. of requests	Percentage of total
Weight 1	8881	35,17%
Weight 2	6432	25,47%
Weight 3	4992	19,77%
Weight 4	4944	19,58%

We decided to include one more metric: request per server (table 2):

Tab. 2. Requests per server

Server	No. of requests	Percentage of total
A	7064	27,98%
B	5524	21,88%
C	6427	25,45%
D	6234	24,69%

The above results are almost ideal.

6. Conclusion

Out experiment gave us really good results. Servers are well-balanced. The difference between the most loaded and the least loaded is around 6%. Moreover the order list is stable between recalculations and still less-loaded servers take more than 60% of requests.

Building a well-balanced CDN one does not need difficult algorithms. Some of them are of course better than others. The most important thing is the following assumption: every node must be same or very similar to others. If not, balancing function must include differences between nodes.

Our set of experiments presents evolution of an balancing algorithm: from very simple to complex. Moreover we have created a technique which is working on such unpredictable environment as PlanetLab. As we have described above PlanetLab is a very good Internet simulation.

The presented technique can deal with following problems:

– Infrastructure – every PlanetLab node should be connected to the internet with the same 100Mb/s cable. We cannot check every node we use in experiment, but during our internal test at Warsaw University of Technology, we discovered network problems. As in real life we cannot be sure of the Internet speed.
– Virtualization – this is the problem with sharing resources. For example on one physical computer we have several virtual machines. They share CPU, disk speed, physical RAM and the most important in this experiments network card.

Nowadays when we start to use virtual computers more than the real ones, we have to deal with different problems than 10 years ago. The above methodology can be used in every virtual environment. Taking into account that implementation details probably have to be adapted in technical implementation.

7. Federation Architecture

7.1. Notation

For defining federation architecture we will use previously defined notation (see chapter CDN architecture) extended by one additional definition:

Mediator (M) – discovers and stores information about other federated CDN's. By information we mean current workload and available resources.

7.2. Architecture (Federation definition)

Based on provided notation we can define: "Federation is built on at least one CDN companied with Mediator (M)". Service Registry (SR) is still responsible for handling user requests, exactly like in basic CDN architecture. The only difference is that for each user request Service Registry (SR) additionally asks Mediator about availability of concrete content in different CDN's. Architecture of Federation is presented below on figure 3:

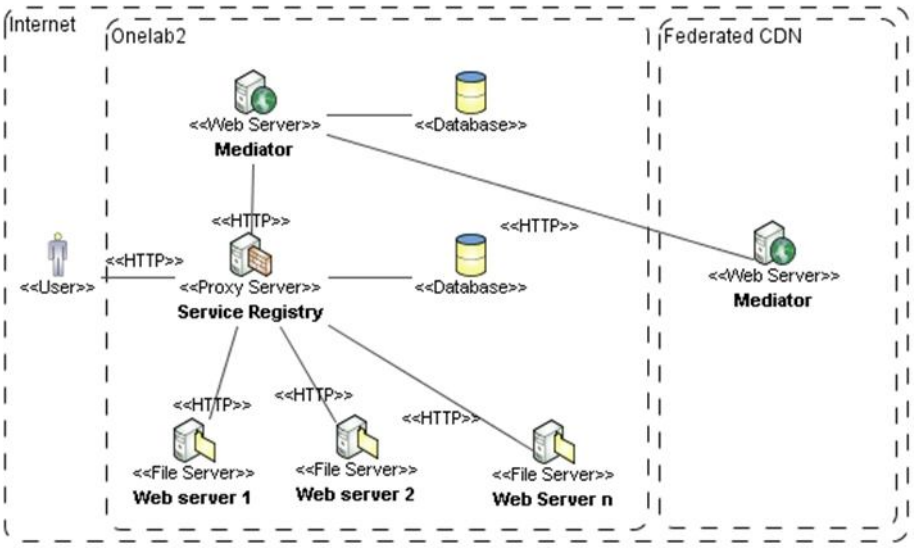

Figure 3. Federation architecture

The main idea is as follows (presented on figure 4 below):

- End user sends a request for some content to the Service Registry (SR);
- SR checks if CDN has particular content and if any of Web Servers is capable to handle the request. If one of these conditions is not met, it ask Mediator for redirection to "the best" CDN which has particular content;
- Otherwise, SR finds "the best" Web Server (WS) for this user;
- SR redirects the user to "the best" WS;
- The WS receives the request;
- The user downloads the requested content from WS.

The figure below presents the idea as a sequence of messages exchanged between objects involved.

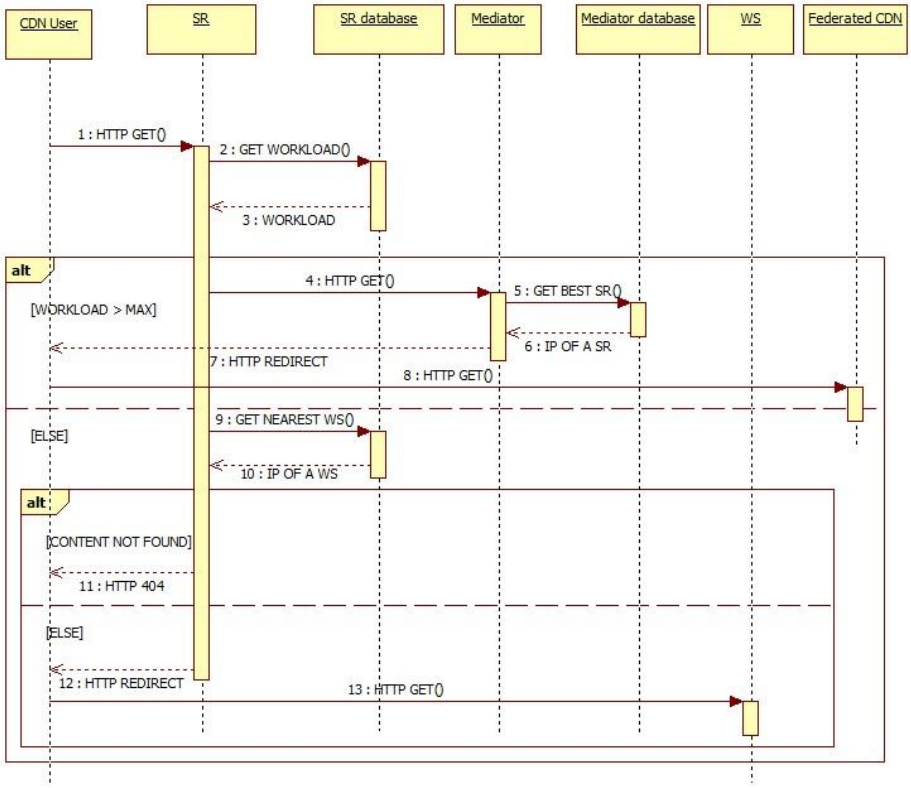

Figure 4. Interaction in federation presented on a sequence diagram

7.3. Possibilities of communication

Albert Einstein said that "Everything should be made as simple as possible, but not simpler", so we will present the most simple idea.

In our option protocol it is not important at all. In these days we can choose Web services, SOAP over REST, OData or even custom one. They have same possibilities. Much more interesting problem is information passing between Mediators. The information must:

- be reliable;
- contains weight of current CDN;
- contains content information.

We have two possibilities of collecting above information: pre-enter and post-enter. The second one is easier to implement, because we ask others mediator after user send request to SR. But this option extends time before user starts to download data. The first one is much better for user, but it is almost impossible to collect. Listing all the data about files stored in CDN can be huge, moreover it can change continuously. The only way it is something between: for popular content we can use pre enter method, for rest we have to use post-enter.

7.4. Example policies

The policy definition is very important. Unfortunately there is not a bad one, because policy will depend on environment. The examples of polices are:

- performance – network traffic and load;
- Geo-IP – nearest CDN is probably faster;
- failover – eliminating nodes which frequently fails;
- round robin;
- mixed of above.

8. Future work

Building CDN federation for tests in academic environment is very difficult, because it needs huge infrastructure. Fortunately well know companies like Microsoft or Google are starting to use such idea in their own datacentres. There is a new component for Windows Azure called Traffic Manager [5], which can be used as base for such experiments.

Building well-balanced CDN is easy, we just need mixed of infrastructure, metric and time, but building CDN federation need much more: academic and companies cooperation. But the idea of federation will be needed by customers. They download every day: games and apps for mobile devices, music and video, text and picture. The number of data grows every day, last year we had 9 billion devices connected to Internet, today estimation is 10 billion [2].

References

[1] ARLITT M., JIN T., *Workload characterization of the 1998 world Cup Web*, IEEE Network 14, pp. 30-37, 2000.

[2] *CRON documentation*, http://en.wikipedia.org/wiki/Cron.

[3] KHARIF O., *Average Household Has 5 Connected Devices, While Some Have 15-Plus*, 2012, http://go.bloomberg.com/tech-blog/2012-08-29-average-household-has-5-connected-devices-while-some-have-15-plus.

[4] *Lighttpd documentation*, http://www.lighttpd.net.

[5] *MSDN documentation*, 2012, http://msdn.microsoft.com/en us/library/windowsazure/hh744833.aspx.

[6] PATHAN M., BUYYA R., BROBERG J., *Internetworking of CDNs, in Content Delivery Networks*, Springer-Verlag Berlin/Heidelberg, pp. 389-413, 2008.

[7] PLUNKETT J., BAUMGARTNER F., *Jump: record 8m watch live on YouTube*, 2012, http://www.guardian.co.uk.

[8] *PostgreSQL documentation*, http://www.postgresql.org.

[9] *Princeton Web page*, http://comon.cs.princeton.edu.

[10] *PSSH project Web page*, http://www.theether.org/pssh.

[11] STAPP P., ZGADZAJ P., *Building well-balanced CDN*, Proceedings of the Federated Conference on Computer Science and Information Systems, pp. 679-683, 2012.

[12] *WGET documentation*, http://www.gnu.org/software/wget.

Budowa federacji CDN

STRESZCZENIE: Praca składa się z dwóch części. Pierwsza opisuje przykład sieci CDN w środowisku PlanetLab. W drugiej części omawiane są sposoby rozszerzenia prawidłowo zrównoważonych sieci CDN do federacji CDN.

SŁOWA KLUCZOWE: Content Distribution Network, CDN, PlanetLab, federacja.

Praca wpłynęła do redakcji: 23.01.2013