Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 9, September 2014, pg.291 – 296

RESEARCH ARTICLE

AN EFFICIENT DISTRIBUTED CONTROL LAW FOR LOAD BALANCING IN CONTENT DELIVERY NETWORKS

Y. Srinivasulu[1], P.Srinivas[2]

[1]M.Tech 2nd Year, Dept. of CSE, TITS, JNTU, HYDERABAD, INDIA

[2]Associate Professor, Dept. of CSE, TITS, JNTU, HYDERABAD, INDIA

[1] srinu.yagnam@gmail.com; [2] srinuvisitonly@yahoo.com

Abstract - Content Delivery Networks (CDN) aim at overcoming the inherent limitations of the Internet. The main concept at the basis of this technology is the delivery at edge points of the network, in proximity to the request areas, to improve the user's perceived performance while limiting the costs. This paper focuses on the main research areas in the field of CDN, pointing out the motivations, and analyzing the existing strategies for replica placement and management, server measurement, best fit replica selection and request redirection. In this paper, we face the challenging issue of defining and implementing an effective law for load balancing in Content Delivery Networks .A formal study of a CDN system, carried out through the exploitation of a fluid flow model characterization of the network of servers. This result is then leveraged in order to devise a novel distributed and time-continuous algorithm for load balancing, which is also reformulated in a time-discrete version.

Keywords — CDN's Fluid flow model, Load balancing Algorithm

I. INTRODUCTION

Content Delivery Network (CDN) represents a popular and useful solution to effectively support emerging Web applications by adopting a distributed overlay of servers. By replicating content on several servers, a CDN is capable to partially solve congestion issues due to high client request rates, thus reducing latency while at the same time increasing content availability. In this paper, we face the challenging issue of defining and implementing an effective law for load balancing in Content Delivery Networks. A formal study of a CDN

system, carried out through the exploitation of a fluid flow model characterization of the network of servers. This result is then leveraged in order to devise a novel distributed and time-continuous algorithm for load balancing, which is also reformulated in a time-discrete version.

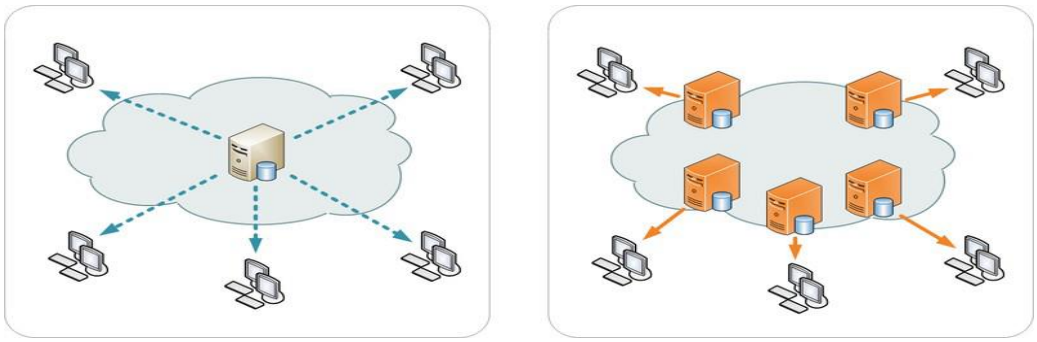

Fig: 1. Normal and Traditional Distribution

II. RELATED WORKS

Request routing in a CDN is usually concerned with the issue of properly distributing client requests in order to achieve load balancing among the servers involved in the distribution net- work. Several mechanisms have been proposed in the literature. They can usually be classified as either static or dynamic, de- pending on the policy adopted for server selection .Static algorithms select a server without relying on any information about the status of the system at decision time. Static algorithms do not need any data retrieval mechanism in the system, which means no communication overhead is introduced. These algorithms definitely represent the fastest solution since they do not adopt any sophisticated selection process. However, they are not able to effectively face anomalous events like flash crowds

Dynamic load-balancing strategies represent a valid alternative to static algorithms. Such approaches make use of information coming either from the network or from the servers in order to improve the request assignment process. The selection of the appropriate server is done through a collection and subsequent analysis of several parameters extracted from the network elements. Hence, a data exchange process among the servers is needed, which unavoidably incurs in a communication overhead.

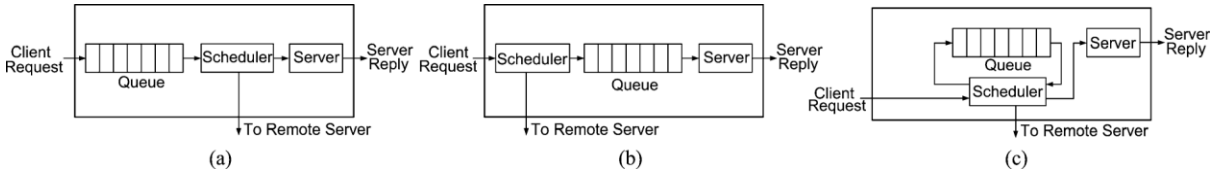

Fig: 2. Local load-balancing strategies. (a)Queue-adjustment.(b)Rate-adjustment.(c)Hybrid-adjustment

Depending on how the scheduler interacts with the other components of the node, it is possible to classify the balancing algorithms in three fundamental models a queue-adjustment model, a rate-adjustment model, and a hybrid-adjustment model. In a queue-adjustment strategy, the scheduler is located after the queue and just before the server. The scheduler might assign the request pulled out from the queue to either the local server or a remote server depending on the status of the system queues: If an unbalancing exists in the network with respect to the local server, it might assign part of the queued requests to the most un- loaded remote server. In this way, the algorithm tries to equally balance the requests in the system queues. It is clear that in order to achieve an effective load balancing, the scheduler needs to periodically retrieve information about remote queue lengths.

In a rate-adjustment model, instead the scheduler is located just before the local queue: Upon arrival of a new request, the scheduler decides whether to assign it to the local queue or send it to a remote server. Once a request is assigned to a local queue, no remote rescheduling is allowed. Such a strategy usually balances the request rate arriving at every node independently from the current state of the queue. No periodical information ex- change, indeed, is requested.

In a hybrid-adjustment strategy for load balancing, the scheduler is allowed to control both the incoming request rate at a node and the local queue length. Such an approach allows to have a more efficient load balancing in a very dynamic scenario, but at the same time it requires a more complex algorithm. In the context of a hybrid-adjustment mechanism, the queue-adjustment and the rate-adjustment might be considered respectively as a fine-grained and a coarse-grained process. Both centralized and distributed solutions present pros and cons depending on the considered scenario and the specific performance parameters evaluated.

In the following, we will describe the most common algorithms used for load balancing in a CDN. Such algorithms will be considered as benchmarks for the evaluation of the solution we propose in this paper. The simplest static algorithm is the Random balancing mechanism (RAND). In such a policy, the incoming requests are distributed to the servers in the network with a uniform probability. Another well-known static solution is the Round Robin algorithm (RR). This algorithm selects a different server for each incoming request in a cyclic mode. Each server is loaded with the same number of requests without making any assumption on the state, neither of the network nor of the servers.

The Least-Loaded algorithm (LL) is a well-known dynamic strategy for load balancing. It assigns the incoming client re- quest to the currently least loaded server. Such an approach is adopted in several commercial solutions. Unfortunately, it tends to rapidly saturate the least loaded server until a new message is propagated . Alternative solutions can rely on Response Time to select the server: The request is assigned to the server that shows the fastest response time.

The Two Random Choices algorithm (2RC) randomly chooses two servers and assigns the request to the least loaded one between them. A modified version of such an algorithm is the Next-Neighbour Load Sharing. Instead of selecting two random servers, this algorithm just randomly selects one server and assigns the request to either that server or its neighbour based on their respective loads (the least loaded server is chosen).

III.LOAD-BALANCED CDN: MODEL FORMULATION

In this section, we will introduce a continuous model of a CDN infrastructure, used to design a novel load-balancing law. The CDN can be considered as a set of servers each with its own queue. We assume a fluid model approximation for the dynamic behaviour of each queue. We extend this model also to the overall CDN system. Such approximation of a stochastic system

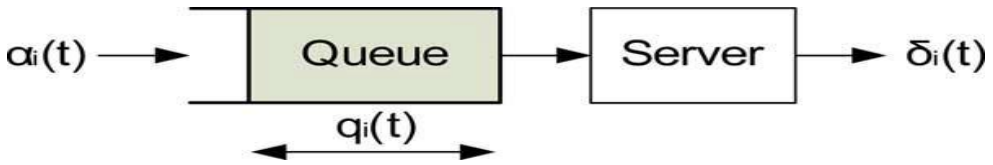

Fig: 3. Fluid queue model.

Actually, this approximation cannot be exploited in a real scenario: The requests arrive and leave the server at discrete times (Fig:5), hence in a given time interval, a discrete number of re- quests arrives at and departs from each server in the system case in a real packet network where the processing of arriving requests is not continuous over time.. The objective is to derive an algorithm that presents the main features of the proposed load-balancing law and arrives at the same results in terms of system equilibrium through proper balancing of servers' loads, as assessed by Lemma

IV.DISTRIBUTED LOAD-BALANCING ALGORITHM

In this section, we want to derive a new distributed algorithm for request balancing that exploits the results presented in Section III. First of all, we observe that it is a hard task to define a strategy in a real CDN environment that is completely compliant with the model proposed. As a first consideration, such a model deals with continuous-time systems, which is not exactly the equal to the traffic received at node from node if no re-quests are lost during the redirection process.

A. Algorithm Description

The implemented algorithm consists of two independent parts: a procedure that is in charge of updating the status of the neighbours' load, and a mechanism representing the core of the algorithm, which is in charge of distributing requests to a node's neighbours based on servers. In the pseudo code of the algorithm is reported. Even though the communication protocol used for status in-formation exchange is fundamental for the balancing process, in this paper we will not focus on it. Indeed, for our simulation tests, we implemented a specific mechanism:

We extended the HTTP protocol with a new message, called CDN, which is periodically exchanged among neighbouring peers to carry information about the current load status of the sending node. Naturally, a common update interval should be adopted to guarantee synchronization among all interacting peers. For this purpose, a number of alternative solutions can be put into place, in which are nonetheless out of the scope of the present work. Every second, the server sends its status information to its neighbours and, at the same time, waits for their information. After a well-defined interval, the server launches the status up- date process. We suppose all the information about peers' load is already available during such a process.

```
// peer status update
prob_space[0]=0; load_diff = 0; load_diff_sum = 0;
for(j=1; j<=n; j++){
    if(load_i - peer[j].load){
       load_diff = load_i - peer[j].load;
       //insert the new difference
       build_prob_space(load_diff, prob_space);
       load_diff_sum = load_diff_sum + load_diff;
    }
    //normalize the vector elements
    update_prob_space(load_diff_sum, prob_space);
}

// balancing process
if(prob_space[] == NULL) //no neighbors with lower load
    //serve locally the request
    serve_request();
else{
    float x = rand(); //random number generator
    int req_sent = 0; int i = 0;
    while(prob_space[i] == 1 or req_sent == 1){
        if(prob_space[i-1] <= x < prob_space[i]){
            //send request to the chosen peer
            send_to(peer[i-1].addr);
            req_sent = 1;
        }
        i++;
    }
}
```

Fig : 4 . Pseudo code description of the proposed algorithm.

V. SYSTEM EVALUATION

A. Balancing Performance

The simulations for the comparative analysis have been carried out using the network topology of Fig 5. We supposed to have 10 servers connected in the overlay, as well as 10 clients, each of them connected to a single server. We model each server as an M/M/1 queue with service rate ,and the generation requests from client as a Poisson process with arrival.
Though in this section, we exclusively want to provide a quantitative evaluation of the solution proposed with respect to the existing algorithms. We will demonstrate that the results herein achieved can be extended to larger scale topologies due to the high scalability of our solution. We implemented both the Random (RAND) and the Round Robin (RR) static algorithms, as well as the Least Loaded (LL)and the Two Random Choices (2RC)dynamic algorithms to make a comparison to our solution [Control-Law Balancing (CLB)].

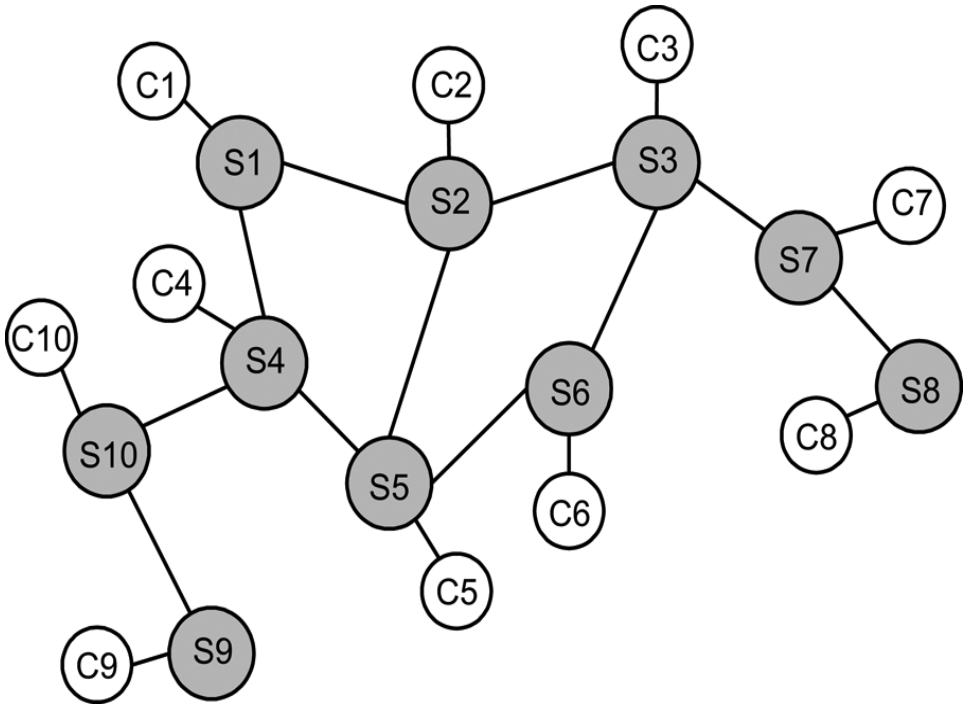

Fig: 5. Simulation topology

Then, for each algorithm, we first evaluated each server's queue length behaviour over time, together with the average value among all servers. Such a parameter represents an excellent indicator of the request distribution degree achieved by the CDN. Another important parameter is the Response Time (RT), which evaluates the efficiency of the algorithm in terms of end-user's satisfaction. For such a parameter, we evaluated both the average value and the standard deviation.

We also introduce an Unbalancing Index to estimate the capability of the algorithms to effectively balance requests among the available servers. Such an index is computed as the standard deviation of queue lengths of all the servers over time; clearly, the lower such value, the better the balancing result. Finally, since some of the proposed mechanisms provide multiple redirections, we also considered a parameter associated with communication overhead due to the redirection of a single request. Such a parameter is computed as the ratio of the number of requests due to redirections to the overall number of requests injected into the system.

As expected, static mechanisms provide worse performance since servers' queue lengths exhibit unpredictable behaviours due to a lack of knowledge about the real status of the server loads. On the other hand, dynamic mechanisms provide better behaviours, and in particular, our solution clearly achieves the best performance since it limits both the number of en queued requests and their oscillations over time, thus reducing the impact

on delay jitter. This confirms the effectiveness of the pro- posed mechanism, as well as its capability to fairly distribute load among the servers crowd.

On the other hand, the LL and the CLB approaches both react quite effectively to the transient abnormal conditions by quickly bringing back queue occupancies to their steady-state levels. However, this is achieved by the CLB with a more fair balancing among the available servers, as it is further confirmed by the analysis of the unbalancing index in Table V. In fact, in such a table we report the values of the unbalancing index analysis for both the normal and the flash-crowd scenarios. We point out once again the low degree of unbalancing exhibited by our solution with respect to the evaluated counterparts. Such a result confirms that the algorithm provides an optimized balancing mechanism.

B. Scalability Analysis

Before providing the testing results, we briefly discuss the scalability properties of the algorithm in terms of overhead introduced by the status update process. By adopting a local data exchange, we can considerably reduce the amount of overhead the rate for each interval with an increasing number of nodes. In particular, the results for exactly match the value estimated by formula.

Furthermore, the capability of our solution to properly scale is also evaluated by analyzing the impact of an increasing request load on the CDN in terms of response time, which, as already said, does represent a very good measure of the Quality of Experience of the CDN users. In particular, we have progressively in- creased the request rate while maintaining a fixed service rate at all servers in the network. Furthermore, we have also considered increasing network topology sizes. We have adopted an initial request rate and a service rate.

VI. CONCLUSION

In this paper we presented a novel load-balancing law for co- operative CDN networks. We first defined a model of such networks based on a fluid flow characterization. We hence moved to the definition of an algorithm that aims at achieving load balancing in the network by removing local queue instability conditions through redistribution of potential excess traffic to the set of neighbours of the congested server.

In future the problem solution model may be extended to live mobile networks where the distribution and congestion is inherent. The model is well suited for multiple application scenarios where users request continuously for sudden bursts and flash crowds occur. This is for generic work model implementation in the future.

REFERENCES

[1] S. Manfredi, F. Oliviero, and S. P. Romano, "Distributed management for load balancing in content delivery networks," in *Proc. IEEE GLOBECOM Workshop*, Miami, FL, Dec. 2010, pp. 579–583.
[2] H. Yin, X. Liu, G. Min, and C. Lin, "Content delivery networks: A Bridge between emerging applications and future IP networks,"*IEEE New.*, vol. 24, no. 4, pp. 52–56, Jul.–Aug. 2010.
[3] J. D. Pineda and C. P. Salvador, "On using content delivery networks to improve MOG performance," *Int. J. Adv. Media Commun.*,vol. 4,no. 2, pp. 182–201, Mar. 2010.
[4] D. D. Sorte,M. Femminella, A. Parisi, and G.Reali, "Network delivery of live events in a digital cinema scenario," in *Proc.ONDM*, Mar. 2008, pp. 1–6.
[5] Akamai, "Akamai," 2011 [Online]. Available: http://www.akamai.com/index.html
[6] Limelight Networks, "Limelight Networks," 2011 [Online]. Available:http://.uk.llnw.com
[7] CDNetworks, "CDNetworks," 2011 [Online]. Available: http://www.us.cdnetworks.com/index.php
[8] Coral, "The Coral Content Distribution Network," 2004 [Online].Available: http://www.coralcdn.org
[9] Network Systems Group, "Projects," Princeton University, Princeton, NJ, 2008 [Online]. Available: http://nsg.cs.princeton.edu/projects

