ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14)

Organized by

Department of CSE, JayShriram Group of Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

A Scalable Approach for Effective Content Delivery Using Enhanced Distributed Load Balancing Mechanism

D.Amrita[1], A.Aruna[2]

II- M. Tech, Department of IT, SNS College of Engineering, Coimbatore, Tamil Nadu, India[1]

AP/IT, Department of IT, SNS College of Engineering, Coimbatore, Tamil Nadu, India[2]

ABSTRACT: The challenging issue of defining and implementing an effective law for load balancing in Content Delivery Networks (CDNs). We base our proposal on a formal study of a CDN system, carried out through the exploitation of a fluid flow model characterization of the network of SERVERS. Starting from such characterization, we derive and prove a lemma about the network queues equilibrium. This result is then lever- aged in order to devise a novel distributed and time-continuous algorithm for load balancing, which is also reformulated in a time-discrete version. The discrete formulation of the proposed balancing law is eventually discussed in terms of its actual implementation in a real-world scenario. Finally, the overall approach is validated by means of simulations.

KEYWORDS: Content Delivery Network (CDN), control theory, request balancing.

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 [2]–[4]. 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.
Usually, a CDN consists of an original server (called back-end server) containing new data to be diffused, together with one or more distribution servers, called surrogate servers. Periodically, the surrogate servers are actively updated by the back-end server. Surrogate servers are typically used to store static data, while dynamic information (i.e., data that change in time) is just stored in a small number of back-end servers. In some typical scenarios, there is a server called redirector, which dynamically redirects client requests based on selected policies

The most important performance improvements derived from the adoption of such a network concern two aspects 1) overall system throughput, that is, the average number of requests served in a time unit (optimized also on the basis of the processing capabilities of the available servers); 2) response time experienced by clients after issuing a request. The decision process about these two aspects could be in contraposition. As an example, a "better response time" server is usually chosen based on geographical distance from the client, i.e., network proximity; on the other hand, the overall system throughput is typically optimized through load balancing across a set of servers. Although the exact combination of factors employed by commercial systems is not clearly defined in the literature, evidence suggests that the scale is tipped in favor of reducing response time.

A critical component of a CDN architecture is the request routing mechanism. It allows to direct users' requests for a content to the appropriate server based on a specified set of parameters. The proximity principle, by means of which a request is always served by the server that is closest to the client, can sometimes fail. Indeed, the routing process associated with a request might take into account several parameters (like traffic load, bandwidth, and servers' computational capabilities) in order to provide the best performance in terms of time of service, delay, etc. Furthermore, an effective request routing mechanism should be able to face temporary, and potentially localized, high request rates (the so-called flash crowds) in order to avoid affecting the quality of service perceived by other users.

Depending on the network layers and mechanisms involved in the process, generally request routing techniques can be classified in DNS request routing, transport-layer request routing, and application-layer request routing. With a DNS-based approach, a specialized DNS server is able to provide a request-balancing mechanism based on well-defined policies and metrics. For every address resolution request received, the DNS server selects the most appropriate surrogate server in a cluster of available servers and replies to the client with both the selected IP address and a time-to-live (TTL). The latter allows to define a period of validity for the mapping process. Typical implementations of this approach can provide either a single surrogate address or a record of multiple surrogate addresses, in the last case leaving to the client the choice of the server to contact (e.g., in a round-robin fashion).

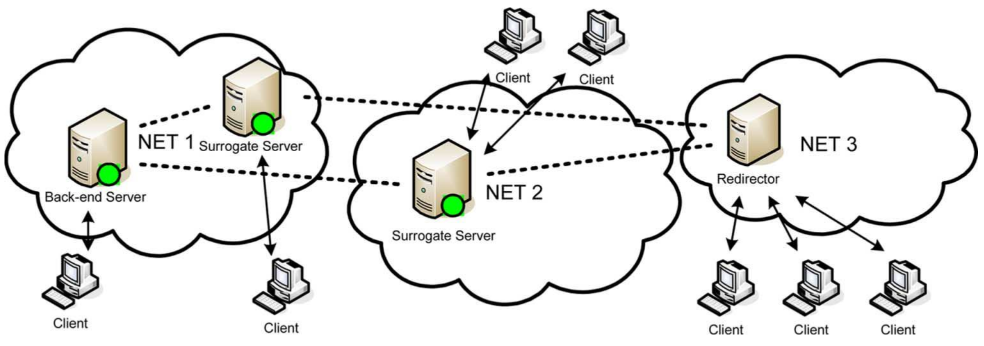

Figure. 1. Content Delivery Network.

With transport-layer request routing, a layer-4 switch usually inspects information contained in the request header in order to select the most appropriate surrogate server. Information about the client's IP address and port (and more generally all layer-4 protocol data) can be analyzed. Specific policies and traffic metrics have been defined for a correct server selection. Generally, the routing to the server is achieved either by rewriting the IP destination of each incoming packet, or by a packet-tunneling mechanism, or by a forwarding mechanism at the MAC layer.

With application-layer request routing, the task of selecting the surrogate server is typically carried out by a layer-7 application, or by the contacted Web server itself. In particular, in the presence of a Web-server routing mechanism, the server can decide to either serve or redirect a client request to a remote node. Differently from the previous mechanism, which usually needs a centralized element, a Web-server routing solution is usually designed in a distributed fashion. URL rewriting and HTTP redirection are typical solutions based on this approach. In the former case, a contacted server can dynamically change the links of embedded objects in a requested page in order to let them point to other nodes. The latter technique instead exploits the redirection mechanism of the HTTP protocol to appropriately balance the load on several nodes.

In this paper, we will focus our attention on the application- layer request routing mechanism. More precisely, we will pro- vide a solution for load balancing in the context of the HTTP redirection approaches. In most of the papers available in the literature, the design of a proper network management law is carried out by assuming a continuous fluid flow model of the network. Validation and testing are then provided by exploiting a discrete packet simulator (e.g., ns-2, Op net, etc.) in order to take into account the effects of discretization and nonlinear nature occurring in practice. This approach is widely used in the communication and control communities

In a similar way, in this paper we first design a suitable load-balancing law that assures equilibrium of the queues in a balanced CDN by using a fluid flow model for the network of servers. Then, we discuss the most notable implementation issues associated with the proposed load-balancing strategy. Finally, we validate our model in more realistic scenarios by means of ns-2 simulations. We present a new mechanism for redirecting incoming client requests to the most appropriate server, thus balancing the overall system requests load. Our mechanism leverages local balancing in order to achieve global balancing. This is carried out through a periodic interaction among the system nodes.

II. RELATED WORK

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 [20].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.

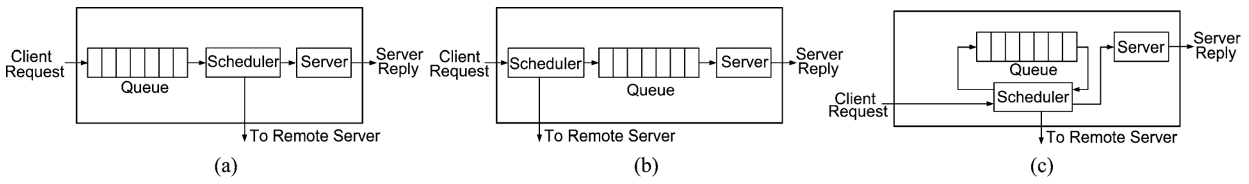

Figure.2.Localload-balancingstrategies.(a)Queue-adjustment.(b)Rate-adjustment.(c)Hybrid-adjustment

The redirection mechanisms can be implemented either in a centralized or in a distributed way In the former, a centralized element, usually called dispatcher, intercepts all the re- quests generated into a well-known domain, for example an autonomous system, and redirects them to the appropriate server into the network by means of either a static or a dynamic algorithm. Such an approach is usually adopted by commercial CDN solutions. With a distributed redirection mechanism, instead any server receiving a request can either serve it or redistribute it to another server based on an appropriate (static or dynamic) load-balancing solution.

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.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14)

Organized by

Department Of CSE, JayaShriram Group Of Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

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 [23]. 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-Neighbor 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 neighbor 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 behavior of each queue. We extend this model also to the overall CDN system. Such approximation of a stochastic system

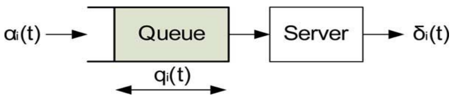

Figure. 3. Fluid queue model.

Actually, this approximation cannot be exploited in a real scenario: The requests arrive and leave the server at discrete times (Fig. 4), 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. For this reason, in the following of this section, we focus on the control law described by (5). 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 neighbors' load, and a mechanism representing the core of the algorithm, which is in charge of distributing requests to a node's neighbors based on (15). 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 neighboring 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, which are nonetheless out of the scope of the present work.

Every seconds, the server sends its status information to its neighbors 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.

V. SYSTEM EVALUATION

A. Balancing Performance

The simulations for the comparative analysis have been carried out using the network topology of Fig. 7.We suppose 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.

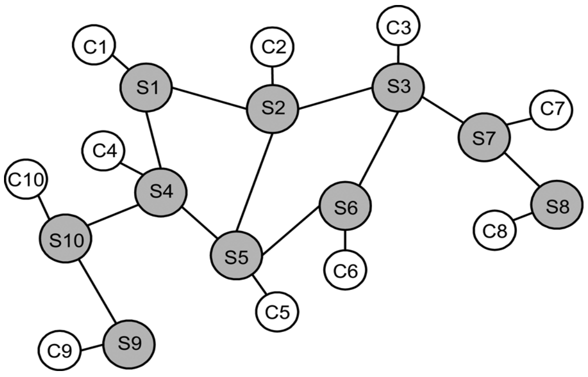

Figure. 4. Simulation topology

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14)

Organized by

Department Of CSE, JayaShriram Group Of Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

Though in this section, we exclusively want to provide a quantitative evaluation of the solution proposed with respect to the existing algorithms. In Section V-B, 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)].

Then, for each algorithm, we first evaluated each server's queue length behavior 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 behaviors due to a lack of knowledge about the real status of the server loads. On the other hand, dynamic mechanisms provide better behaviors, 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 T 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. DISCUSSION ON POTENTIAL TUNING STRATEGIES

A. Effects of Queue Threshold on Algorithm Performance

The algorithm we devised tends to balance load in the CDN, independently from the fact that a specific server might not be overloaded at a certain point in time. Simulation results have shown that response time figures always outperform the other algorithms we analyzed. Nonetheless, with our approach, as long as a server has neighbors with lower load, incoming re- quests are redirected among them even when the server itself is under loaded. Therefore, redirections can happen very frequently, which might have an impact on response time. We hence decided to evaluate the possibility of better striking the balance between equalizing queue occupancies at the servers on one side and reducing the number of redirections on the other. With this aim in mind, we configured our simulator in such a way as to impose a lower limit on the queue length, below which no redirection mechanism is applied. With this configuration in place, we ran a whole new set of simulations and derived the main performance evaluation figures. Results are shown in Fig. 15 for what concerns Response Time, and in Fig. 16 for what concerns the Unbalancing Index.

As the figure clearly indicates, more than 95% of the requests receive less than eight redirections. We have then carried out a whole new set of simulations after having introduced the possibility to explicitly impose a limit on the overall amount of redirections that each server can make. Based on the above consideration about the request redirection frequency, we expect that a redirection threshold over the detected bound of 8 would prove almost useless in the scenario analyzed in the paper.

Given the definitions above, we report in Table VIII the results of an extensive set of simulations aimed at evaluating the settling time in different configuration scenarios (in terms of queue occupancies at time 0) and under different load conditions (as before, 33% of the overall capacity, 66% of the overall capacity and saturation).

VII. CONCLUSION AND FUTURE WORK

Presented a novel load-balancing law for co- operative CDN networks. We first defined a model of such net- works 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 neighbors of the congested server.

REFFERENCES

[1] S. Manfredi, F. Oliviero, and S. P. Romano, "Distributed manage- ment 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 Netw., 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

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14)

Organized by

Department Of CSE, JayaShriram Group Of Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

[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
[10] A. Barbir, B. Cain, and R. Nair, "Known content network (CN) re- quest-routing mechanisms," IETF, RFC 3568 Internet Draft, Jul. 2003 [Online]. Available: http://tools.ietf.org/html/rfc3568
[11] T. Brisco, "DNS support for load balancing," IETF, RFC 1794 In- ternet Draft, Apr. 1995 [Online]. Available: http://www.faqs.org/rfcs/rfc1794.html
[12] M. Colajanni, P. S. Yu, and D. M. Dias, "Analysis of task assignment policies in scalable distributed Web-server systems," IEEE Trans. Parallel Distrib. Syst., vol. 9, no. 6, pp. 585–600, Jun. 1998.
[13] D. M. Dias, W. Kish, R. Mukherjee, and R. Tewari, "A scalable and highly available Web server," in Proc. IEEE Comput. Conf., Feb. 1996, pp. 85–92.