Synthetic Instances for a Management Problem in Content Distribution Networks

Tiago Neves[a,b,*], Luiz Satoru Ochi[b], Célio Abuquerque[b]

[a]*Departamento de Ciências Exatas, Universidade Federal Fluminense-Av. dos Trabalhadores 420, Volta Redonda/RJ, Brazil. 27255-250*
[b]*Instituto de Computação, Universidade Federal Fluminense-R Passo da Pátria 156, Bloco E, São Domingos, Niterói/RJ, Brazil. 24210-240*

Abstract

The Replica Placement and Request Distribution Problem (RPRDP) is an optimization problem related to Content Distribution Networks (CDN) management. The objective is to reduce the operational costs of CDN providers without violating servers' and clients' constraints. To the best of authors knowledge, there are no instances available for this problem. This technical report presents four classes of instances for RPRDP and explains, in detail, how these instances were built.

Keywords: Content Distribution Networks, Replica Placement and Request Distribution Problem, Instances

1. Introduction

A Content Distribution Network (CDN) is a well known kind of overlay network that is used to reduce the network congestion by replicating contents in servers that are geographically close to clients. One of several problems related to CDN management is the Replica Placement Problem (RPP), that consists in finding the best servers to place the contents in order to reduce the total traffic in the network.

The focus of this technical report is the Replica Placement and Request Distribution Problem (RPRDP) that is a dynamic and online problem whose

*Corresponding author

Email addresses: tneves@ic.uff.br (Tiago Neves), satoru@ic.uff.br (Luiz Satoru Ochi), celio@ic.uff.br (Célio Abuquerque)

objectives are to find the best position for the content replicas and to distribute the requests through the servers, aiming at reducing the network traffic without violating clients Quality of Service (QoS) constraints. Servers have limited capacity in bandwidth and disk space. The QoS constraints are given by a required minimal bandwidth and a maximum delay in which a client's request must be served. In the beginning, the contents are positioned only in their origin servers. As clients' requests arrive in the CDN system, contents may be replicated over the network and such requests may be redistributed through the servers taking into account QoS constraints. A server is allowed to handle a request partially or totally as long as it has a replica of the desired content, i.e., a request may be served by several servers at the same time. The time is divided in periods, and the optimization process works over all periods (horizon). Besides, clients may not have enough bandwidth to download a content in a single period of time, meaning that several periods are necessary to handle a single request. Moreover, since the overlay network[1] may lack the resources to handle all the requests within the desired QoS, some requests may have the QoS requirements partially fulfilled but every time a request is not handled within the QoS specified a cost is paid.

RPRDP is a dynamic and online problem, meaning that costs of communication between servers can change, new contents and requests can come up and the future scenario is not known *a priori*.

To the best of our knowledge, there are no instances available for the RPP and for the RPRDP, thus, we created a set of 80 synthetic instances that are based on the available literature on the subject. The instances include many realistic details of the problem such as minimal and maximum bandwidth for clients, limited capacity on servers, QoS requirements of clients, etc. The objective of this technical report is to explain how such instances were created so that the created instances can be used in other experiments. The instances are divided in four classes : A, B, C and D . Each class has exactly 20 instances, 5 for each number of servers.

2. Problem Description

As mentioned before, the objective of the RPDRP is to find the best servers to place contents replicas and to define how many and which servers will handle each request over an horizon so that CDN providers costs are minimized, servers constraints are not violated and requests QoS require-

ments satisfied when possible. in order to better explain the problem, let the optimization horizon be divided in units called periods of time. The entire set of periods of time is then the optimization horizon and is denoted by T. Also let S be the set of available servers, R be the set of requests and C be the set of contents. Each content $c \in C$ has a size and a life time (submission period and removal period) associated to it. Each server $j \in S$ has a storage capacity and a bandwidth limit. There is an overlay network [1] interconnecting the servers and between each pair of servers j and l there are two delay values (from j to l and from l to j) and one Round Trip Time (RTT), that corresponds the sum of such delays. We assume that each client i) establishes a connection with the CDN system, ii) make a single request and iii) leaves the CDN system after his request is completely handled. Associated to each request $i \in R$ there is a desired content in C, an arrival period in T, a server $o_i \in S$ that represents the server to which the client is connected, a local delay ld_i that represents the distance between o_i and client's computer. There are also a delay limit TD_i that represents the maximum delay tolerated and two values of bandwidth, minimum (nb_i) and maximum (xb_i), associated to each request. Since client's bandwidth is limited, it is not always possible to deliver the desired content in a single period, meaning that several periods are typically required to fully handle a request. In this paper, we use the maximum bandwidth of the client as demand, meaning that whenever is possible, a client is served in its maximum bandwidth. If some part of the bandwidth a client request, say i, is not used in some period t, it means that i is not handled in its full capacity in period t and this missing part is called *backlog* of request i in period t. In order to achieve clients maximum bandwidth, multiple servers are allowed to handle the same request in the same period. There are costs associated for handling a request by each server on the CDN and there are also costs associated to the replication of the contents. Since the objective of the RPRDP is to reduce the operational costs of CDN providers and also improve the quality perceived by end users, the objective function of the problem can be expressed as follows:

$$Min \sum_{i \in R} \sum_{j \in S} \sum_{t \in T} c_{ijt} x_{ijt} + \sum_{i \in R} \sum_{t \in T} p_{it} b_{it} + \sum_{k \in C} \sum_{j \in S} \sum_{l \in S} \sum_{t \in T} h_{kjlt} w_{kjlt} \quad (1)$$

Where x_{ijt} is a continuous variable that represents the fraction of content

asked by request i handled by server j in period t; b_{it} represents the backlog of request i in period t, w_{kjlt} assumes 1 only if server j replicates content k from server l in period t; c_{ijt} represents the cost of handling request i by server j, in period t. This cost calculated based on the fitness of each server, thus, the servers that can handle a request in a better way have lower costs; p_{it} is the backlog penalty paid for request i in period t. This penalty is greater the cost for handling request i in period t in any server; h_{kjlt} is the cost that server j pays for downloading content k from server l in period t.

In the RPRDP every request must be fully handled, servers bandwidth and storage capacity must be verified and at least one replica of each content must exist in each period (except for those contents that were already removed, or not submitted yet in the considered period). Besides, the backlogged portion of a request must be handled in some of the periods ahead and servers are allowed to handle requests if, and only if, they have replicas of the desired content. The reader interested in a more detailed description of the RPRDP is referred to [2], where the meaning of each variable and the costs involved are better explained.

3. Duration and Number of Periods

We desingned the instances so that each period could represent approximatly 60 seconds on all instances. The number of periods varies according to the classes. Class A intances have 15 periods. Classes B, C and D have 35 periods.

4. Network Topologies

The instances creation process is divided in stages. The first stage is to create the network topologies for the CDN servers. To do so, we used the well known BRITE [3] topology generator. We chose BRITE because it is free, it is capable of generating many different kinds of topologies and because it is well known by the scientific community. We decided to create eight topologies using the *Waxman* model for autonomous systems. The topologies created have, respectively, 10, 20, 30, 50, 100, 200, 300, 400 and 500 servers. Two parameters used in the *Waxman* model were intentionally chosen: The number of nodes in the topology (parameter N) and the number of connections of each node (parameter M). The value of N was set in order to create topologies containing the number of servers mentioned. The M

value was set to 2 on topology with 10 servers and to 3 on the other cases. All other parameters were used in its default value (according to BRITE definition).

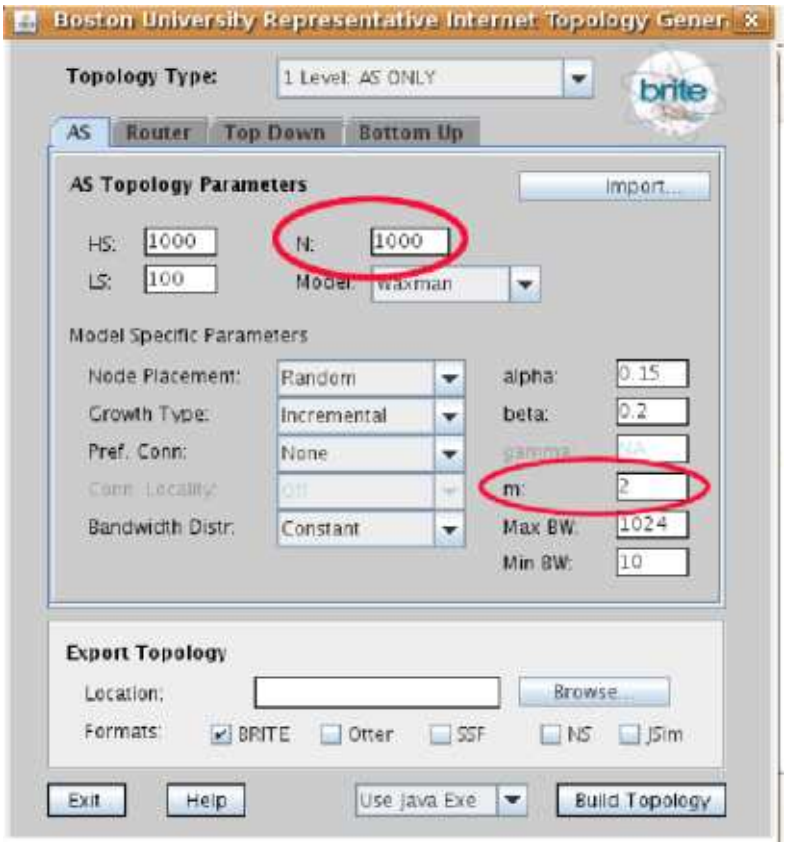

Figure 1: BRITE Topology Generator

Figure 1 shows BRITE window with the values used to create the instances for the RPRDP. The parameters N and M are marked because their values were changed as explained before. Note that the number of server chosen is greater than the ones used in [4, 5]

5. Network Dynamics

Second stage is to generate the network data for all periods. In order to model the problem as described in Section **??**, we need three information for

each arch linking two servers j and l in each period: i) the *Delay* between j and l, ii) the *Delay* between l and j and the *Round Trip Time* (RTT) j and l. Since *Delay* and RTT are basic concepts of network computing we shall not define such concepts here. For more explanation about *Delay* and RTT the reader is referred to [1].

To determine the initial *Delays* for each link we used values that varies between 60 and 100 milliseconds according to Uniform distribution. For instances of classes A, B and C the values of $Delay_{ij}$ (the delay between server i and server j) and $Delay_{ji}$ are the same. For instances of class D the values of $Delay_{ij}$ and $Delay_{ji}$ are different. The definition of RTT is the same for all classes: $RTT_{ij} = Delay_{ij} + Delay_{ji}$. These values of *Delay* and RTT are referent to the first period of time considered.

After the *delays* and RTT are set correctly for each link in the topology it is necessary to obtain the total *delay* and total RTT for each pair of servers of the CDN system. To do so, Dijkstra algorithm [6] is used to obtain the minimal *delay* and the minimal RTT for each pair of servers. As the CDN is actually an overlay network it is reasonable to use the notation *Delay* and RTT for each pair of servers in the CDN system. Although the path between to server i and j is composed by several links, an overlay network allows to create virtual links between each pair of servers. Thus the delay of the virtual link between i and j is the sum of the delays of all links in the path from i to j in the topology.

Every time a period is multiple of 5 one link in the topology taken according to Uniform distribution and its delays and RTT are changed in the same way they were set in the first period. Note that changing one link in the topology can affect the delays and RTT of the whole network. We then use the Dijkstra algorithm again in order to update the values of RTT and *delay* between each pair of servers.

6. Servers

The third stage is to create servers data. To model the disk space, a Uniform distribution was used with values between 100 and 200 MB for instances of class A; 100 and 150 GB [7] for class B; 3 and 4 GB for class C; 2.5 and 3.2 GB for class D. The values of class A were arbitrarily chosen. Values for class B are based on the literature, values for classes C and D were arbitrarily chosen in order to reduce resources availability.

The bandwidth available for each server is chosen randomly between 1500 and 2000 MB per period on class A; 4000 and 4050 MB per period on classes B and C; 2300 and 2350 MB per period on class D. The values chosen for classes B and C are based on Brazilian real network providers logs available on the Internet on 2008. It is important to mention that each server has an unique identifier. Since class A is composed only by test instances, the bandwidth limits were chosen arbitrarily. As class D instances is supposed to have less resources available, we decided to use approximately 60% of the bandwidth used on instances of class B and C.

7. Contents

The fourth stage is to generate the contents information. To do so we must decide the number of permanent contents and the number of volatile contents. Permanent contents are those that are submitted in the first period and removed in the last period. The volatile contents are those that are submitted after the first period and removed before the last period. The number of permanent contents is different for each class: class A uses 3; Classes B and C use 10; Class D uses 12. The number of volatile contents is randomly chosen in different intervals according to the instance classes: class A uses the interval $[1, 3]$; Classes B and C use $[1, 5]$; Class D uses $[1, 7]$. The submission periods for the volatile contents are randomly chosen between periods 1 and 5, and the removal period is chosen randomly between $lastPeriod - 5$ and $lastPeriod$ for all instances.

The size of the contents in classes B, C and D was based on typical 40 minutes long videos available to download on the Internet. These values are set to be between 250 and 400 MB for classes B and C, between 350 and 450 for class D. Since class A is built only for tests the values used were arbitrarily chosen between 10 and 20 MB.

The last information needed to create the contents data is the origin server. This information represents the server where the content is first submitted. Each content must have only one server as origin although the same server can be chosen as origin for many servers. The origin server is chosen according to the Uniform distribution for all contents.

8. Requests

The last stage to create an instance is to generate the request information. This stage must be the last because the information about the servers and

about the contents are necessary here. The information about contents are necessary because it is known that some contents are more popular than others [8, 7] and because we can not create requests for contents that are not available yet in the current period nor for contents that were already removed from the CDN system. Information about the number of server is required as well since the greater the number of servers the greater the number of clients expected. Another reason for the fact that we need the number of servers to create the requests is that each requests has an origin server associated to it. This origin server is the client's access point to the CDN system. In other words is the CDN server to which the client is connected.

It is known that some contents are more popular than others and that it implies in a greater number of requests for such popular contents. In order to model this popularity differences we decided that the permanent contents should be more popular than the volatile contents and that, contents of lower id should be more popular than the higher id contents. This decision was made in order to establish the priority order of the contents, however, the real popularity of the contents (number of requests) in the instances is determined according to a Random Distribution based in such priority. To decide the number of requests per period of the instances we used different methodologies in each instance class. In class A we decide to use an arbitrary number of requests. In classes B, C and D the number of requests is given by the following equation: $req_t = rand(20, 25) \times numberOfServers/5$, where req_t is the total of requests in period t. Having calculated the number of requests for a period we must determine how we should divide such requests through the contents. According to [9] and [10] search requests for multiple keys tend to follow a distribution pattern that is similar to Zipf distribution [11]. In this distribution, most popular keys attract the great majority of the requests. In this sense, we decide that in instances of classes A, B and C, the number of requests of content k in a period t is given by $req_t^k = req_t \times (NC - (p_k - 1))/\sum_{s=1}^{NC} s$, where NC is the total number of contents and p_k is the position of content k in the popularity list of contents. For example, the most popular content has $p_k = 1$, the second most popular has $p_k = 2$ and so on. The especial case for the number of requests is the less popular content. We use the mentioned equation to determine the number of requests for all contents but one, the less popular one. After deciding the number of requests of all other contents, the less popular one take the remaining requests. For example, suppose that there are 100 requests and 3 contents. According to the given equations, the most popular content

gets 50 requests, the second most popular gets 33 and the third gets the remaining requests, i.e., 17 requests. It is important to say that in classes B and C, all contents have requests in all periods. On instances of class D there are no defined fraction of requests for each content. The requests are distributed randomly according to Zipf distribution, as described in [10] using the parameter $\alpha = 0.75$.

It is also necessary to associate a local delay for each request. These delays represent the distance between the client's computer and the CDN server to which such client is connected (origin server). The values for such delays are chosen according to the following intervals: $[50, 100]$ ms for instances of classes A, B and C, and $[250, 300]$ for class D. The QoS data for each request are maximum delay, minimum bandwidth and maximum bandwidth. The maximum delay used is randomly chosen in the interval $[400, 800]$ for all classes and is expressed in ms. The minimum bandwidth uses the following intervals: $[5, 6]$ MB per period ($bandwidth \times \delta$) on class A and $[40, 42]$ MB per period on classes B, C e D. Maximum bandwidth is chosen according to the following intervals: $[7, 10]$ MB per period on class A and $[45, 50]$ MB per period on classes B, C and D. An origin server is also attributed to each request according to the Uniform distribution.

9. Summary

Table 1 briefly presents the instances sets and some of the intervals used. Each set contains 5 instances and all values were chosen using uniform distribution on the respective intervals. First column shows the instance class and the number of servers. Column two depicts the server bandwidth interval in MB per period. Column three describes the server disk space interval in GB. Column four exposes the intervals for the number of contents. Column five exposes the size of the contents interval in MB. Column six shows the intervals for the number of requests. Columns seven and eight expose intervals, in MB per period, for the requests' minimum and maximum bandwidth, respectively.

10. Use in publications

As the instances set is always growing, we decide to mention here which instances were used in each publication.

1. [12] All instances of classes A,B and C.

Instance	Serv. Band	Serv. Disk	# Cont.	Cont. Size	# Req.	Min Band	Max Band
10A	[1500,2000]	[0.1,0.2]	[4,6]	[10,20]	[400,650]	[5,6]	[7,10]
10B	[4000,4050]	[100,150]	[11,15]	[250,400]	[600,700]	[40,42]	[45,50]
10C	[4000,4050]	[3,4]	[11,15]	[250,400]	[600,700]	[40,42]	[45,50]
10D	[2300,2350]	[2.5,3.2]	[13,17]	[350,450]	[600,700]	[40,42]	[45,50]
20A	[1500,2000]	[0.1,0.2]	[4,6]	[10,20]	[700,1400]	[5,6]	[7,10]
20B	[4000,4050]	[100,150]	[11,15]	[250,400]	[1200,1700]	[40,42]	[45,50]
20C	[4000,4050]	[3,4]	[11,15]	[250,400]	[1200,1700]	[40,42]	[45,50]
20D	[2300,2350]	[2.5,3.2]	[13,17]	[350,450]	[1200,1700]	[40,42]	[45,50]
30A	[1500,2000]	[0.1,0.2]	[4,6]	[10,20]	[1700,2400]	[5,6]	[7,10]
30B	[4000,4050]	[100,150]	[11,15]	[250,400]	[1900,2400]	[40,42]	[45,50]
30C	[4000,4050]	[3,4]	[11,15]	[250,400]	[1900,2400]	[40,42]	[45,50]
30D	[2300,2350]	[2.5,3.2]	[13,17]	[350,450]	[1900,2400]	[40,42]	[45,50]
50A	[1500,2000]	[0.1,0.2]	[4,6]	[10,20]	[3200,3800]	[5,6]	[7,10]
50B	[4000,4050]	[100,150]	[11,15]	[250,400]	[3200,3400]	[40,42]	[45,50]
50C	[4000,4050]	[3,4]	[11,15]	[250,400]	[3200,3400]	[40,42]	[45,50]
50D	[2300,2350]	[2.5,3.2]	[13,17]	[350,450]	[3200,3400]	[40,42]	[45,50]

Table 1: Instances intervals

2. [2] All instances of classes A,B and C.
3. Future publication will be included here.

11. Conclusions

This technical report presents how 80 synthetic instance to the RPRDP were created. The instances are available on the LABIC project [13] website. To the best of our knowledge, this set of instances is the first set that consider simultaneously several realistic characteristics such as QoS requisites, different server capacities and content dynamics available.

References

[1] J. Kurose, K. Ross, Computer Networking: a Top-Down Approach Featuring the Internet, Addison-Wesley, 2003.

[2] T. Neves, L. Drummond, L. Ochi, C. Albuquerque, E. Uchoa, Solving replica placement and request distribution in content distribution networks, Electronic Notes in Discrete Mathematics 36 (2010) 89–96.

[3] BRITE, World Wide Web, http://www.cs.bu.edu/brite/, 2007.

[4] W. Aioffi, G. Mateus, J. Almeida, A. Loureiro, Dynamic content distribution for mobile enterprise networks, IEEE Journal on Selected Areas in Communications 23 (2005) 2022–2031.

[5] O. Wolfson, S. Jajodia, Y. Huang, An adaptive data replication algorithm, ACM Transactions on Database Systems 22 (1997) 255–314.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to Algorithms, MIT Press and McGraw-Hill, second edition, 2001.

[7] X. Zhou, C.-Z. Xu, Efficient algorithms of video replication and placement on a cluster of streaming servers, Journal of Network and Computer Applications 30 (2007) 515–540.

[8] T. Bektas, O. Oguz, I. Ouveysi, Designing cost-effective content distribution networks, Computers & Operations Research 34 (2007) 2436–2449.

[9] T. Wauters, J. Coppens, B. Dhoedt, P. Demeester, Load balancing through efficient distributed content placement, in: Next Generation Internet Networks, pp. 99–105.

[10] L. Breslau, P. Cao, G. Phillips, S. Shenker, Web Caching and Zipf-like Distribuitions: Evidence and Implications, in: Proceedings of IEEE INFOCOM 99, pp. 126–134.

[11] G. K. Zipf, Relative Frequency as a Determinant of Phonetic Change, volume 40 of *Harvard Studies Classical Philology*, Harvard University Press, 1929.

[12] T. Neves, L. Drummond, E. Uchoa, L. Ochi, C. Albuquerque, Replicação e distribuição *Online* em redes de distribuição de conteúdos, in: Anais do XLI Simpósio Brasileiro de Pesquisa Operacional, pp. 2717–2727.

[13] LABIC, World Wide Web,http://labic.ic.uff.br/, 2005.