A Scalable Framework for Content Replication in Multicast-Based Content Distribution Networks

Yannis Matalas[1], Nikolaos D. Dragios[2], and George T. Karetsos[2]

[1]Digital Media & Internet Technologies Department,
Intracom Telecom, Athens, Greece
[2]School of Electrical and Computer Engineering,
National Technical University of Athens, Greece
imata@intracom.gr, ndragios@telecom.ntua.gr, karetsos@cs.ntua.gr

Abstract. This paper proposes a framework for replicating content in multicast-based CDNs. We focus on the design of a scalable and robust system that provides local availability and redundancy of content. The system takes on-line and distributed replication decisions on a per-object basis. The scalability and local redundancy is achieved by partitioning the overlay of surrogate servers into fully meshed groups. The proposed framework can incorporate any set of local metrics and constraints for deciding the placement of replicas, thus allowing the CDN designer to tune it to his specific deployment characteristics.

1 Introduction

Content Distribution Networks (CDNs) have become a common technology that enables content providers to distribute their popular content to a large number of users. Herein, we assume such a CDN system for distributing bulky files over a satellite network. Scalability, availability and efficiency of such a system are of vital importance especially in deployments that include a large number of receivers covering extended and distant regions. Efficient and scalable content distribution is achieved by applying: (a) multicast transmission, and (b) a distributed content replication algorithm that places content close to clients. The scalability of this algorithm is assisted by the partition of the CDN into relative small neighborhoods and the restriction of its scope within their bounds. At the same time, replication algorithm aims at providing content redundancy and load balancing in the CDN.

Different formulations for the problem of content placement in CDNs have been proposed in the literature, each one focusing on different objectives. From our perspective, content placement can be broken in two sub-problems. The first one, referred as *server placement* problem, is that of finding the locations where replica servers must be placed. *Server placement* is related to the design phase of a CDN and the deployment of the networking infrastructure. So far, it has been addressed in [2, 4, 9, 10, 11, 12] by algorithms, which are centralized, encompass high complexity and are executed off-line. The second sub-problem, referred as *content replication,* is the selection of the subset of the available servers to store replicas of a specific object in a way that minimizes the replication cost [3, 5, 7]. This is an optimization problem that must be usually solved on-line, i.e., the decision algorithm must run after some event. Hence, heuristics and distributed algorithms are preferable.

Herein, we address only *content replication*, and we propose a generic approach to its formulation and solution. Our objective is to describe a framework for this formulation, which permits an efficient on-line solution. This solution, is heuristic and sub-optimal, but has the advantage that is scalable and provides local redundancy guarantees for the content. Our approach has similarities with several previous approaches in its various aspects. It is distributed but considers cooperation of neighboring nodes as [5, 7, 8] have also suggested. It can take into account various different metrics, such as storage space availability, server load, previous user accesses and subscriptions [3, 5, 7]. On the other hand, it does not consider metrics related to delivery performance (e.g., latency) as in [2, 5, 6, 7, 9, 11]. In our view, network conditions should be considered later during request routing. Also, it can be assumed that statistical data about workload and link properties have been taken into account during a prior network design phase. Section 2 presents our CDN model and our content replication policy, and its evaluation is attempted in section 3.

2 CDN Model and Content Replication Policy

Satellite networks are a particularly appealing solution for content distribution and cache pre-filling due to their inherent broadcasting capabilities [1]. In our model, a one-hop satellite network is used for the distribution of large files from the origin server to a large number of geographically dispersed surrogate servers via IP multicast. Content distribution is triggered either by the content provider according to some schedule (push model), or by client requests (pull model). A few seconds before the eventual content transmission, the origin server multicasts an announcement in a well-known multicast address. All surrogate servers receive the multicast announcement, and each one decides whether it should replicate the advertised content or not. By applying a cooperative decision scheme, where all the servers in the neighborhood use the same decision policy based on identical neighborhood information, all servers eventually take a common decision. A surrogate that decides to replicate the content joins the corresponding multicast addresses (made known via the announcement) and receives it. A reliable multicast protocol, whose details are beyond the scope of this paper, is used for the content transmission to the surrogates.

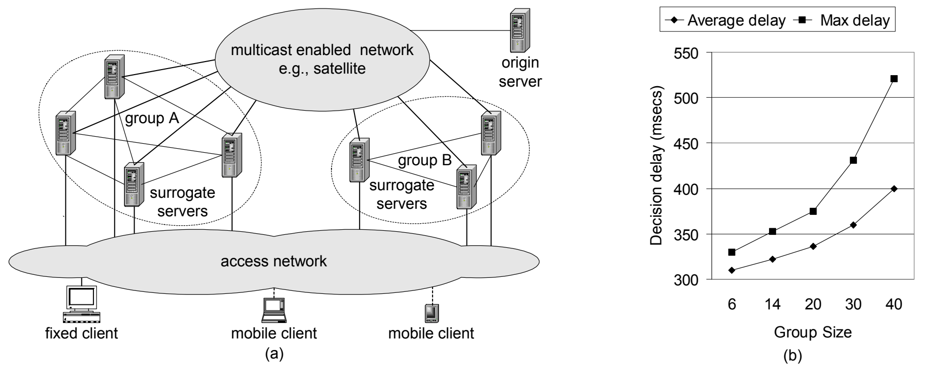

Fig. 1. (a) CDN model. (b) Simulation results for decision delay (*K=0.3*GroupSize*).

Fig. 1.a shows an abstract view of our CDN model that skips deployment specific details. A fully meshed grouping of the surrogate servers, based on geographical or network proximity criteria, has been determined during a prior network design phase. There are several reasons justifying this partitioning: (a) Replication decisions in a neighborhood must be independent from the decisions taken in other neighborhoods. (b) The server delivering to a client should not have long distance from this client. (c) The cooperation between servers in a group can ensure local content availability and redundancy. (d) The decision algorithm and the exchange of control messages inside a restricted neighborhood can be fast and incur only local traffic. (e) It maps to realistic CDN deployments where the nodes serving a geographical area (e.g., city, state, country) are relatively few and can be connected by fast links.

Ideally, if we replicated all objects in all the surrogate servers we would achieve the maximum possible content availability and the best delivery performance. There are, however, certain restrictions in doing so: (a) the overall storage space in a CDN is limited and, thus, we must restrict replication to the necessary, and (b) the distribution mechanism, although multicast-based, may present scalability issues due to the use of a reliable multicast protocol based on negative acknowledgements. A way to improve the CDN scalability is to multicast each object only in a subset of the receivers.

Apart from scalability, content replication aims at content availability in the sense of local replica redundancy and load balancing. In order to account for possible server failures and departures or conditions of overloaded servers, multiple replicas of an object should be available to support the demand in a given geographical area or in a given group of clients. To further enhance the content and system availability, the replication policy should avoid storing content in already loaded servers.

Most existing content placement approaches [2, 5, 6, 7, 9, 11] consider as known the locations of the clients and their distances from the candidate locations, and try to optimize the content delivery quality (e.g., minimize average latency). Using such a formulation they try to solve two problems in one step: the problem of finding the best server to replicate an object and the problem of finding the best server to deliver the object to clients. This formulation is valid for the off-line solution of the *server placement* problem where the distribution of clients and the network topology can be assumed to be static. In the case of *content replication*, however, such a formulation presents difficulties because: (a) its solution is computationally expensive and not scalable [6, 7], and (b) it ignores the network dynamics and the fact that content replication and content delivery do not usually occur at the same time. For these reasons, we de-correlate the content replication from the content delivery phase.

Each surrogate computes various local metrics that reflect different aspects of the server's current status and preferences of its local community. In our experimental system the metrics considered were related to the current *load* of the server, the available *storage space* and the *users interest* in the specific object. The parameters contributing to the estimation of the server *load* were the CPU usage (*CPUusage*), the memory usage (*RAMusage*), the aggregate bit-rate of all active multicast receptions (*InRate*), the aggregate bit-rate of all active client connections (*OutRate)* and the number of active client connections (*ActiveConns*). The *storage space* availability ($Storage_{avail}$) was reflected by the amount of currently free disk space that is reserved for object replicas. The *users interest* in an announced object was quantified by two terms: (a) the *Subscriptions* number that is the score found when we match the metadata of the object against the subscriptions of the local clients, and (b) the *PastUsage* that is the number of client hits to previous versions of the object.

When a surrogate server has computed the above values, it sends a message containing this value-set to all its neighbors. At the same time, all the neighbors perform exactly the same steps, and the result is that the value-sets describing the status of the servers are disseminated in the neighborhood. Each server finally collects an identical list of *N* value-sets (where *N* is the current size of the neighborhood), which then uses to evaluate a cost function and a number of constraints. A generic cost function for the replication of object *i* in server *n* could have the form:

$$\begin{gathered}
replicationCost_{ni} = a_1f_1(CPUusage_n)+a_2f_2(RAMusage_n) +a_3f_3(InRate_n) + \\
a_4f_4(OutRate_n)+a_5f_5(ActiveConns_n)+a_6f_6(Subscriptions_{ni})+{}_7f_7(PastUsage_{ni})
\end{gathered} \tag{1}$$

where $a_1,...,a_7$ are the weights of the various involved terms. An example constraint related to storage space is $Storage_{avail,n} > Storage_{thresh}$ (say $Storage_{thresh} = 100MB$). Of course, there is an infinite space of possible cost functions and constraints that could be alternatively applied. For instance, one could introduce a term related to storage space availability in the cost function above, e.g., $a_8f_8(Storage_{avail,n}, Storage_{max,n})$ or apply constraints related to server load such as (e.g., $InRate_n < 10Mbps$, $OutRate_n < 50Mbps$, $RAMusage_n < 80\%$). In any case, the focus of our work is not in the identification of a specific problem formulation, but in the design of a cooperative framework that permits the simple on-line solution of a whole family of formulations. Each surrogate server *m* computes the cost for all members in the group and checks if the identified constrains are satisfied. For the sub-list of group members that satisfy the constraints, the cost values are sorted and the members corresponding to the *K* lowest cost values are selected for local replication. If the server *m* is in the sub-list of the low cost servers it takes the decision to replicate the object. Note that exactly the same procedure is carried out in all servers, and since all servers of a group use identical input values, objective function and constraints, their decisions are identical.

There are different policies to define the number *K* of replicas taken in the group. A simple one is to set it proportional to the group size *N*, but not let it drop below a minimum value. This policy for *K* implies that all objects will have the same number of replicas in a group regardless from their popularity or their properties. Another option is to have *K* depending also on the *users interest* for the specific object. Also, we can have *K* depending on intrinsic object properties, such as its *importance* (e.g., the base layer of a scalable video is more important than the enhancement layers, and should have more replicas) and the *targeted audience* (e.g., when the content provider wants to increase or decrease the availability of specific objects in certain regions).

3 Evaluation

Scalability: An on-line content replication algorithm must have low complexity and should generate the least possible network traffic. As [6] suggests, existing centralized content placement algorithms [2, 5, 9, 10, 11, 12] are not scalable because the *computational complexity* increases with network size *N* (in the best case *O(N)*). In the proposed approach, the decision algorithm running in a specific node is independent from the state and the decisions taken at nodes belonging to other groups. Assuming a bounded group size (say $M \leq 20$), the computational complexity at any processing node is also bounded *O(M).* Of course any distributed algorithm with restricted scope (neighborhood size) shares the same advantage. At the limiting end, purely local algorithms [3, 5, 7] have very low complexity *O(1)* but the decision quality is lower. *Messages transfers* influence the scalability in two ways: They are additional network traffic and they add extra delay in the decision process. In a centralized algorithm each replication decision requires *2N* message transfers. The problem with these messages is that many of them travel long distances through several network hops and through links that may be slow. Thus, the effective network traffic is higher and the added delay due to these messages is large and, in the general case, increases as the size of the CDN grows. In our algorithm, each surrogate sends *M-1* messages to its neighbors and receives *M-1* messages from them. Hence, the number of messages required to take replication decisions for an object in the *M* nodes of a group is *M(M-1)*, but these messages do not induce end-to-end traffic. Also, the average added delay due to message transfers can be always kept below some threshold if the network distance between group members is bounded. In general, the delay due to message transfers of distributed algorithms decreases when the scope of the algorithm becomes narrower. In fig. 1.b we have plotted the simulation results for the replication decision delay (mean values of average and maximum decision delay) for different group sizes in a 100Mbps LAN environment. These results show the applicability of our algorithm for on-line replication decisions, since for relatively small groups the replication decision delay is acceptable (<1 sec).

Local content redundancy: A drawback of existing centralized and purely local replication algorithms is that they do not provide any guarantees for local redundancy of replicas since they do not set any constraints related either to the relative placement of replicas or to the number of replicas taken in a specific region. Our approach inherently guarantees local redundancy as each group of surrogates takes exactly *K* replicas, where *K* may be derived in different ways (see previous section).

Flexibility of problem formulation and solution: The decision quality of most centralized algorithms is related to the client perceived latency. However, the solution ignores the actual network conditions, which may change dynamically and deviate a lot from the initial assumptions, and does not account for mobility of clients. Also, various assumptions are made in order to simplify the solution of the optimization problem, e.g., the constraints are relaxed and incorporated in the cost function [7,11], or large and small objects are not differentiated [7]. It is not always easy to incorporate additional metrics in their cost or constraint functions without serious impact in their complexity. On the other hand, our approach makes no simplification and the constraints are always satisfied by the solution. It is flexible and can easily incorporate any type of local metrics and constraints according to the model at hand.

Easiness of deployment: Centralized approaches assume that the central node knows the network topology and the latency to any node or client. Also, keeping this central node synchronized with the contents of all the nodes is a very difficult task. Achieving quick decisions and synchronization in such a system requires that the links with all the nodes are fast and reliable. The above requirements cannot be easily satisfied in real deployments with many nodes placed at distant locations. In our approach, configuration and synchronization involves only the neighborhood and is much simpler. Also, redundancy of processing nodes is not an issue. If any node fails the algorithm runs without problem in other nodes. The quality of the network links is important only between the nodes of the same group. And it is reasonable to assume that in a real deployment these nodes are placed topologically close to each other.

4 Conclusions and Future Work

We have proposed a scalable and flexible cooperative approach for content replication that solves several problems encountered by centralized and purely local algorithms. The grouping of surrogates provides a trade-off between scalability and local replica redundancy. Our approach lies somewhere in the middle between: (a) centralized algorithms, which take a near-optimal number of replicas but are not scalable and do not care about their relative placement, and (b) purely local algorithms, which are simple and scalable but the number of replicas taken may be far from the optimal while they make no provisions at all for their relative placement.

An important topic for future work is the storage management of surrogate servers, and particularly the use of cooperative replica removal and regeneration policies.

References

1. A. Armon and H. Levy. Cache satellite distribution systems: modeling and analysis. In Proceedings of IEEE INFOCOM, 2003.
2. Y. Chen, R. H. Katz, and J. D. Kubiatowicz. Dynamic replica placement for scalable content delivery. In Proceedings of 1st Int. Workshop on Peer-to-Peer Systems, 2002.
3. M. Chen, J. P. Singh and A. LaPaugh. Subscription-enhanced content delivery. In Proceedings of WCW'03, 2003.
4. C. Huang and T. Abdelzaher. Towards content distribution networks with latency guarantees. In Proceedings of 12th IWQoS, 2004.
5. J. Kangasharju, J. Roberts, and K. W. Ross. Object replication strategies in content distribution networks. In Proceedings of WCW'01, June 2001.
6. M. Karlsson, C. Karamanolis and M. Mahalingam. A framework for evaluating replica placement algorithms. Technical Report HPL-2002-219, HP Labs, 2003.
7. M. Karlsson and C. Karamanolis. Choosing replica placement heuristics for wide-area systems. In Proceedings of IEEE ICDCS, 2004.
8. M. Korupolu, G. Plaxton, and R. Rajaraman. Placement algorithms for hierarchical cooperative caching. Journal of Algorithms, January 2001.
9. S. Jamin, C. Jin, A. R. Kurc, D. Raz, and Y. Shavitt. Constrained mirror placement on the Internet. In Proceedings of IEEE INFOCOM, 2001.
10. B. Li, M. J. Golin, G. F. Italiano, X. Deng, and K. Sohraby. On the optimal placement of web proxies in the Internet. In Proceedings of IEEE INFOCOM, 1999.
11. L. Qiu, V. N. Padmanabhan and G. M. Voelker. On the placement of web server replicas. In Proceedings of IEEE INFOCOM, 2001.
12. P. Radoslavov, R. Govindan, and D. Estrin. Topology-informed internet replica placement. In Proceedings of WCW'01, June 2001.
13. H. Yu and A. Vahdat. Minimal replication cost for availability. In Proceedings of the 21st annual Symposium on Principles of Distributed Computing, 2002.