A Novel Approach to Managing Consistency in Content Distribution Networks

Zongming Fei
Laboratory for Advanced Networking
Department of Computer Science, University of Kentucky
773 Anderson Hall, Lexington, KY 40506
fei@cs.uky.edu

Abstract

Content distribution network (CDN) is a technique deployed to push content from the origin server to geographically distributed replicas, usually located at the edge of the network where the clients are attached. One of the important problems in CDNs is how to manage the consistency of content at replicas with that at the origin server, especially for those documents changing dynamically. In the traditional propagation approach the updated version of a document is delivered to all replicas whenever a change is made to the document at the origin server. It may generate significant levels of unnecessary traffic if documents are updated more frequently than accessed. Another approach is invalidation, in which an invalidation message is sent to all replicas when a document is changed at the origin server. This approach does not make full use of the distribution network for content delivery and each replica needs to fetch an updated version individually at a later time. This can also lead to inefficiency in managing consistency at replicas. In this paper, we propose a novel hybrid approach that will generate less traffic than the propagation approach, *and* the invalidation approach. The origin server makes the decision of using either propagation or invalidation method for each document, based on the statistics about the update frequency at the origin server and the request rates collected by replicas. We develop a technique that can reduce the burden of request rate collection at replicas and get rid of the implosion problem when replicas send the statistics to the origin server. Extensive simulations are performed to examine how the traffic generated and freshness rate at replicas are affected by various parameters. We experiment with a wide range of request rate, update frequency, and the number of replicas. The results show that our approach can take advantage of content distribution network and significantly reduce the traffic generated.

Keywords: Content distribution network, consistency, caching, propogation, invalidation

1 Introduction

Over the past few years we have witnessed an unprecedented growth in the number of Internet users. Perhaps even more impressive is the rate at which new "content" is being added to the Web each day. Moreover, there doesn't appear to be any end in sight to the addition of new web content. Making information available to a rapidly growing user population is quickly becoming a challenging problem, which has been attacked from both the client side and the content provider side.

- On the client side, the *caching* technique has been employed to store documents accessed by clients and serve the future accesses to the same documents without having to retrieve the document from the origin server. A web cache can be maintained at each client or can be shared by a group of clients via a caching proxy. It is also possible to organize caches hierarchically.

- On the content provider side, the *replication* technique has been used to equip a farm of servers and the load of client requests is distributed. More recently, the *Content Distribution Network* (CDN) is used to push the content from the origin server to geographically distributed replicas, which bring content to the edge of the network where the clients are attached.

Our focus is on the CDNs and how content at the origin server is delivered to replicas. The two common approaches to this problem are to deliver the data over (1) N unicast channels or (2) over an application-level (tunneled) multicast tree that connects the replicas [1, 2]. Clearly the unicast approach wastes network bandwidth and can cause congestion at bottleneck links, while application-level multicast approaches are more efficient in delivery (although not as efficient as native IP multicast). Though currently more deployed CDNs use unicast over multicast, there are increased interests in designing (application layer) multicast for the content delivery. We believe the multicast approach will be finally adopted because of its efficiency. In this paper, our discussion will be based on the assumption that there is some multicast delivery network established between the origin server and replicas, either native multicast or application layer multicast.

The efficiency of delivery schemes depends on the characteristics of documents. Roughly speaking, documents requested by clients can be divided into three categories.

- Static documents. Such documents usually do not change at the origin server, or if they do change, do so at very large time scales.

- Dynamic documents. These are documents that change at a relatively high rate. They can generate large amounts of update traffic if not kept in check.

- Dynamically generated documents. These documents are generated on-the-fly by the server, or replica, based on the information provided by clients.

For dynamically generated documents, the server typically needs to run some "cgi" code to produce the document and thus cannot be implemented with the caching approach. However, in CDNs, the replica, like the origin server can be instantiated with the appropriate code to generate these documents dynamically. The information used to generate the documents may change frequently or infrequently like other standard web documents and thus this information is dealt with just like dynamic or static web content.

The update of static documents is not a problem because it is, in most cases, one time effort and thus does not generate too much traffic. This is not the case for updating dynamic document. It involves generating much higher levels of traffic. It is the goal of this paper to design an efficient scheme for updating dynamic documents and maintaining consistency of documents at replicas with the origin server. There are two ways to update dynamic documents:

- Propagation. Whenever an object is changed at the origin server, it is propagated to all replicas.

- Invalidation. Whenever an object is changed, an invalidation message is sent to all replicas. Each replica will fetch the new version of the object individually at a later time, if necessary.

Unfortunately, the propagation approach often generates significant levels of unnecessary traffic if documents are updated more frequently than they are requested. On the other hand, the invalidation approach may fail to make full use of the delivery network and can degrade to unicast delivery from the origin server to each individual replica. More importantly, the response time will also suffer because the object has to be fetched from the origin server after the request arrives at the replica. This is not true of the propagation approach, where an object is always fresh at replicas.

In this paper, we propose a novel hybrid approach that will generate less traffic than the propagation approach, *and* the invalidation approach. The origin server makes the decision of using either propagation or invalidation method for each document, based on the statistics about the update frequency at the origin server and the request rates collected by replicas. We develop a technique that can reduce the burden of request rate collection at replicas and get rid of the implosion problem when replicas send the statistics to the origin server. Extensive simulations are performed to examine how the traffic generated and the freshness rate at replicas are affected by various parameters. We experiment with a wide range of request rate, update frequency, and the number of replicas. The results show that our approach can take advantage of content distribution network and significantly reduce the traffic generated, compared with the propagation approach and the invalidation approach.

The remainder of this paper is organized as follows. Section 2 provides an overview of our architectural model and assumptions. Section 3 describes schemes for managing consistency of documents at replicas with the origin server. In Section 4 we present extensive simulations to evaluate the performance of various delivery schemes. Related work is discussed in Section 5 and we conclude the paper in Section 6.

2 Architectural Overview

In this section we briefly outline the architectural model and assumptions. The Internet can be described as a set of interconnected, autonomous domains (see Figure 1). A domain may have zero or more content providers (called *origin servers*), which can be fully or partially replicated in other domains. An origin server can specify whether it is replicable or not, and similarly, a domain may have policies about whether it allows replicas in the domain or not.

In the example in Figure 1, domain D_1 contains an origin server S, which has seven replicas (R_2, R_3, R_4, R_5, R_8, R_9, R_{11}) located in different domains. We assume that a delivery infrastructure (or *delivery network*) has been established from the origin server to the replicas. For example, an application layer multicast [1] protocol may have set up unicast tunnels from S to R_2, from R_2 to R_3, R_4, R_5, from R_4 to R_8, R_9, and from R_5 to R_{11}. This delivery network can be used to propagate information from the origin server S to all replicas. Information collected by replicas can be combined along the way when they are sent back to the ori-

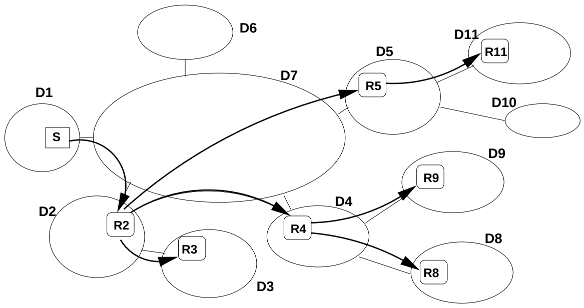

Figure 1: Architecture for Content Delivery

gin server through this delivery network.

We view the establishment of replicas as an incremental process. When a new replica is set up, we need to decide how to fill up its content (all or partial). Later when a change is made to a document at the origin server, we need to decide how the update will be made aware to all replicas. Our goal is to guarantee that any object served by any replica to clients is always fresh.

We assume all client requests will be directed to some replica. Which specific replica will be chosen for a request by a client is the server selection problem. Different schemes for server selection may result in different response time of clients and the traffic generated from clients to replicas. Because server selection is not the focus of this paper, we ignore the traffic generated from the clients to replicas. The traffic we are interested in and used as one of performance measures is the traffic generated for delivering content from the origin server to replicas. In the following sections, we will describe several consistency management schemes, which will generate different levels of traffic.

3 Schemes for Managing Consistency

3.1 Propagation

Propagation is an approach often used in CDNs. Whenever an object is changed at the origin server, it is propagated to all replicas through the delivery network by multicast. This can guarantee that an object is always up to date at any replica when it is requested from a client. We assume that the propagation will not take too long, so we ignore the temporal state in which a request for a document is processed at a replica when the document has been changed at the origin server and is being propagated towards replicas. In these cases a client will get a copy of document different from the one at the origin server, but we still consider it fresh. Under this assumption, we know that the documents served by replicas are always the newest version.

For a newly established replica, all documents at the origin server will be delivered to it by unicast. This will guarantee that each replica has an up-to-date version of all objects at the origin server when it is set up.

3.2 Invalidation

Another approach is invalidation, in which an invalidation message is delivered to all replicas through the delivery network by multicast when an object is changed at the origin server. Each replica will mark the object invalid upon the reception of this invalidation message. When a request for the object is processed at the replica at a later time, the replica will send a request to the origin server, which will use unicast to deliver the object to this replica. We also ignore the temporal state in which an invalidation is being delivered from the origin server to replicas and consider the copy served by replica during the time fresh. Though a document in a replica may be invalid for a certain period of time, the copy served by the replica to clients will always be the newest version.

In the invalidation approach, a newly established replica can simply assume that all objects are invalid. The replica will be filled up as it gets more requests from clients and tries to fetch a fresh copy from the origin server.

3.3 Our Approach

We propose a hybrid approach based on the observations of problems existing in the propagation and invalidation approaches. With propagation, a document is propagated to replicas even if it is not accessed at all. This may generate extra traffic and we find that if a document is not accessed frequently enough, it should not be propagated.

In the invalidation approach, a document may be delivered to replicas through unicast because it is invalidated by the origin server and each replica fetch the document individually. Thus the traffic is more than it could have been if the multicast delivery is used. The observation is that if a document is accessed frequently enough, we should use multicast to deliver it to replicas, rather than let each replica fetch it by unicast.

In addition to the access frequency (or *request rate*), another factor is the *update rate* of documents at the origin server. It determines how frequently a document is propagated or invalidated and thus, the traffic for propagation or unicast fetching. Actually it is the relative values of the request rate and the update rate that determine what method we should use. It is simple in the caching scheme because we can use propagation if the request rate is greater than the update rate, otherwise use invalidation. We cannot generalize this to CDNs by simply comparing the rate of requests and updates, because we have to take into account that the methods of transmitting documents from the origin server to replicas are different, propagation using multicast and invalidation using unicast. Therefore, we need to factor in the relative efficiency of unicast and multicast.

Relations between unicast and multicast transmissions were first studied in [3] and recently validated by empirical studies [4, 5]. Assume that L_m is the total number of multicast links in the distribution tree and L_n is the total number of unicast hops for distribution from the source to all replicas. Then we have

$$\frac{L_m}{L_n} = 1 - \delta \tag{1}$$

where δ represents the percentage gain in multicast efficiency over unicast. We can have an estimate of δ in terms of the number of receivers, N and an associated *efficiency factor*, ε. That is, $\delta = 1 - N^{\varepsilon}$. The value of ε tends to range between -0.34 and -0.30 [5]. In our simulations we set $\varepsilon = -0.3$.

We consider a specific object, assuming its update rate at the origin server is U and the total request rate is R. As discussed above, we focus on the traffic generated for transmission of the document from the origin server to replicas. Assume that the document size is s and time we measure the traffic is t (Actually s and t will later be removed from the inequality because they appear at both sides.). If we use propagation, the traffic generated is $T_p = U * L_m * s * t$.

In the invalidation approach, we ignore the traffic generated for sending invalidation messages for now (because they are relatively small) and focus on unicast fetch by replicas. The request rate by each replica is $\frac{R}{N}$ and the average number of hops from the origin server to each replica is $\frac{L_n}{N}$. Therefore, the traffic generated by invalidation [1] is $T_i = (\frac{R}{N} * \frac{L_n}{N}) * s * t * N = R * L_n * s * t/N$.

Our strategy is based on the relative value of T_p and T_i. Use propagation if $T_p < T_i$; use invalidation if $T_p > T_i$. With the power law relation (1) between L_m and L_n, we can get our final decision rule:

For each object, if $U * N^{1+\varepsilon} < R$, use propagation;
otherwise use invalidation.

Where U is the update rate, R is the total request rate, N is the number of replicas, and ε is the efficiency factor ($-0.34 \leq \varepsilon \leq 0.30$).

The number of replicas is available at the origin server. The update rate for each object can be calculated based on the history. The main problem is to get the total request rate at the origin server, while all requests are made to replicas. The simple solution is that each replica collects statistics about the request rate and periodically pushes the data to the origin server. The implosion problem may occur if the number of replicas is large. Our solution is to make use of the delivery network existing between replicas and the origin server. Rather than let replicas push the statistics directly to the origin server, we let the origin server periodically send poll messages to those replicas directly connected in the multicast distribution tree. When a replica receives a poll message, it sends a poll message to replicas directly connected to it downstream, if any. If a replica does not have any replica connected to it downstream, it will send a reply with request rate information to its upstream replica (or the origin server). Those replicas having sent out poll messages will calculate the aggregate request rate for each object after it receives all replies to its poll messages. Then these aggregate rates are sent further upstream. The origin server will get the total request rate information after it receives replies from all replicas directly connected to it. This request rate collection process will take place in the time frame of hours. So the traffic generated is not significant compared to the volume of traffic generated for shipping objects from the origin server to replicas. The collection process is similar to the concast in the context of active networking [6, 7], but we can deploy the algorithm in our distribution network without requiring the network being active.

One way to reduce the size of request rate information is that we can ignore those objects requested less than or equal to a threshold value. For example, if we set the threshold value to 1, we can ignore objects requested once or not requested at all. Actually the number of these objects may be large and the size of the reply messages can be made much smaller. It is up to the origin server whether to make adjustment to the final result, if such optimization takes place at replicas. For example, if the threshold value is 1, we can let the origin server add N/2 to the request rate of each object.

One remaining problem is how to deal with the newly established replica. Our solution is based on the statistics currently available at the origin server. Those documents satisfying $U * N^{1+\varepsilon} < R$ will be delivered to the replica by unicast, and all other documents will be marked invalid at the replica.

4 Performance Evaluation

4.1 Simulation Setup

We use GT-ITM [8] to generate 1600 node transit-stub graphs. The origin server is randomly located in one of stub domain. We experiment with 10, 50, 90 replicas, which are located in different stub domains at the node directly connected to transit domain. These replicas and the origin server are connected by a multicast delivery network, which is established by first calculating a complete distance graph

[1] Note that traffic generated by the invalidation approach is usually less than T_i because it is the worst scenario value.

for replicas and the origin server and then generating a spanning tree. This tree is used for delivering propagation and invalidation messages. We assume that the average size of objects is 10 KBytes and the size of an invalidation for each object is 100 Bytes.

Assume that there are ten thousand objects at the origin server and the total request rate to these objects is either 0.1, 1 or 10 million times per day. We assume the distribution of requests to different objects follows the Zipf distribution [9, 10]. If we order the objects from 1 to 10,000 according to their popularities, the probability of requesting objects $1, 2, 3 \cdots$ will be proportional to $\frac{1}{1}, \frac{1}{2}, \frac{1}{3}, \cdots$. Each request will be made to one replica selected randomly.

Next problem is to determine the *inter-update time* (the inverse of the update rate) of each object. Previous research has shown that there is no direct relationship between the request rate to an object and its inter-update time [10]. We use a random allocation scheme to set the inter-update time for each object. We first choose an average inter-update time of all objects, τ. The inter-update time of an object is randomly chosen from $(0, 2 * \tau)$. We experiment with τ ranging from 1 to 200 hours. Note this is average inter-update time. The real inter-update time of an object can be very close to 0 and thus the update frequency can be very high.

We have two performance measures. The first one is *traffic generated* during one day. If an object of size s goes through n hops, the traffic generated is counted as $s * n$. This is used in both multicast and unicast delivery. Another interesting measure is the *freshness rate*, which is the percentage of objects that are fresh when a request arrives at the replica. We want to emphasize that if a replica finds an object is not fresh, it will fetch a fresh copy from the origin server and deliver this new copy to the client. Therefore, all objects served by replicas are always consistent with that in the origin server, even if the freshness rate is less than 100%. The freshness rate only measures the probability that replicas can serve directly without resorting to the origin server. Therefore, the higher the freshness rate, the shorter the response time.

4.2 Results

We compare three approaches: (1) the propagation approach, (2) the invalidation approach, and (3) our hybrid approach. We first experiment with the request rate of one million per day and let the average inter-update time change from 1 to 100 hours. We always use 50 replicas in the following simulations, unless we explicitly specify otherwise. The traffic generated by each approach is shown in Figure 2. When the average inter-update time is less than 10, the propagation approach generates extremely high volume traffic

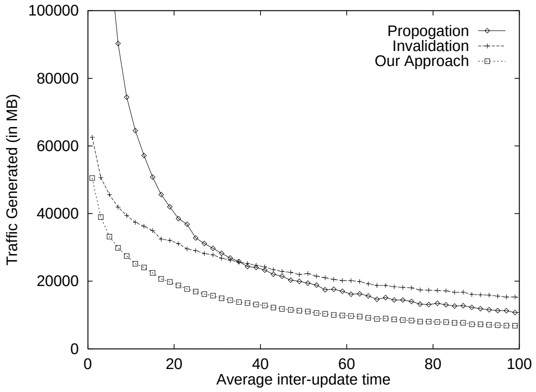

Figure 2: Traffic generated (request rate = 1M)

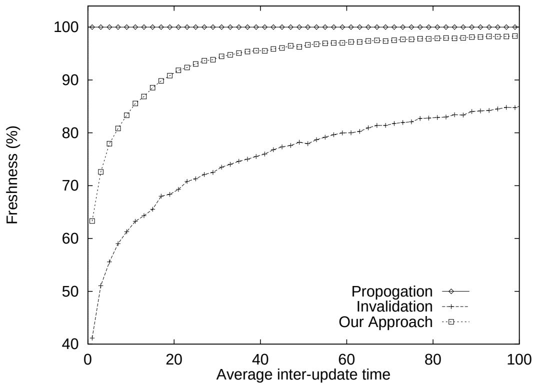

Figure 3: Freshness rate (request rate = 1M)

because of high frequency of blind propagation. When the inter-update time increases from 1 to 100, the traffic generated by all three approachs decreases. The propagation approach decreases faster than the invalidation approach, because it takes advantage of multicast delivery network while the invalidation approach relies on unicast fetching by replicas from the origin server. When the average inter-update time is larger than 35, the propagation approach generates less traffic than the invalidation approach. Our approach chooses different methods for different kinds of objects and can always generate less traffic than both the propagation approach and the invalidation approach. Figure 3 shows the freshness rate at replicas. The propagation approach always has 100% freshness rate. Our approach has a significantly higher freshness rate than the invalidation approach. The freshness rate is greater than 90% when the average inter-update time reaches 19, at which point the propagation approach generates more than twice as much traffic as ours and the invalidation approach has a freshness rate less than 70%.

Figure 4 and 5 give the traffic and the freshness rate when the average inter-update time changes from 100 to 200. We

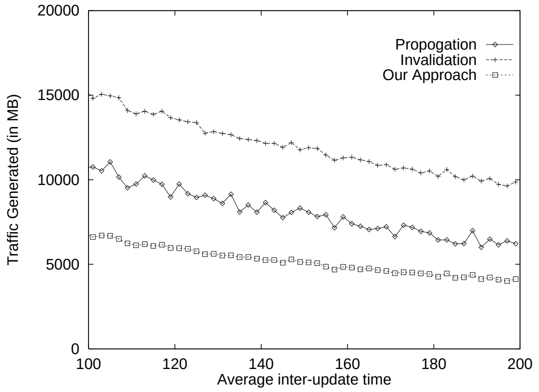

Figure 4: Traffic generated (request rate = 1M, average inter-update time > 100)

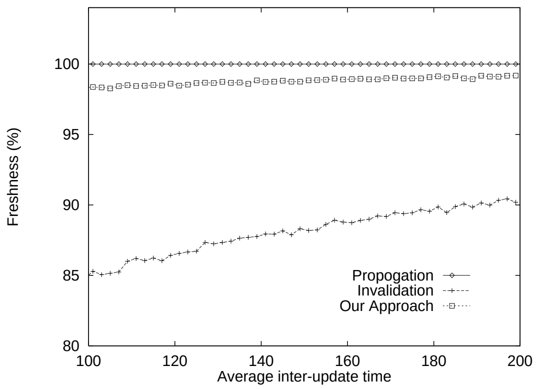

Figure 5: Freshness rate (request rate = 1M, average inter-update time > 100)

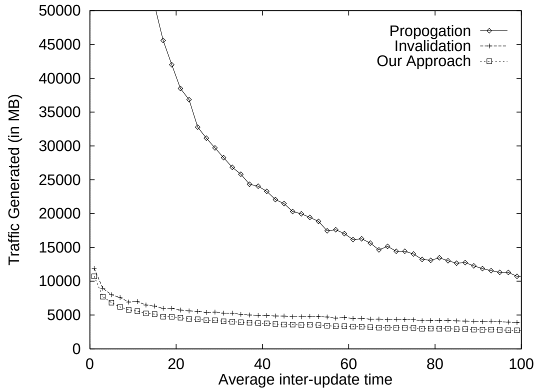

Figure 6: Traffic generated (request rate = 0.1M)

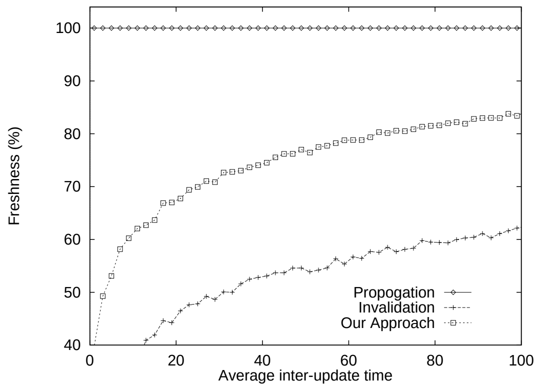

Figure 7: Freshness rate (request rate = 0.1M)

get a closer look and can find that the traffic generated by our approach is still significantly less than that generated by both the propagation approach and the invalidation approach, while the freshness rate of our approach is much closer to 100% and the freshness rate of the invalidation approach is almost always less than 90%.

Figure 6 and 7 give the traffic and the freshness rate when the request rate is 0.1 million per day. The traffic generated by the invalidation approach and our approach is much less than in the previous 1M request rate case. They also come closer, and the difference between them and the propagation approach becomes larger. The freshness rate of our approach is always 20% higher than the invalidation approach.

We increase the request rate to 10 millions per day in Figure 8 and 9. The traffic generated by the three approaches is all higher than in the 1M request rate case. This time the traffic generated by the invalidation approach is significantly higher than both the propagation approach and our approach. This tells us that in these relatively high request rate case, propagation is more appropriate. At the same time, the freshness rate of our approach is very close to 100% and much better than that of the invalidation approach.

Next we examine the effect of the number of replicas. We set the request rate to one million. In Figure 10 and 11, we have 10 replicas. All three approaches generate less traffic than the previous 50 replica case in Figure 2 and 3. The cross point of the propagation approach and the invalidation approach shifts left. It changes from 35 in the 50 replica case to 10. Also the freshness rate is higher than in the 50 replica case. Figure 12 and 13 show the traffic and the freshness rate when the number of replicas is 90. As expected, all three approaches generate more traffic than the 50 replica case and the cross point increases to 53. The freshness rate is lower than in the 50 replica case.

5 Related Work

Efficient delivery of content to clients has attracted much attention over a decade. The interests in it are also exemplified by the commercial development of delivery net-

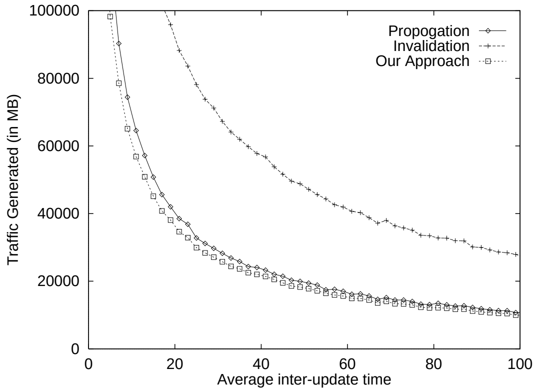

Figure 8: Traffic generated (request rate = 10M)

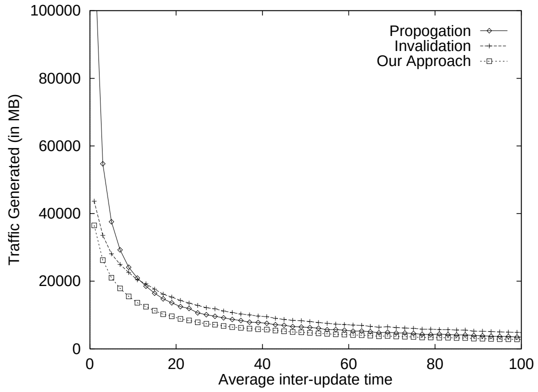

Figure 10: Traffic generated (10 replicas)

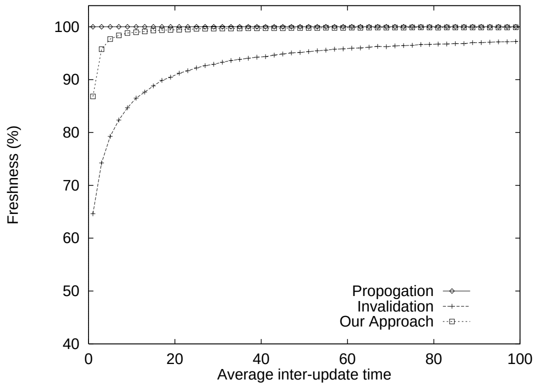

Figure 9: Freshness rate (request rate = 10M)

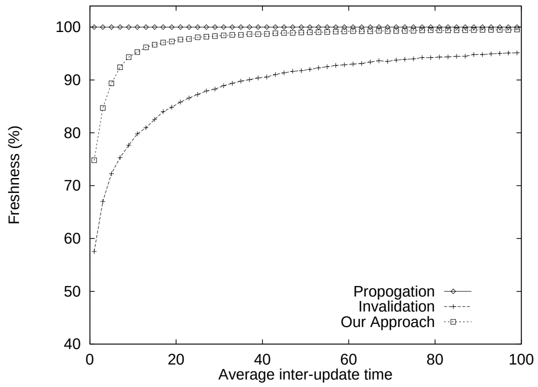

Figure 11: Freshness rate (10 replicas)

work [11, 12, 13]. The design and techniques used in these systems are mostly proprietary.

Early research on content delivery concentrates on using native multicast to push the content to the clients directly [14, 15]. These schemes are most efficient when the content is changed at a high frequency [16]. It is not an easy task to determine whether a document should be multicast or not. The lack of native multicast support leads people to design alternative ways for delivery. Prominent among them are application layer multicast schemes: Yallcast [2], end system multicast [1], and scattercast [17]. Basically they run an auto-configuration protocol to establish a delivery structure of tunneled topology among participating members. These unicast connections are used for delivery of content from the origin server to participating nodes. While the Yallcast set up tunnels directly among members, the end system multicast and the scattercast build a mesh topology first and then run the spanning tree algorithm to establish a delivery tree. While the self-organization topology setup mechanism complements our work, these schemes do not discuss how to update dynamic content efficiently.

Caching is an area closely related to content delivery. Harvest [18] is an earliest proposal trying to establish a hierarchical caching structure, in which the bandwidth can be more efficiently used and the popular web pages can be pushed closer to the clients. Other hierarchical schemes are adaptive web caching [19], access driven cache [20]. The problem with caching schemes is how to maintain the freshness of the content stored in the cache. There are several proposals dealing with this problem. A simple way to guarantee the freshness is that the cache always check with the origin server. Using IMS, a cache can get the reply from the origin server that the page is fresh, or get a fresh page itself. There are some heuristic schemes to determine the freshness by observing the life time of a page [21]. The longer a page has not been modified, the longer it will not be modified in the future. However, these adaptive schemes cannot guarantee the freshness of a page. The invalidation scheme was proposed to deal with the problem [22]. Unfortunately, requiring a server to contact a huge number of caches is simply not feasible from a scalability standpoint. In CDNs, the invalidation can be used because knowledge about replicas is available and the delivery network can be used to forward the invalidations.

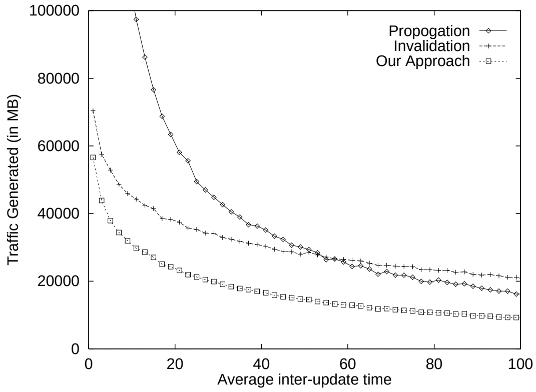

Figure 12: Traffic generated (90 replicas)

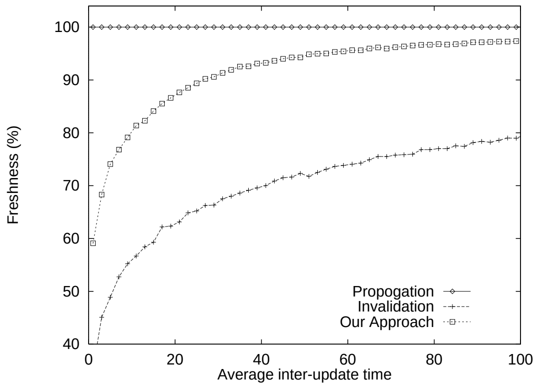

Figure 13: Freshness rate (90 replicas)

Invalidation can be used most efficiently when combined with propagation and other approaches in CDNs. SPREAD architecture [23] analyzed the tradeoff among client validation (by using IMS), invalidation, and propagation (called replication in that paper). It analyzed the effects of different update distributions on the determination of the thresholds that decide which method to use. The discussion was based on the translucent proxying architecture that achieves transparency to clients by intercepting SYN of a TCP connection and using IP tunneling. The paper used a trace-driven simulation to show the benefits of the threshold-based approach with unicast validation, invalidation and propagation in terms of consumed bandwidth and client latency. There is a brief discussion about multicast extension to the architecture. MMO protocol [24] proposed to use multicast for delivery and invalidation of updates. Whenever an object is changed at the origin server, an invalidation is sent to all replicas by multicast. The new version is immediately delivered to a multicast group, through which all subscribed replicas can get the updated version of an object. A decision is made at each replica whether to join the multicast group or not, based on a threshold W. If there is no request during previous W invalidation periods, the replica does not join; otherwise, it does. The paper shows the performance gain of the protocol without details about how to determine the threshold and how it is related to the inter-update time of an object.

The focus of this paper is on the impact of multicast delivery on the decision making at the origin server whether to use propagation and invalidation. We design a very simple decision rule to select the best strategy on the per object basis. Equipped with our efficient scheme for access rate collection, our approach makes propagation and invalidation perform in a way to generate maximal benefit from each of them, and achieves the goal of reducing the traffic for consistency management.

6 Concluding Remarks

Content distribution network can be deployed to improve client performance by pushing the content towards the edges of the network and closer to clients. One of the key problems in CDN is how to manage the consistency of content at replicas with the origin server, especially for those documents changing dynamically. This problem will become more prominent when more web sites use CDNs. In this paper, we proposed a novel approach in which the origin server determines the best way for each document. We explored the impact of multicast delivery network on our decision making process. By using the power law relation between multicast and unicast distributions, we can more accurately determine the traffic generated and make a right decision. The process of collecting request rates is made scalable to a very large number of replicas by aggregating information at the intermediate replicas. We performed extensive simulations to explore the effects of the request rate, the average inter-update time and the number of replicas. The simulations demonstrate that our approach can generate significantly less traffic than both the propagation approach and the invalidation approach.

One question we can further investigate is what the results will be affected if the demand for an object is not uniform across all replicas. One way to deal with it is that we can form multiple multicast groups with each group for a set of objects closely related. Each replica will be associated with those groups having objects that the replica has a high request rate. The decision will be made at the origin server for each group based on the number of replicas in the group and information collected from those replicas. We will study issues of how to divide objects into groups and how replicas will be associated with them in the future.

Acknowledgement

The author would like to thank Dr. Jim Griffieon for his discussions of the subject and anonymous reviewers for their comments on the paper.

References

[1] Y. Chu, S. Rao, and H. Zhang, "A case for end system multicast," in *Proceedings of ACM Sigmetrics*, June 2000. Santa Clara, CA.

[2] P. Francis, "Yallcast: Extending the Internet multicast architecture." http://www.yallcast.com.

[3] J. Chuang and M. Sirbu, "Pricing multicast communication: A cost based approach," in *Proceedings of INET'98*, July 1998. Geneva, Switzerland.

[4] G. Phillips, S. Shenker, and H. Tangmunarunkit, "Scaling of multicast trees: Comments on the Chuang-Sirbu scaling law," in *Proceedings of ACM SIGCOMM'99*, August 1999. Cambridge, Massachusetts.

[5] R. C. Chalmers and K. C. Almeroth, "Modeling the branching characteristics and efficiency gains in global multicast trees," in *Proceedings of INFOCOM'2001*, April 2001.

[6] K. Calvert, J. Griffioen, A. Sehgal, and S. Wen, "Concast: Design and implementation of a new network service," in *Proceedings of 1999 International Conference on Network Protocols, Toronto, Ontario*, November 1999.

[7] K. Calvert, J. Griffioen, A. Sehgal, and S. Wen, "Implementing a concast service," in *Proceedings of the 37th Annual Allerton Conference on Communication, Control, and Computing*, September 1999.

[8] K. Calvert, M. Doar, and E. W. Zegura, "Modeling Internet topology," *IEEE Communications Magazine*, June 1997.

[9] G. Zipf, ed., *Human Behavior and the Principle of Least Effort*. Reading, MA: Addison-Wesley, 1949.

[10] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: Evidence, and implications," in *Proceedings of INFOCOM'99*, March 1999.

[11] Akamai, http://www.akamai.com.

[12] Digital Island, http://www.digitalisland.net.

[13] Inktomi Corporation. http://www.inktomi.com.

[14] J. Nonnenmacher and E. Biersack, "Asynchronous multicast push: AMP," in *Proceedings of ICCC'97*, pp. 419–430, Nov. 1997. Cannes, France.

[15] Rodriguez and E. Biersack, "Continuous multicast push of web documents over the Internet," *IEEE Network Magazine*, vol. 12, pp. 18–31, Mar-Apr 1998.

[16] P. Rodriguez, E. W. Biersack, and K. W. Ross, "Improving the www: Caching or multicast?," in *Proceedings of the 3rd International Web Caching Workshop*, June 1998. Manchester, England.

[17] Y. Chawathe, S. McCanne, and E. A. Brewer, "An architecture for Internet content distribution as an infrastructure service," Feb. 2000. Available at http://www.cs.berkeley.edu/ yatin/papers/scattercast.ps.

[18] A. Chankhunthod, P. B. Danzig, C. Neerdaels, M. F. Schwartz, and K. J. Worrel, "A hierarchical Internet object cache," in *Usenix'96*, January 1996.

[19] S. Michel, K. Nguyen, A. Rosenstein, L. Zhang, S. Floyd, and V. Jacobson, "Adaptive web caching: Towards a new caching architecture," *Computer Network and ISDN Systems*, Nov. 1998.

[20] J. Yang, W. Wang, R. Muntz, and J. Wang, "Access driven web caching." UCLA Technical Report 990007.

[21] J. Gwertzman and M. Seltzer, "World-wide web cache consistency," in *Proceedings of the USENIX Conference*, Dec. 1996. Copper Mountain Resort, CO.

[22] H. Yu, L. Breslau, and S. Shenker, "A scalable web cache consistency architecture," in *Proceedings of ACM Sigcomm'99*, August 1999.

[23] P. Rodriguez and S. Sibal, "SPREAD: Scalable platform for reliable and efficient automated distribution," in *Proceedings of the 9th International World Wide Web Conference*, May 2000. Amsterdam.

[24] D. Li and D. R. Cheriton, "Scalable web caching of frequently updated objects using reliable multicast," in *Proceedings of 2nd USENIX Symposium on Internet Technologies and Systems (USITS'99)*, October 1999. Boulder, Colorado.