Caching and Multicasting in DBS Systems

Mingyan Liu Manish Karir John S. Baras
Center for Satellite and Hybrid Communication Networks
Department of Electrical and Computer Engineering & Institute for Systems Engineering
University of Maryland, College Park, MD 20742, USA
{daphnel, karir, baras}@isr.umd.edu

Abstract

The use of Caching and Multicasting has been studied extensively in the context of terrestrial networks. However, the use of these technologies in a Direct Broadcast Satellite(DBS) system remains unclear. In this paper we discuss possible choices of caching and multicasting schemes, motivated by current applications in the terrestrial Internet, that could be considered for a DBS system. We examine their advantages and disadvantages as well as the tradeoffs involved in combinations of different approaches. We also propose some uses of these technologies and describe an architecture that enhances the performance and efficiency of a DBS system.

1. Introduction

Direct Broadcast Satellite(DBS) systems provide a way of distributing information to a large number of users over a large geographical area. A DBS system is characterized by a single uplink site, which is also the Network Operations Center(NOC), and end-user systems(possibly thousands), each of which include a receive-only satellite dish and a back channel(e.g., telephone line) for communication from the end user system to the NOC. Such a system is illustrated in Figure 1. The DirecPC™ developed by Hughes Network Systems(HNS) is a practical implementation of such a DBS system. In its present form, such a system has been shown to provide data rates of up to 400kbps [6, 2].

However, protocols such as TCP, which were primarily designed for terrestrial environments, do not perform as well in a satellite based system. Optimization schemes have been proposed to overcome this problem [9] [12]. Motivated by the extensive study and vast deployment of caching and multicasting in terrestrial networks [1, 16], we argue that these technologies are equally, if not more essential for a DBS system for the following reasons. First, delay in a DBS system is more prominent than in a pure terrestrial network due to propagation delay incurred by a GEO satellite. By moving resources and information closer to the end user, caching provides a potential means of reducing the delay remarkably. Secondly, efficient bandwidth utilization is more important in a DBS system because of limited satellite bandwidth. Using broadcast medium (satellite channel) to provide unicast service (web browsing) is an apparent waste, so it is necessary to further exploit multicasting techniques for DBS systems.

This is nonetheless a nontrivial problem. While many aspects of caching and multicasting are well understood in

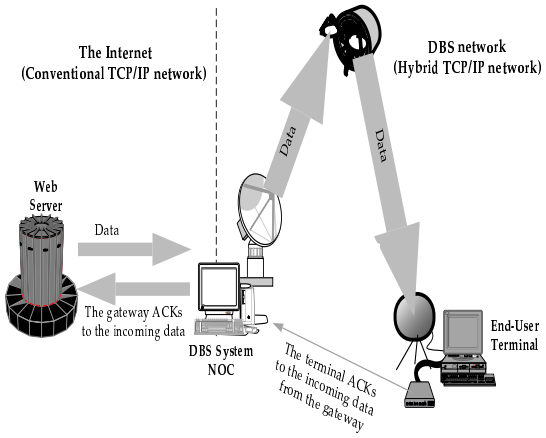

Figure 1. Overview of a Typical Internet over Satellite DBS System

[1]The work described here was carried out as part of an effort at CSHCN to extend the DirecPC™ system jointly developed with Hughes Network Systems in 1992.

[2]Research supported in part by NASA through cooperative agreement NCC3-528, Hughes Network Systems, and the State of Maryland through the Maryland Industrial Partnerships (MIPS) program.

a terrestrial network, they are relatively less studied for a DBS system. Further investigation is needed to choose the right combination of technologies and to integrate them.

In this paper, we examine schemes originated from the terrestrial Internet community for their applicability to DBS systems, and propose different ways of using them. We also show how caching and multicasting are closely related and should be used together.

It is beyond the scope of this paper to discuss in depth the detailed implementation issues of each of the proposed schemes. Rather, we discuss their pros, cons and applicability from a systems engineering point of view. Also, our analysis is more conceptual and intuitive rather than rigorous and mathematical at the current stage.

We focus on latency reduction and efficient bandwidth utilization as well as their tradeoffs in a DBS system. In Section 2 and 3 we discuss caching and multicasting strategies. Section 4 presents the integrated system, incorporating caching and multicasting. In Section 5 we give our conclusions and discuss future work.

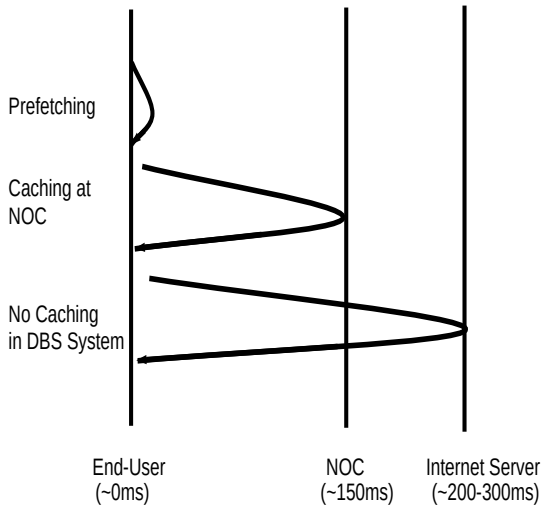

Figure 2. Contributors to Latency in a DBS System

2. Reducing Latency – Caching

Latency is defined as the interval between the time the user requests for certain web content and the time at which it appears in the user browser. In a DBS system, we divide this latency into three components: the delay for the user request to reach the NOC; the delay for fetching the data from the web server to the NOC; and the delay in sending data from the NOC to the end host over satellite.

The first and third parts of the delay are fixed, but can be avoided by prefetching and creating a local cache on each of the individual end user systems. The solution for reducing the second component of latency is to place a cache at the NOC, similar to other adaptive web caching systems[awc,zmnr]. This will result in lower average latency and in the best case the second latency component will be nearly zero.

Therefore, ideally, when a user generates a request, we would like the request to be satisfied locally via a hit in the prefetch cache. If this fails, then we would like the request to be satisfied at the NOC without have to go through the Internet. If this fails again, which is the worst case scenario, then data is fetched from some server in the Internet and transferred to the user. This discussion is illustrated in Figure 2. The different curves indicate the different amounts of latency that are introduced depending on where the data is located. Figure 2 shows that the main contributor of latency is the satellite link. This is sufficient to justify the importance of prefetching as the most important factor in reducing latency.

The overall DBS latency could be reduced even in the cases where there is a miss in the prefetched cache by proper use of multicasting, as we describe later.

2.1. Prefetching

Prefetching of web documents to the end host has been recommended in previous studies as a means of reducing the latency in a terrestrial based web access system [8].

Traditionally, prefetching is considered a ”pull” operation, where objects are actively fetched before they are required. In our case, prefetching does not involve an explicit ”pull”. Instead end user systems speculatively pick objects from a broadcast stream. If prefetching is perfect, an end user perceives almost zero latency. In a DBS system, this technique can provide maximum benefit as it can completely nullify the negative impact of the presence of the satellite link. A separate channel can be set aside for the purpose of broadcasting prefetched data. The end hosts choose to subscribe to this channel and constantly update the contents of their local cache.

However simple prefetching initially may seem, upon deeper analysis one finds several nontrivial issues. These issues include: what should be broadcast; what should be saved in the local cache on the end-host; how to update the prefetch information at the NOC; how should the pricing structure be changed; should the prefetching be global or per user; what prediction algorithm should be used. We will try to answer these questions by formulating an appropriately detailed prefetch strategy.

The first important question regarding any prefetch strategy is what data should be prefetched. One approach is to maintain per-user profiles at the NOC. These would supply usage patterns which could be used by the prediction algo-

rithms. However, maintaining such information for a potentially large number of users is not scalable. Alternatively, a global profile for all users could be maintained. However, as usage patterns may be very diverse, the benefits of such a global profile might be limited.

A more efficient method would be to use both global and per user profiles. In this dual system, the local prefetch cache is built by selecting data being broadcast on the prefetch channel. The NOC maintains a global profile and broadcasts objects based on this information. Each end user system maintains a per user profile to help it select objects from the broadcast stream. Part of the data in the local cache is stored because it has a high rank in the global profile. The remaining is selected out of the prefetch channel because of its high rank in the local user profile.

Updating the prefetch filter at the end host is performed by the prefetch prediction algorithm. A form of Prediction by Partial Match (PPM) algorithm, could be used. Basically, the algorithm maintains a data structure of that keeps track of the URL's following another URL. This information can be used to guess user browsing patterns, and determine the filter which obtains information from the prefetch channel and places it in the local cache. This algorithm has been described in more detail in [5].

The contents of the prefetch channel at the NOC can be updated on the basis of a simple "most popular data" scheme which represents the global profile.

2.2. NOC Caching

To avoid fetching data from Internet servers, caching should be performed at the NOC. This is a conventional terrestrial cache and may be connected to other terrestrial caches as a part of distributed web caching system. There has been extensive study and implementation on adaptive web caching. Interested readers are referred to [1, 16].

It's worth pointing out that NOC caching does not suffer as much from the disadvantages of a conventional terrestrial caching system, which forces web requests to be routed to the cache first and might increase web access latency in case of a miss. In a DBS system, NOC is a natural collector of all requests and introducing caches at the NOC only results in minimal latency increase. Moreover, a NOC has complete information on traffic patterns and statistics, so adding caches at the NOC can provide significant improvement in reducing latency.

2.3. Hierarchical Caching Architecture

To summarize, our hierarchical caching architecture is shown in Figure 3. Each end system is equipped with local prefetch cache, constituting the first layer caching. Caches

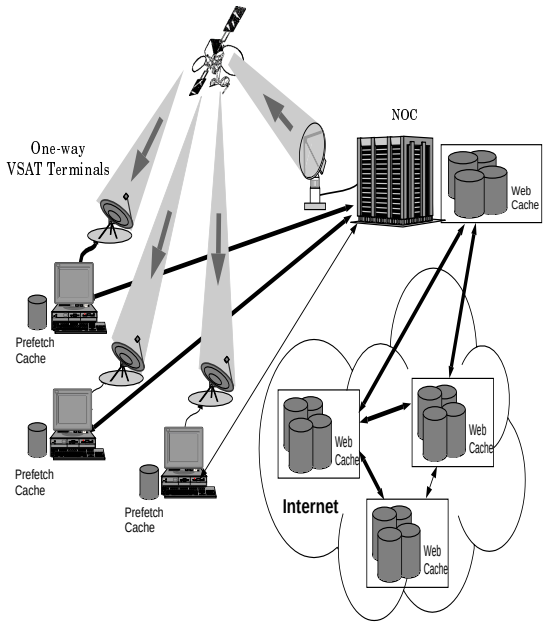

Figure 3. Overview of Caching in a DBS System

inside NOC and various Internet caches form the second and third layer caching, respectively.

While the use of prefetching reduces the average total latency for web browsing, it does involve using sophisticated algorithms both at the NOC as well as at the end user systems. Moreover, a certain amount of bandwidth has to be dedicated for this purpose. The performance gain from the use of this technique will also vary on how much disk space is allocated at the end user systems for the purpose of building these local caches. The DBS system provider can offer users incentive for allocating larger space on disks by offering lower prices for higher hit rates in these local caches.

Caching at the NOC is perhaps the simplest among these techniques to implement. However the cache size and depth of caching hierarchy should be designed together with prefetching and multicasting.

We give a simple comparison of perceived latency with and without caching. Let t_{SAT} stand for the round trip time from the end host to the NOC. This includes the time from host to NOC via terrestrial wire and the time from NOC to host via satellite. Let t_{INT} denote the round trip time for NOC to fetch data from the Internet. Also let t_{PF} and t_{NOC} indicate the additional cache processing time incurred at the local prefetch cache and the NOC, respectively.

Using the proposed caching architecture, the total la-

tency D_C can be calculated as follows in different cases:

$$D_{C,M} = \begin{cases} t_{PF} & best \\ t_{PF} + t_{SAT} + t_{NOC} & \\ t_{PF} + t_{SAT} + t_{NOC} + t_{INT} & worst \end{cases}$$

where best case indicates all requests are satisfied locally. The middle case is where requests can be satisfied at the NOC, and the worst case scenario is where data is fetched from the Internet. Whereas without caching, the latency D is always $D = t_{SAT} + t_{INT}$.

It is clear that the efficiency of caching depends on the probability that requests can be satisfied locally or at the NOC, i.e., the hit rate of the caches.

3. Efficient Bandwidth Utilization – Multicasting

The broadcast nature of the DBS system naturally lends itself to the use of multicasting to save bandwidth. The use of multicast is also motivated by the fact that a large percentage of web requests are concentrated on a relatively small set of highly requested documents [3, 4, 11, 7].

Normally, in a DBS system, a certain amount of bandwidth is allocated for continuous broadcasting of certain contents, e.g., news, sports, stock quotes, etc.. Users subscribed to such services can tune to the designated channel and get information directly. However, the volume of data that can be accommodated into this channel is limited because the broadcasting cycle determines the average waiting time (half the cycle) for a user to get the desired content. Furthermore, this is only suitable for contents whose request pattern is well known and does not change very frequently.

For contents less frequently requested or whose request pattern can change dramatically within short time periods, we argue that more adaptive schemes using on-line request pattern estimates to dynamically change broadcast schedules are needed. This has led to the following solutions: continuous multicasting/air caching and spontaneous multicasting, discussed in subsections.

Note that the degree of validity and performance improvement from these proposed approaches depends on the actual request pattern seen by the NOC, which can be affected by various social, cultural and political reasons. Additionally, performance can be affected by charging and pricing strategies imposed by DBS system providers and ISPs. Therefore, evaluation of approaches to save bandwidth and reduce latency should be studied with traffic analysis, dynamic bandwidth allocation among different unicast and multicast channels, and effective charging and pricing schemes for unicast and multicast applications. As a first step, we are currently in the process of analyzing real user request traffic traces obtained from DirecPC™ NOC. The results could lead to better approaches towards incorporating multicast in a DBS environment.

3.1. Air Cache – Continuous Multicast

Detailed description of Air Cache can be found in [15, 14]. The basic idea is to use satellite bandwidth as a "cache" – Air Cache – to "store" data by continuously broadcasting them. This scheme also includes an algorithm which adapts the contents of the air cache based on the "misses". In this context, "misses" are explicit data requests. The algorithm has been shown to perform reasonably well when compared to a system that has complete information on both hits and misses [15].

A brief sketch of the air cache operation in a DBS system is as follows. The data set is divided into three subsets: vapor, liquid and frigid data depending on their frequency of being requested. Data items are constantly examined and are subject to move from one subset to another based on their request pattern seen by the NOC. Data belonging in the "vapor" set is continuously and repeatedly broadcast. If NOC receives a request for data that is currently being broadcast, it simply drops the request. In this case the end user is required to listen to the channel and get the content from the air cache. This process involves waiting on an average, half of the time of the broadcast cycle. If the NOC receives a request for data that is not in the air cache, then it will be unicast to the user. At the same time, statistics are updated to decide whether the requested data should be reassigned to a different subset. The longer the multicast period, the larger amount of data this air cache can hold which means a higher hit rate.

This method is called Air Cache but we also recognize it as a type of multicast because this approach is essentially continuous multicast push. This approach has potential application in broadcasting databases but has limited usage in web browsing. This is primarily because database applications can tolerate larger delay and so the air cache can contain large volumes of data. For web applications, it is necessary to restrict the size of the air cache so that the broadcast period is smaller than the actual time it might take for the NOC to fetch data from the server in the Internet. Otherwise unicast will always outperform multicast which offers no incentive for using multicast from a user's point of view.

The Air Caching/Continuous Multicast system adds another level of caching and its use as a cache is limited compared with the additional complexity this adds to a DBS system. However, properly configured, this method can offer very good performance improvement if requested data are relatively highly concentrated in periods of time.

3.2. Spontaneous Multicast

An alternative approach to continuous multicast is to aggregate multiple requests for the same content at the NOC and then reply with a single multicast transmission, assuming that certain requests arrive in bursts or clusters. This could potentially provide significant increase in bandwidth utilization. Similar work can be found in [13].

On receiving a web request, the NOC waits for a fixed or variable amount of time for other requests for the same content, while fetching the content from either cache or some Internet server. If there are other requests for the same content, then the NOC replies via terrestrial back channel to the end user by sending the multicast address and the time of the multicast. The NOC then proceeds to transmit data on that multicast address. Thus multicast groups are formed spontaneously at the NOC and by a single transmission, multiple end users receive the requested data.

Note that the aggregation of requests is not limited to requests for exactly the same contents. Requests for closely related (see prefetching) contents can also be grouped together. In this way, for each user the requested content is multicast and relevant contents are prefetched.

Clearly by using spontaneous multicast, the bandwidth saving is obtained at the expense of larger delay that might be experienced by a subset of the users. The tradeoff lies in the design parameter: the amount of time the NOC waits and holds the data before transmitting it. If this parameter is less than the average time for NOC to fetch the data from cache or from Internet server, then latency will not be affected but we may not be able to collect enough requests to justify bandwidth savings – this in fact becomes normal unicast.

3.3. Channel Allocation

The resulting channel allocation from our discussion is shown in Figure 4. The dotted line in the allocation map indicates that the bandwidth allocation between different channels is not fixed and should be done dynamically based on estimated actual traffic pattern and the system cost/performance specifications. The NOC periodically decides whether certain contents should be prefetched, continuously multicast, spontaneously multicast or unicast depending on the demand and changes in demand. For example, increasing demand for some unicast data can result in NOC's decision to multicast them or even to prefetch them to groups of users, and vice versa. Quantified analysis on switching between channels would require further study, which is beyond the scope of this paper but is one of our ongoing projects.

Additional latency is incurred by using multicast due to the constant tradeoff between latency and bandwidth efficiency. We introduce two new parameters: t_{cyc}, the continuous multicast cycle and t_{agg}, the time NOC waits to aggregate requests before it sends data in spontaneous multicast.

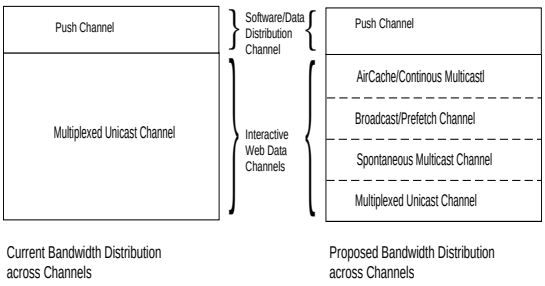

Figure 4. Efficient Distribution of Available Bandwidth in a DBS System

When no caching is used, the average latency D_M is expressed as

$$D_M = \min\{\frac{1}{2}t_{cyc},\ t_{SAT} + \max\{t_{INT}, t_{agg}\}\}$$

where the first term, half the multicast cycle, is the average waiting time for the end system to get contents directly from the continuous multicasting channel. The second term is the time to fetch data via NOC.

If in addition to this we use the proposed hierarchical caching, then the resulting latency under different cases are as follows:

$$D_{C,M} = \begin{cases} \min\{\frac{1}{2}t_{cyc}, t_{PF}\} \\ t_{PF} + t_{SAT} + \max\{t_{NOC}, t_{agg}\} \\ t_{PF} + t_{SAT} + \max\{t_{NOC} + t_{INT}, t_{agg}\} \end{cases}$$

where the first case is where the end system can get required contents either from the local prefetching cache or from the continuous multicast channel. The second case is where the request is satisfied by the NOC cache, and the last case is where the contents have to be fetched from a Internet server.

Our analysis is by no means refined or thorough, but it gives the starting point for future work in systems engineering and tradeoff study. For approaches on bandwidth allocation between unicast and multicast, see [10].

4. Integrated Caching and Multicasting System

Our goal is to integrate these various options into one high performance system taking into the account of their costs. We recognize that even when optimal performance is theoretically achievable, sub-optimal operating decisions are often made by service providers for the sake of simplicity or ease of implementation.

An example system combining caching and multicasting functions as follows. The user generates a request for data, at first the local cache is checked to find if the document exists there. This local cache is maintained by the the prefetch or push data from the NOC. If the document is not found in the local cache, the request is forwarded to the NOC. At the same time, while waiting for the reply from the NOC, the Air Cache channel is scanned to check if it contains the required data. When the request reaches the NOC it checks to see if the request was for a document that is present in the Air Cache database, if so, then the request is dropped. Otherwise, the NOC checks to see if the request falls into any of the request cluster that it is currently building, if so the NOC replies to the end user with the multicast address it is going to use to multicast a reply to that request. If no such cluster is currently present, then the NOC creates a new cluster and sets a spontaneous multicast timer for it. On the expiry of a timer, the NOC multicasts the requested set of data to the end users. While waiting for the spontaneous multicast timer to expire the NOC locates the data in the cache at the NOC, or contacts neighboring caches or the Internet server for it.

5. Conclusion and Future Work

In this paper, we examined possible choices of caching and multicasting schemes that can be applied to a DBS system. We discussed their advantages and disadvantages in reducing overall system latency and improving bandwidth utilization. We also discussed their interoperability and presented an integrated system architecture.

As we have pointed out in previous sections, this paper serves as our first step toward a thorough and in-depth study of how to improve the performance and enlarge the capacity of DBS systems under current implementation constraints. This includes detailed traffic analysis, bandwidth allocation study, unicast and multicast pricing and charging.

Acknowledgment

We would like to thank Vijay G. Bharadwaj and Stephen M. Payne for helpful comments and discussions.

References

[1] The adaptive web caching home page. http://irl.cs.ucla.edu/AWC.

[2] V. Arora, N. Suphasindhu, D. Dillon, and J. Baras. Effective extensions of internet in hybrid satellite-terrestrial networks. *Procedings of the 1st Conference of Commercial Development of Space*, 1996.

[3] P. Barford, A. Bestavros, A. Bradley, and M. Crovella. Changes in web client access patterns: Characteristics and caching implications. *To appear in, World Wide Web, Special Issue on Characterization and Performance Evaluation*, 1999.

[4] A. Chankhunthod and M. Schwartz. A hierarchical internet object cache. Proceedings of USENIX'96.

[5] K. Curewitz, P. Krishnan, and J. Vitter. Practical prefetching via data compression. *Proceedings of SIGMOD'93*, pages 257–266, May 1993.

[6] A. Falk, N. Suphasindhu, D. Dillon, and J. Baras. A system design for a hybrid network terminal using asymmetric tcp/ip to support internet applications. *Proceedings of the Conference Technology 2004*, 1994.

[7] G. J. and M. Seltzer. The case for geographical push-caching. *Proceedings of the Fifth Annual Workshop on Hot Operating Systems, Orcas Island, WA*, pages 51–55, May 1995.

[8] Q. Jacobson and P. Cao. Potential and limits of web prefetching between low-bandwidth clients and proxies. *To appear in Proceedings of SIGMETRICS'99, Atlanta,GA*, May 1999.

[9] V. Jacobson, R. Braden, and D. Borman. RFC 1323: TCP extensions for high performace. ftp://ftp.internic.net/rfc/rfc1323.txt, May 1992.

[10] A. Legout, J. Nonnenmacher, and E. Biersack. Bandwidth allocation policies for unicast and multicast. *INFOCOM'99, New York, NY*, Mar 1999.

[11] B. M. and R. Alonso. Dynamic hierarchical caching in large-scale distributed file systems. *Proceedings of the 12th International Conference on Distributed Computing Systems, Yokohama, Japan*, pages 521–528, June 1992.

[12] S. F. M. Mathis, J. Mahdavi and A. Romanow. TCP selective acknowledgment options. *RFC 2018*, October 1996.

[13] J. Nonnenmacher and E. Biersack. Asynchrous multicast push: Amp. *Proceedings of ICCC'97, Cannes, France*, November 1997.

[14] K. Stathatos. *Air-Caching: Adaptive Hybrid Data Delivery*. PhD thesis, Institute for Systems Research, University of Maryland, College Park, Maryland, 1999.

[15] K. Stathatos, N. Roussopoulos, and J. Baras. Adaptive data broadcasting using air-cache. *Proceedings of First International Workshop on Satellite based Information Services(WOSBIS'96), Rye, New York*, pages 30–37, Nov 1996.

[16] L. Zhang, S. Michel, K. Nguyen, and A. Rosenstein. Adaptive web caching: Towards a new global caching architecture. *Proceedings of the 3rd International WWW Caching Workshop, Manchester, UK*, July 1998.