BUDDIES: Bus Driven Data Dissemination System

Gitika Aggarwal Krishnaveni Budati Kamalakar Karlapalem
Centre for Data Engineering
International Institute of Information Technology
{gitika,krishnaveni}@students.iiit.net, kamal@iiit.net

ABSTRACT

The dramatic improvements in global interconnectivity due to intranets and the Internet have led to an explosion in the number and variety of data-intensive applications. More importantly, the large volumes of data available for general access and the rapidity with which these data change make it very difficult for users to track the data that they are interested in. Also, scalability is a central requirement in dissemination systems due to the huge number of users and large amount of information. This is demonstrated by frequent delays and service disruptions when accessing shared networked data sources. To overcome scalability issues, considerable research has been going on to minimize redundant server operations and maximize and prioritize bandwidth utilization while aiming for a reasonable (may not be the best) response time. In this paper, we introduce the idea of BUDDIES, a new protocol for data dissemination that can also be extended to automatic data dissemination and differentiated services.

Keywords

Data dissemination, data transfer, web data access.

1. INTRODUCTION

Various data applications like servers that provide online monitoring of election results and results of cricket matches suffer from scalability problems. This issue of scalability manifests itself mainly due to the ever increasing size of the user base, which results in several serious problems like server overloading and clogging of the network leading to unacceptably high response times and sometimes, never ending waits (maybe because of server crashes). Therefore, to overcome these issues concerning scalability, massive work on efficient data dissemination has been going on. Several server side-scheduling algorithms [1, 2, 8, and 9] have been developed that aim at reducing redundant server side operations by calculating the results to a query once and disseminating the results either by broadcast or multicast.

The ability of a web service to provide reasonably low response times to access its contents is constrained by available network bandwidth. It is important for the service to manage available bandwidth wisely. In the past, web sites have used ad-hoc solutions to deal with peak loads or in general to come up with a strategy to tackle user requests. Differentiated service can allow the system to provide better service for certain customers while gracefully degrading the service for other less important customers. While providing differentiated quality of service is typically enforced through network mechanisms, some systems have been introduced that provide a robust mechanism for managing network resources at the application level. One such system [3] uses a method called Quality Aware Transcoding that allows web servers to customize the size of objects constituting a web page, and hence the bandwidth consumed by that page, by dynamically varying the size of multimedia objects on a per-client basis.

Another pressing issue is that of automatic data dissemination. The tools and systems for information dissemination are becoming crucial in a variety of application environments in both the Internet and intranets [4]. In general, users subscribe to such systems by providing lists of topics they are interested in. Whenever new data is available for distribution, the system disseminates it to the users who may be interested in such data based on the user preferences. Several such systems have been developed [4, 5 and 6]. However, the method of user subscription provides very static user profiles. In the case where in the users' interests change dynamically, this method will need a more efficient and user-friendly alternative so as to facilitate the users in specifying their preferences. Also, as the user base increases, scalability issues also come in to the picture

The remainder of the document is organized as follows. Section 2 introduces the core idea of BUDDIES, explains the terminology used in the subsequent sections and presents the architecture of BUDDIES. Section 3 describes the simulation model developed, presents the issues of concern and the simulation results. Section 4 concludes the document with a note on the future work.

2. BUDDIES

In the system developed by Wolf et al [7], the users in the network are grouped based on some heuristic (like physical proximity) and each group is assigned a router. BUDDIES combines the idea of grouping users along with the idea of data multicast. However, instead of grouping users purely on the basis of physical proximity, BUDDIES groups them on the basis of similar interests as well as proximity. For example, consider two kinds of data – election results and cricket scores. All sites that are interested in the election results and are also sufficiently close to one another fall in one group. Sites those are interested in cricket scores are grouped similarly. The main point behind grouping sites like this is to try to avoid redundant server operations (section 2.1) as far as possible. Therefore, if one can group users with similar interests and string those along one or more virtual network routes, then sending just one packet of data along this route may suffice, with a little compromise on the response time. This will not only save a number of redundant server operations but will also save on the network bandwidth.

2.1 Terminology

Redundant Server Operations - The operations executed by a server are said to be redundant when it has to execute the same query again and again but for different users, even when these query requests are not much separated in time. Several factors determine the minimum distance between two query requests such as desired response time. The role of response time is explained more in section 2.2.

Bus - A data structure that is a wrapper around a set of queries and a set of query results. The bus hops from one site to another along a pre-specified route.

Route - A virtual path, connecting sites having common interests or that exchange some data that is of common interest and are also reasonably close to each other with respect to the network topology. The routes may be cyclic or acyclic.

Hub - The sites those are capable of query processing. If a hub does not have the data required to process a query, it will forward the query to another hub that can either process this query or that will forward it. Hubs can accept queries from users.

Host - The sites that cannot process any queries. These are only client machines that accept queries from the user and forward those to the designated hub (explained later) and send the query results back to the user.

Terminal – The sites that form the beginning and end points for a bus along a route. In the case of an acyclic route, there will be two terminals, which are the first and the last sites along the route that constitute the beginning and the end points of the route. For a cyclic route, both the terminal points are the same.

Query - A client request for data. The type of query depends on the particular domain. It may be a simple set of keywords or it can also be an SQL query. Each query typically accesses some data objects that may be present either at a single hub or may be distributed at different hubs. A data object may be a relational table or a flat file (or any other form of data chunk), depending on the domain.

Database – The entire collection of data objects distributed among the hubs.

Query frequency - The frequency with which a host/hub gets query requests from users. This is measured in terms of the inter-arrival time between successive query requests.

Bus frequency - The frequency at which a bus is generated at the beginning terminal point along a route. This is measured in terms of the time between successive bus generation events.

The higher the inter arrival time; the lower is the bus/query frequency. In the rest of the document, the terms bus/query frequency and bus/query inter-arrival times are used interchangeably while bearing in mind the interpretation of each term as specified earlier.

Commonality – The set of data items common to two or more queries is called the commonality between these queries. It is expressed as a percentage. For example, if the commonality between two or more queries is 10%, it implies that the intersection of common data objects accessed by these queries is 10% of the total number of data objects accessed by each query.

Response Time - The total time span between the arrival of the user request and the time when the entire result of the query reaches the respective host/hub.

Wait time - The total time span between the arrival of a user request and the time when it is loaded on to a bus.

Knee bus frequency – For a given query frequency, as the bus frequency decreases the average response time increases gradually till a particular value and then increases drastically. This particular value of the bus frequency is called as the knee bus frequency for the given query frequency.

Accumulation effect – When more number of queries get accumulated, there is a high probability that the result of one might satisfy several others. This, leads to a low average response time. This effect is called ‘accumulation effect’.

2.2 Architecture of BUDDIES

Web sites in this system are organized in a two level hierarchy. The topmost level consists of sites acting as hubs, while the second level consists of sites, referred to as hosts which are in the regional area of one or more hubs. Hubs are connected with each other through prefixed routes, referred to as bus routes, based on their common interests and their proximity to each other. Hubs with similar interests and placed very far from each other are connected through different routes. The hosts in a regional area are connected to a particular hub, via routes. The hosts forward their queries to this hub. Figure.1 shows the web system architecture in BUDDIES.

2.2.1 Data Dissemination

The results to the user queries are disseminated as follows:

- A query arriving at a host is forwarded to its designated hub.
- Query and hub metadata relate the query content with certain hubs that can either directly satisfy the query or are known to contain links to hubs that can.
- Buses operating along the various routes take the queries and deposit them at hubs that are either capable of processing the queries or forwarding those to other hubs. (Depending on the query and hub metadata stored in it, a hub may decide what alternative routes should be followed).
- Buses transfer intermediate results found to the hub/host that issued the query.

For example, consider Figure 1, the broken lines represent the routes along which we have hosts/hubs that are interested in election results, whereas the firm lines represent the routes that connect hosts/hubs that are interested in cricket scores. Cluster B represents a group of hosts that are interested in cricket scores. There may be one or more routes connecting these hosts together. The requests generated by these hosts are forwarded to the hub *f*. From this hub, the requests are forwarded to hub *e* along route *9*. Hub e forwards the requests to hub *c* along route *10*, which in turn forwards it to the server *'g'* (which is also a hub). The requests are processed at the server and the results may be returned along the same route or through a different route (that serves similar interests). Similarly, requests for hosts that need election results can be satisfied.

A bus arriving at a hub takes the queries/results along its route, deposits any results, issues new queries to the hub and moves to its next station. Following are the key features of the interaction of a bus with the hubs:

a) Each hub maintains several queues, namely, queues for queries/results to be sent along bus routes, queues for results to be received, and a query execution queue.
b) Results brought by the bus are bulk data from which the query results can be selected. Thus, there is high likelihood of result sharing (saving the cost of re-executing the query from scratch).

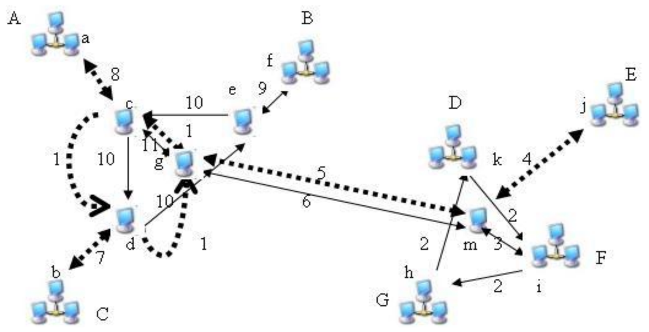

Figure 1: Architecture of BUDDIES

——	Cricket
■ ■ ■	Election
a-m	Hubs
A-G	Groups of hosts

3. SIMULATION MODEL

A simulation model of BUDDIES has been developed using the method of Discrete Event Simulation. As for now, this is a restricted model. It takes in to account a network that consists of 4 Hubs. The hierarchy is not included in the model so as to keep it simple. All the hubs are assumed to have the same interests. They are connected via two acyclic routes. If a, b, c and d are the four hubs, then along one route those are connected as a-b-c-d while the other route connects the hubs as d-c-b-a. Because the routes are acyclic, along each route, there are two terminal points, which are the first and the last sites respectively. The query frequency at each hub is assumed to be a constant and is the same for all the hubs. The bus frequency along each route is also assumed to be a constant and is the same for both the routes. Every query that is posed by a client is assumed to access a fixed a number of data objects.

Let 'B' be the bus inter-arrival time along the two routes and 'Q' be the query inter-arrival time at each hub. A query is modeled as a set of sub-queries, where each sub-query is a request for one of the data objects, that the query accesses. The query is said to be satisfied, only if the requests for all the sub-queries are satisfied.

After every Q milli seconds, a query is generated at each hub. Once a query is generated, it is enqueued in the wait list, maintained for each of the hubs. Using the metadata at the hub, the route along which each of the sub-queries is to be sent is calculated. For each sub-query, it is checked if the same sub-query has already been sent along the particular route. If so, the sub-query is kept on wait for the result, or else the sub-query is enqueued in a queue that maintains all the sub-queries that are to be sent along that route.

After every B milli seconds, a bus is started at each beginning terminal. Starting from the terminal point, the bus hops from one site to the other, depending on the order in which the sites are strung together along the route. When the bus arrives at a particular site the following are done:

1) For every result in the bus, it is checked if there is any sub-query in wait for the result along that route. If so, the sub-query is removed from the wait queue and is said to be satisfied.

2) For every sub-query in the bus, it is checked if the hub can process it. If so, the sub-query is removed from the bus and a request for processing it is placed at the hub.

3) All the sub-queries that are to be sent along that route are loaded onto the bus.

4) When the hub gets a request for processing a sub-query, it retrieves the result of the sub query and enqueues the result into a queue, which maintains the list of results that have to be sent along a particular route. Results from this queue are dequeued one after the other and loaded onto the bus if it has sufficient space.

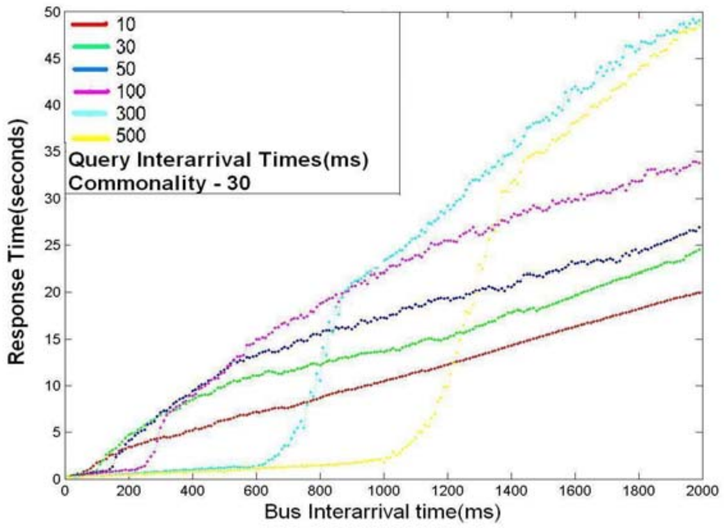

Figure 2: Response times for different query inter-arrival times at commonality 30

3.1 Issues to be addressed

The simulation model aims to resolve the following issues regarding the performance of BUDDIES across various network topologies and application domains:

- For different query frequencies, evaluate the effect of commonality on the average response time.
- Evaluate the behavior of average response times for different query frequencies, when the average size of a data object is large/small and the network bandwidth is high/low.

- Given a query frequency, the maximum tolerable response time, the commonality level and various other factors like average size of the data objects and network bandwidth, find the operable bus frequencies and the optimum bus size.

3.2 Results and Observations

As of now, the simulation model has been tested for two different scenarios.

Scenario 1: The size of each of the data objects in the database is small, the network bandwidth is low and the number of data objects that a query accesses is small. This is analogous to the case in which a server disseminating cricket scores has to handle a very large number of requests and the allocated bandwidth is considerably low.

To simulate this scenario, the values of the different parameters of the model are as follows:

Size of each of the data objects	1000 bytes
Size of the database	200 data objects
No. of data objects accessed by a query	10 data objects
Maximum no. of queries per bus	40
Maximum no. of query results per bus	4
Network bandwidth	0.8kbps

Figure2 and Figure3 show the results for scenario 1.

From Figure2 it is observed that:

1) For a given query frequency, as the bus frequency decreases, the response time increases gradually till a particular bus frequency (knee bus frequency) and then shoots up.

2) At higher bus frequencies, higher query frequencies have higher response times than those at the lower ones. However, for considerably low bus frequencies, the higher query frequencies show lower response times than those at the lower ones.

3) At higher query frequencies the knee bus frequency is almost 3-4 times lower than the query frequency, whereas for lower query frequency it is only 2-2.5 times lower than the query frequency.

The total response time of a query can be divided in to 3 parts, namely, wait time, processing time (at the intermediary and final hubs) and transmission time (time taken to transmit the results from the server to the client along a route). As the bus frequency decreases, not only more queries get accumulated, but also the average wait time per query also increases. Below a certain bus frequency, the wait time starts dominating the accumulation effect and hence the response time shows a sharp increase beyond the knee frequency.

At low query frequencies, there is less accumulation of common queries, because of which more number of results may have to be transmitted. This increases the average transmission time per query. Because the network is a very slow one, the high transmission time dominates the total response time. Hence, at lower query frequencies, the response time increases at a faster rate when compared to that at higher query frequencies. In fact, below a certain bus frequency, it is higher than that at high query frequencies.

On comparing Figures 2 and 3 the following observations are made.

1) For a given query frequency, the knee bus frequency is lower at a higher commonality, when compared to that at a lower commonality.

2) For a given query frequency, at any bus frequency the response time is low when the commonality is high and vice-versa.

In this case, because the commonality is higher, the average response time decreases due to higher accumulation effect. The same explains for the shift in the knee bus frequency for a given query frequency.

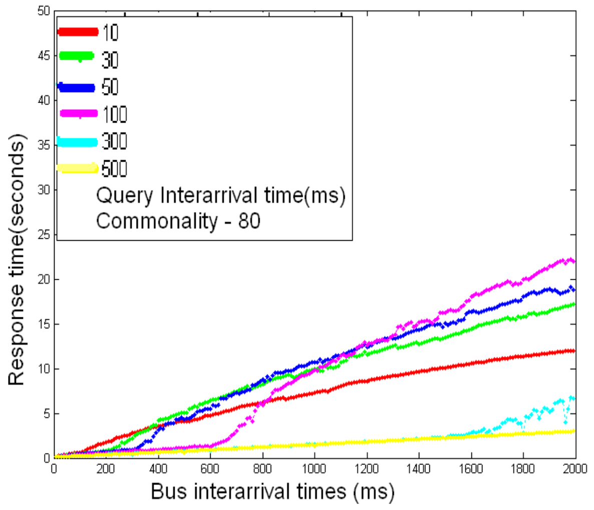

Figure 3: Response times for different query inter-arrival times at commonality 80.

Scenario 2: The size of each of the data objects in the database is large, the network bandwidth and the number of data objects that a query accesses is high. This is analogous to a situation in which the data of interest is multimedia files of very large sizes, when the bandwidth available is high and the response time should be reasonable, even at peak loads.

To simulate this scenario, the values of the parameters taken are as follows:

Size of each of the data objects	50,000 bytes
Size of the database	1000 data objects
No. of data objects accessed by a query	50 data objects
Maximum no. of queries per bus	500
Maximum no. of query results per bus	10

Network bandwidth	50kbps

Figures 4 and 5 show the results for scenario 2

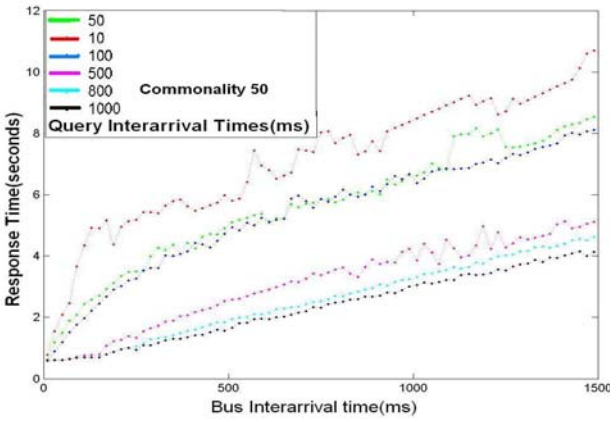

Figure 4: Response times for different query inter-arrival times at commonality 50

The following are observed from Figure 4:

1) For a given query frequency, as the bus frequency decreases, the average response time gradually goes up.

2) The average response times for higher query frequencies are higher than those of the lower frequencies.

For a given query frequency, as the bus frequency decreases the average wait time increases, thus increasing the average response time. As the query frequency increases, the average wait time and the average processing time per query also increases and vice versa. Although there is higher accumulation of queries at high query frequencies, compared to that at the lower query frequencies, due to the high network bandwidth, average transmission time per query fails to dominate the response time. This results in the response time being always higher at high query frequencies than that at low query frequencies.

Since this is a network with a high bandwidth, one can afford to have high bus frequency, without compromising much on the network resources. This works out fine, because as one can see from Figure4, for a given query frequency, as the bus frequency increases, the response time decreases. To strike a balance between response time and utilization of network resources, one can choose a suitable bus frequency (given the query frequency).

In Figure5, initially, as the query frequency decreases, the lowest bus frequency for the desired response time gradually increases, and beyond a threshold value it settles to an almost constant value. This is explained as follows:

When the query frequency decreases below a particular value (depending on the desired response time), less number of queries will be accumulated before the arrival of a bus. As a result, although total number of queries generated in the network is less, the total number of queries that will need individual processing increases. This also increases the need for more number of buses to ship the results to the originating sites. Hence, to get the desired response time, at low query frequencies, the bus frequency needs to be kept the same as that at higher query frequencies (despite the decrease in query frequency).

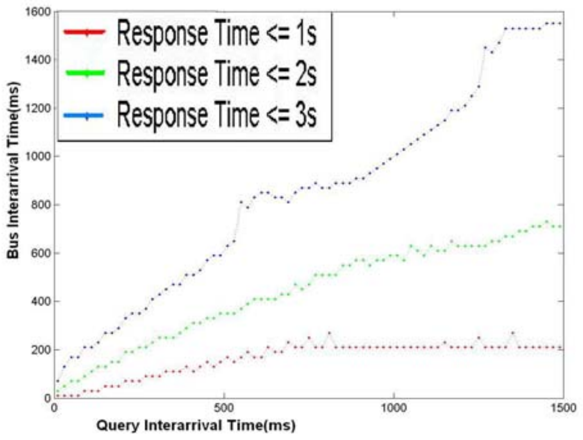

Figure 5: Cut off bus frequencies for different query inter-arrival times at commonality 50.

4. CONCLUSION

We have shown that BUDDIES provides satisfactory results under certain restricted network and domain conditions. Considerable experimentation still remains to be done so that the various parameters (e.g., the optimum bus frequency on a route given the query frequency on that route and the network constraints) can be adjusted accordingly. This experimentation has to extend not only across various network topologies, but also across various domains, so as to see for which kind of applications BUDDIES is best suited to.

The BUDDIES framework can be extended to provide differentiated services. This can be accomplished by using priority-based routes. For a given source and a destination, different routes may have different priority levels, depending on the importance of the data and the users along the particular route. Accordingly, network bandwidth can be distributed among the routes. Thus routes having high bandwidth can operate buses at high frequencies to provide very low response times, whereas those having low bandwidth can operate buses at considerably low frequencies, thus compromising the response time to an extent.

One can also extend the framework to provide automatic data dissemination. For example, consider a cricket result server. Instead of waiting for user requests for cricket scores, the server can send the results on its own accord on the appropriate routes. However, this may increase the data-filtering overhead on the client side. Therefore, the decision regarding what kind of data can be disseminated automatically has to be taken very meticulously.

REFERENCES

[1] H.D. Dykeman, M. Ammar, and J.W.Wong. Scheduling algorithms for videotex systems under broadcast delivery.

In IEEE International Conference on Communications, pages 1847{1851, Toronto, Canada, 1986

[2] J.W.Wong. Broadcast delivery. Proceedings of IEEE, 76(12): 1566{1577, December 1988

[3] Surendar Chandra, Carla Schlatter Ellis, Amin Vahdat. Differentiated Multimedia Web services using quality aware transcoding

[4] T.W. Yan and H. Garcia-Molina. The SIFT Information Dissemination System. ACM TODS, 24(4): 529.565, 1999.

[5] D. Gifford, R. Baldwin, S. Berlin, and J. Lucassen. Architecture for Large Scale Information Systems. In Proc. Of the Symposium on Operating System Principles, pages 161.170, 1985.

[6] D. Goldberg, D. Nichols, B. Oki, and D. Terry. Using Collaborative Filtering to Weave an Information

[7] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Content-based addressing and routing: A general model and its application. Technical Report CU-CS-902-00, Department of Computer Science, University of Colorado, Jan. 2000

[8] C.J. Su and L. Tassiulas. Broadcast scheduling for information distribution. In Proc. IEEE INFOCOM, 1997.

[9] N.H. Vaidya and S. Hameed. Data broadcast in asymmetric wireless environments. In Proc. of Workshop on Satellite-based Information Services (WOSBIS), New York, November 1996.