Bounced - Improving Data Availability through Replication in P2P Networks

Raghav Sethi, Naved Alam, Mayank Pundir, Pushpendra Singh
IIIT-Delhi
New Delhi, India
Email: {raghav09035, naved09028, mayank09025, psingh} @ iiitd.ac.in

***Abstract*— Traditional models of file-sharing are less effective for mobile devices due to the low availability of files on the network. In this paper, we discuss the replication strategy of the *Bounced* file-sharing system, and create a framework to evaluate the benefits of this approach. *Bounced* intelligently replicates requested files on-demand to improve availability of scarce files and increase the probability of a successful transfer.**

I. Introduction

Effective peer-to-peer file-sharing in local area networks such as those of universities and workplaces is becoming an increasingly difficult problem due to the decreased file availability in largely wireless networks.

In this paper, we present the architecture of a file-sharing system *Bounced*, which attempts to mitigate this problem by increasing the availability of scarce files on the network through replication. We also attempt to determine the replication factor and show that the *Bounced* network can yield greater profits than traditional file-sharing systems using a simple economic model.

Although replication is an integral part of several strategies to increase availability of files in P2P file sharing systems[1], our architecture is novel as it augments the standard hybrid-P2P model of DC++ and Napster by giving users the option to request replication of scarce offline files.

II. System Design

Bounced currently consists of a client program for Windows-based PCs and one or more servers that act as the centralized directory for the files on the network. We selected the hybrid-P2P architecture, as institutional networks are significantly smaller in scale and typically do not have the same performance bottleneck issues on the server as internet-based file-sharing.

We utilize the popular Node.js framework to further improve the performance of our server software [2] while also supporting a large number of connected clients.

A. Search

Traditional file-sharing systems display search results for a query that only consists of files that are available on the network at the time of the query. *Bounced* differs in that it also returns files that are unavailable at query time.

If users choose to download a file from an online node, the requesting node connects to the file holder directly and downloads the file normally. For files that are not currently available on the network, instead of giving users the option to download the file, we give them the option to 'bounce' the file instead as explained in Section II-C.

B. Download

The server keeps track of 'bounce' requests as a queue for each node and automatically directs clients to perform the 'bounce' transfers without user intervention. This means that queued transfers begin as soon as both parties are online, thus maximizing the utilization of the limited time that nodes spend online. For online files, the server plays no part – the nodes can simply download from each other as in a traditional network.

To be similarly optimal (even if no replication takes place) in terms of time taken to transfer an offline file, the user of a traditional file-sharing network would have to perform the same search at regular intervals and manually begin the transfer when the file is available.

C. Replication

The *Bounced* server automatically keeps track of 'friendships' among nodes. We define 'friendship' between two nodes as the number of times that these nodes have been online at the same time. When a request for an offline file occurs, the server computes the top-k friends of the requester and stores instructions for each of these friends to download the file from the original file holder.

When any of the k friends of the requester is online with the original file holder, the server will direct the friend to download the file from the holder, replicating the file and completing the first leg of the 'bounce'.

When the original requester is online next, it has the option to download the file from either the original holder or the friend (if they are online), thus increasing the probability that the requester will obtain the file. When the file reaches the original requester, the 'bounce' is said to have completed its second leg. It follows from this that a file will be replicated a maximum of K times. This helps us see that the maximum total data transferred for a file of size s is $(K+1)s$. This worst-case condition (in terms of data transferred) will occur when the original requester receives the file only after all k replicas have been created.

We will now derive a relationship that will determine whether this additional data transfer is profitable.

III. Theoretical Modelling

If p is the probability of a node (selected to serve a given file) being online, then the probability of the file being available on the network is also p. In a traditional file-sharing network, this is also the probability that the requester will get the file.

As described above, the *Bounced* replication strategy will be followed if the file holder is not available on the network. When replication has occurred R times, where $R \leq K$, we see that the probability of obtaining the file becomes $1-[(1-p)^R]$, (the probability that at least one of the holders is online). Rangnathan et al. [3] have discussed a similar model in their work.

It is clear from Figure 1 that for scarce files (files with lower values of p) *Bounced* offers significant benefits in terms of transfer probability. For example, with traditional probability p=0.3 and R=5, we see that *Bounced* increases probability of a successful transfer to 0.832. Therefore, a file in the *Bounced* network has a higher probability of being delivered compared to a traditional network.

Higher the value of R, higher is the probability of a successful transfer. However, more replicas consume more network bandwidth and storage. This implies that we need to find the value of R at which the system will be advantageous even after taking into account replication costs.

For simplicity, we will assume that the transfer cost of any file is C. As described in Section II-C, the worst case cost of using the *Bounced* network is $C\ (k+1)$, while the cost of using a traditional network is C.

In a file sharing network, the value of a file is usually inversely proportional to the time required to obtain it. In other words, the longer a user must wait to receive a file, lower is the value of the file. This assumption is a logical and intuitive one to make; a file is clearly worth more to the user at the time it is requested, rather than, say, a week later.

Let $V(t)$ be a strictly decreasing function of time t, which gives the value of a file at a given time t. Suppose that the initial values of the file is $V(t_0)$ and that the *Bounced* network delivers the file to the user at time t_B with value $V(t_B)$. Similarly, if a traditional network delivers the file at time t_N, the value of the file at the time becomes $V(T_N)$. This loss in value is clearly a factor in determining the cost of a transfer.

Let C_B be the cost of using *Bounced* architecture to get the file and C_N be the cost of using a traditional network. We have:

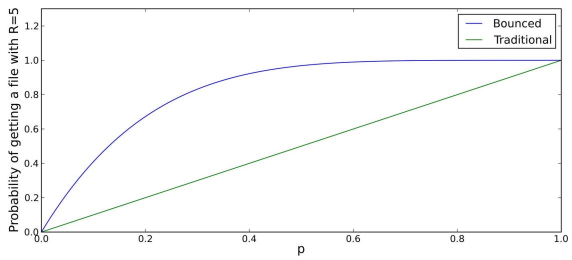

Fig. 1. Probability of a requester obtaining a given file if R=5 as a function of availability. For readily available files, replication will have less effect

$$C_B = (\ (k+1)C)(\ V(t_0) - V(t_B))$$
$$C_N = C\ (\ V(t_0) - V(t_N))$$

From Section II-B, it is clear that $t_N \leq t_B$ and hence that the value of the file when delivered by *Bounced* will be higher or equal to the value when delivered by traditional networks. The *Bounced* architecture will be a better choice if $C_N > C_B$.

$$C\ (V(t_0) - V(t_N)) > ((k+1)C)(V(t_0) - V(t_B))$$
$$(V(t_0) - V(t_N)))/(V(t_0) - V(t_B)) - 1 > k$$

We see that this reduces to an inequality dependent on the value of k and the function $V(t)$. Our aim is to determine the value of k for which the extra cost incurred by the Bounced architecture is less than the decrease in the value of file had that cost not been incurred.

Our planned deployment of *Bounced* in the IIIT-Delhi network will help us derive the function $V(t)$. We will do this by recording all instances of users cancelling existing 'bounce' requests to replicate the requested files. We will assume that the value of the file is zero at the time this occurs – since a cancellation clearly indicates that the user no longer feels the file is of enough value to download.

We will perform regression analysis on these data points and determine $V(t)$. We will then be able to determine the appropriate replication factor k.

IV. Conclusion and Future Work

We have described the need for and the architecture of the *Bounced* file-sharing system and have built a framework to analyze its costs and benefits. We are in the process of rolling out the *Bounced* system to users in the IIIT-Delhi network and will experimentally determine the factors described above by tracking cancellation events in this network.

.

References

[1] S. Haiying."An efficient and adaptive decentralized file replication algorithm in P2P file sharing systems." IEEE Transactions on Parallel and Distributed Systems, 2010, pp. 827-840.

[2] S. Tilkov and S. Vinoski. "Node. js: Using JavaScript to build high-performance network programs." Internet Computing, IEEE 14.6, 2010 pp. 80-83.

[3] K. Ranganathan, A. Iamnitchi, and I. Foster."Improving data availability through dynamic model-driven replication in large peer-to-peer communities." 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, 2002