UsenetDHT: A Low Overhead Usenet Server

Emil Sit, Frank Dabek and James Robertson
MIT Laboratory for Computer Science

1 Introduction

Usenet is a large, distributed, messaging service that serves thousands of sites world wide. Since its introduction in 1981, Usenet has been growing exponentially, doubling in size approximately every ten months. Currently, approximately 1 TB of new Usenet data, corresponding to about 4 million articles, is created per day. Each server that wishes to carry the full content of Usenet (a "full feed") must replicate this amount of data each day, which requires a 100 Mbps connection. Further, to provide any degree of retention, a significant amount of disk storage space is needed.

UsenetDHT provides a service that is substantially similar to Usenet but which reduces aggregate bandwidth and storage requirements by organizing the storage resources of servers into a shared distributed hash table that stores all article data. Instead of being stored at each server, article data is stored on a small number of replicas in the DHT. The DHT allows articles to be distributed geographically and located quickly, maintaining the benefits of wide-area replication without incurring the bandwidth and storage costs of full replication. This storage mechanism obviates the need for article transmission via NNTP [9]. However, the existing flood-fill distribution mechanism is still used to disseminate index-information. In order to preserve the primary interface of Usenet, UsenetDHT servers support the NNTP reader protocol.

This system reduces the storage requirements for a Usenet server — all UsenetDHT servers will be able to provide high-retention access to the entire Usenet feed. For a n-host system, per-host storage requirements are a factor of $2/n$ smaller than those of a traditional Usenet server offering a full-feed.

The bandwidth required to host a Usenet feed using UsenetDHT is proportional to the percentage of articles that are read rather than to the total number of articles posted. Given the size of the current Usenet and known readership patterns, we expect this to translate into a significant reduction in the bandwidth required to host a feed.

The rest of the paper is structured as follows. Section 2 provides a basic overview of Usenet's features and functionality. Our design is fleshed out more completely in Section 3. Our evaluation is presented in Section 4. Section 5 briefly summarizes prior and related work and we conclude in Section 6.

2 Background

2.1 Usenet

Usenet is a highly distributed mesh of servers that are connected in a mostly ad-hoc topology that has evolved since its creation in 1981. Servers are distributed worldwide and traditionally serve readers located in the same administrative realm as the server. Each server peers with its neighbors in the topology to replicate all articles that are posted to Usenet. The servers employ a flood-fill algorithm to ensure that all articles reach all parties: as a server receives new articles (either from local posters or from other feeds), it floods NNTP `CHECK` messages to all its other peers who have expressed interest in the newsgroup containing the article. If the remote peer does not have the message, the server feeds the new article to the peer with the `TAKETHIS` message. Because relationships are long-lived, one peer may batch articles for another when the other server is unavailable. RFC977 [9] and RFC2980 [1] describes the NetNews Transfer Protocol in more detail.

Certain newsgroups are moderated to keep discussions on-topic and spam-free. Moderation is enforced by requiring a special header — articles posted without this header are sent to a pre-configured moderation address instead of being flood-filled as normal.

Special messages called control messages are distributed through the regular distribution channels but have special meaning to servers. There are messages that create new newsgroups, remove existing newsgroups and cancel news postings (i.e. remove them from the index and storage).

2.2 Server-side policies

Each Usenet site administrator has complete control over the other sites that it peers with, what groups it is interested in carrying and the particular articles in those groups that it chooses to keep. This flexibility is important as it allows administrators to utilize local storage capacity in a manner that best suits the needs of the site. For example, commercial Usenet providers will often invest in large amounts of storage in order to be able to retain articles over a longer period of time. Some Usenet providers will choose only to receive articles affiliated with text newsgroups in order to minimize the bandwidth and storage required. Most servers today will also filter incoming articles with CleanFeed to remove articles that are considered to be spam.

2.3 Usenet data characterization

The size of Usenet is a moving target; many estimates have shown growth by a factor of two in traffic volume every 10–12 months. Growth has largely been driven by an increase postings of "binaries" — these are encoded versions of digital media, such as pictures, audio files, and movies. Users are increasingly taking advantage of Usenet as a distribution mechanism for large multimedia files. In 2001, there was an average of 300 GB of "binary" articles posted per day, but present day volume is closer to 1 TB [8]. In contrast, the volume of text articles has remained relatively stable for the past few years at approximately 1 GB of new text data, from approximately 300,000 articles [7].

2.4 Distributed hash tables

A distributed hash table (DHT) is a peer-to-peer storage system that offers a simple `put` and `get` interface to client applications for storing and retrieving data. DHTs are often built out of a robust self-organizing lookup system (such as Chord [13]) and a storage layer (such as DHash [4]).

The lookup layer maintains state information about nodes and their identifiers while the storage layer automatically handles balancing storage load between the participants in the system. The storage layer automatically rebalances data as nodes join and leave the system. In order to keep data reliably available, DHTs will typically use replication or erasure codes to store data on geographically distributed machines. This also allows DHTs to optimize reads by using nearby servers.

3 Architecture

The main goal of this system is to reduce the resources consumed by Usenet servers — specifically, the data transfer bandwidth and the on-disk storage requirements — while mostly preserving the major semantic features of Usenet. UsenetDHT accomplishes this by replacing the local article storage at each server with shared storage provided by a DHT. This approach saves storage since articles are no longer massively replicated; it also saves bandwidth since servers only download articles that their clients actually read.

Article announcements and control data are exchanged through the existing Usenet mesh network. This allows cancellation and moderation of articles to work much as they do now. We considered using application-level broadcast to transmit meta-data. An efficient broadcast tree would reduce link stress by ensuring that data is transmitted over each link only once. However, Section 4 shows that the meta-data streams are relatively small, so the bandwidth reduction does not outweigh the complexities introduced by such a system. Further, the mesh network system is more robust, so we chose to leverage the existing infrastructure.

Figure 1 illustrates this design. Dotted-lines in the figure denote the transfer of meta-data, and solid lines denote article data. In the UsenetDHT design, each node donates storage capacity to form a DHT, which acts like virtual shared disk (right). Articles are stored in the DHT and fetched as required to satisfy read requests.

Consider how a message is posted and eventually read using UsenetDHT. A newsreader in communication with a local server posts an article using the standard NNTP protocol. By design, the reader is unaware that the server is part of UsenetDHT, instead of a standard server. Upon receiving the posting, the server stores the article data in the DHT under the content-hash of an inode-like structure that points to fixed size chunks of the article [4]. As an optimization, if the article fits within a single 8KB DHT block, the body is stored directly in the inode [6]. The use of a secure hash function such as SHA-1 to produce the content hash guarantees that articles will be evenly distributed through the nodes in the system. It also ensures that servers can test the validity of data received over the network.

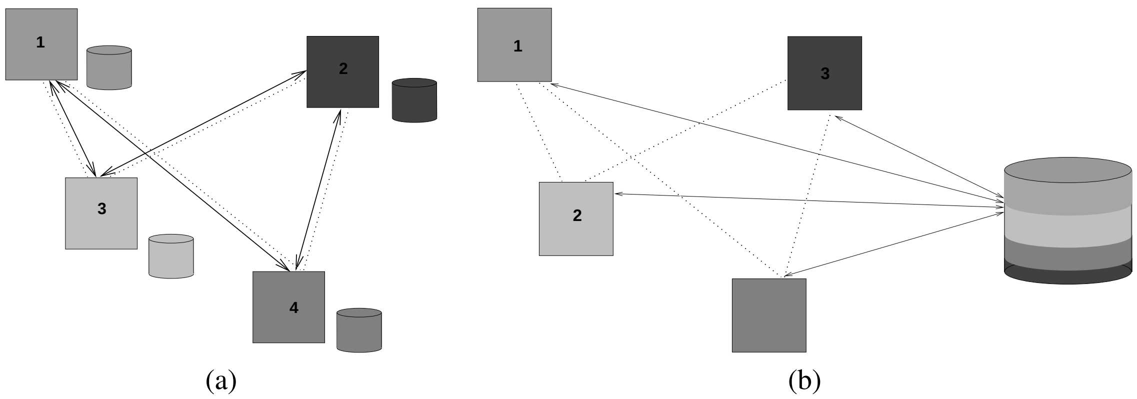

Figure 1: The design of Usenet (a) and UsenetDHT (b). Usenet servers are connected to form a mesh network and exchange article data (solid lines) and meta-data such as article announcements and cancellations (dotted lines). Each Usenet server provides enough storage for an entire feed. UsenetDHT servers are connected by the same mesh network to exchange meta-data, but coalesce their storage into a large, shared DHT (at right).

After the block has been successfully stored in the DHT, the server sends an announcement with the article's header information and content hash to its peers. This information is sufficient for the peer servers to insert the article into the appropriate group indices, provide a summary of the article to readers connecting to the server and retrieve the contents of the article when necessary. Each server, upon receiving the article, also shares the announcement with its other peers. In this manner, the article's existence is eventually flooded to all news servers.

When a reader wishes to read a newsgroup, he requests a list of new articles from his local news server. As is done in Usenet, the server responds with a summary of articles that it has accumulated from its peers. This summary is used by readers to construct a view of the newsgroup. When the client requests an article body, the server sends a `get` request to the DHT for the content hash associated with the article. Once the server obtains the article data it can be returned to the reader. As with posting, the reader is not aware that the news server has been modified.

We expect that servers will locally cache recently requested documents to avoid making multiple requests to the DHT for the same article. If servers cache locally, caching inside the DHT will be unnecessary. The main benefit of local caching is to reduce load on other servers in the system — if servers cache locally, no server is likely to experience more than n remote read requests for the DHT blocks corresponding to a given article.

The primary drawback of this design is a loss of control by individual administrators over what data (articles) are stored on and transit their machines. In a DHT-based storage system, article data will be evenly spread across all participating nodes. Similarly, all nodes must participate in processing DHT lookups for articles, even those in groups that they do not wish to carry. This sacrifice is inherent to the current design.

From a resource standpoint, this drawback is probably not critical: each server in UsenetDHT need only provide a fraction of the space required to store a feed. Even if a traditional server only served a fraction of newsgroups, its storage requirements are not likely to rise if it participates in UsenetDHT. In addition, the use of virtual servers can scale the storage requirements of each site to match capacity.

More troubling are issues relating to content filtration. For instance, some servers might not want to store any data associated with `alt.binaries.pictures.erotica` and normally would not receive those articles from their upstream server. Under this design, a fraction of the articles from these groups will be randomly distributed to servers that would have refused them in the current Usenet. Ongoing work suggests that it might be possible to organize subsets of a single Chord ring such that a lookup could be targeted at an item's successor in a given subset [10]. With this modification it might be possible to form a "no erotic binaries" sub-ring that would not store or participate in routing messages associated with erotic groups. We have not fully investigated this possible solution.

	Bytes Transferred	Storage
Usenet	2 TB	1 TB
UsenetDHT	$2r$ TB	$2/n$ TB

Table 1: UsenetDHT reduces the bandwidth and storage requirements of hosting a Usenet feed in proportion to the fraction of articles read (r) and the number of servers in the network (n), respectively. This table compares the current transfer and storage requirements per day for a full Usenet feed in both systems.

4 Evaluation

In this section, we attempt to quantify the impact of UsenetDHT on the bandwidth and storage requirements of a participating server. In addition, we describe what performance we require from the underlying DHT.

We parameterize our analysis based on a simplistic model of the input rate to the system and estimated readership. Suppose that there are n servers in the system. Let a represent the number of articles injected into the system per second. Correspondingly, let b represent the number of bytes injected per second. Today $a \approx 50$ articles per second and $b \approx 11$ MB per second. Because the bandwidth savings of UsenetDHT depend on readership patterns (namely the fraction of articles which are read), we perform this evaluation as a function of this fraction, r. For the purposes of this model, we will treat these as constants, though of course, they will fluctuate and grow over time.

4.1 Storage

UsenetDHT reduces system-wide storage by storing articles once in the DHT instead of copying them to each site. We will also assume that we have a low-overhead DHT that maintains and replicates data with a replication factor of 2, using some sort of erasure coding to ensure high availability. This means that the system receives $2b$ bytes of new article data each second. This load is spread out over all n servers in the system instead of being replicated at all hosts, resulting in an overall per-host load that is $2/n$ times the load of traditional Usenet. If the number of servers participating in UsenetDHT increases and the number of articles posted remains constant, each server must bear less load. This difference makes it feasible for UsenetDHT to increase the retention of the system.

There is also a small incremental cost required for local indexing. Suppose that each article requires about 512 bytes to store the overview data, which includes article author, subject, message-id, date, and references headers. This data adds an additional $512 \cdot 86400a$ bytes per day to the cost of supporting a full feed. Today, this corresponds to approximately 2 GB daily.

4.2 Bandwidth

UsenetDHT servers save bandwidth by not downloading articles that are never going to be read at their site. However, it is hard to quantify these savings exactly because few studies have been done to measure how many articles are actually read on different servers. In 1998, Saito *et al.* observed that roughly 36% of all incoming articles were read on the server at Compaq [12]. An analysis of logs produced by MIT's news server in August shows that between 27% and 70% of articles received on the server were read each day. However, this fraction would likely be smaller if MIT received a full news feed. If the average actual fraction is r, then proportionally, each server will require roughly rb bytes per second of downstream bandwidth. In addition, each node is required to receive the header and DHT information for each article, which corresponds to $512a$ bytes per second. This may need to be multiplied by a small factor, corresponding to the overhead of communicating with multiple peers.

Similarly, the upstream bandwidth for serving reads is at least rb. The downstream load of a single server will be spread out over all n servers, but in aggregate, each server must have at least rb bytes per second of bandwidth to satisfy DHT read requests from other servers. Additionally, each site must have sufficient upstream bandwidth to inject locally-authored articles into the system.

This analysis ignores the effects of caching at servers. If servers do cache, the same analysis applies if r is the fraction of unique articles that are read.

4.3 DHT performance

The DHT must provide sufficient read and write performance to support the aggregate load on the system. Because the news servers themselves will form the DHT, we expect that the membership of the DHT will be stable. A stable membership makes it easier for the DHT to provide highly reliable storage [2]. Recent DHTs [5] have demonstrated performance scaling properties that suggest they can meet the demands of UsenetDHT.

The read latency and throughput of the DHT will be visible to end-users when they request an uncached article. Although many DHTs routes requests to nearby nodes, read operations on the DHT will certainly have a higher latency than direct requests to a local server. We believe that the bandwidth and storage savings of UsenetDHT are worth this delay. Further, this delay can be partially masked by prefetching articles in a newsgroup when the indexing information is requested but before they are actually read.

4.4 Status

We have implemented a prototype UsenetDHT server that accepts a small fraction of a news feed, inserts articles into the DHash DHT [4], and makes them available to clients. To better understand the dynamics of handling a large feed, we hope to deploy a collection of UsenetDHT servers on the PlanetLab testbed and make a full feed available as a service.

5 Related work

There have been some other proposals to reduce the resource consumption of Usenet. Newscaster [3] examined using IP multicast to transfer news articles to many Usenet servers at once. Each news article only has to travel over backbone links once, as long as no retransmissions are needed. In addition, news propagation times are reduced. However, Newscaster still requires that each Usenet server maintain its own local replica of all the newsgroups. Also, each news server still consumes the full news feed bandwidth across the network links closest to the server.

DNews [11] is a high-performance server that can operate in a mode where it learns what groups are read by the local user population and only fetches articles in groups that are read. Thus, so long as the upstream server has a full feed, this server can emulate the view of a full feed. This feature constrains local bandwidth requirements to be proportional to readership as well as more optimally using local storage.

6 Conclusion and future work

UsenetDHT has the potential to reduce the bandwidth and storage requirements of providing access to Usenet by storing article data in a DHT. UsenetDHT's bandwidth savings depend heavily on difficult-to-predict readership patterns, but provides storage savings that scale with the number of participants. A planned deployment on PlanetLab will help to quantify the benefits of UsenetDHT.

References

[1] BARBER, S. Common NNTP extensions. RFC 2980, Network Working Group, Oct. 2000.

[2] BLAKE, C., AND RODRIGUES, R. High availability, scalable storage, dynamic peer networks: Pick two. In *9th Workshop on Hot Topics in Operating Systems* (May 2003).

[3] BORMANN, C. The newscaster experiment - distributing usenet news via many-to-more multicast.

[4] DABEK, F., KAASHOEK, M. F., KARGER, D., MORRIS, R., AND STOICA, I. Wide-area cooperative storage with CFS. In *18th ACM Symposium on Operating Systems Principles (SOSP '01)* (Oct. 2001). `http://www.pdos.lcs.mit.edu/chord/`.

[5] DABEK, F., SIT, E., LI, J., ROBERTSON, J., KAASHOEK, M. F., AND MORRIS, R. Designing a dht for low latency and high throughput. In *submission process* (2003).

[6] GANGER, G. R., AND KAASHOEK, M. F. Embedded inodes and explicit grouping: exploiting disk bandwidth for small files. In *Proceedings of the USENIX 1997 Annual Technical Conference* (Anaheim, California, January 1997), pp. 1–17.

[7] GRADWELL.COM. Diablo statistics for news-peer.gradwell.net. `http://news-peer.gradwell.net/`. Accessed 6 November 2003.

[8] GRIMM, B. Diablo statistics for newsfeed.wirehub.nl (all feeders). `http://informatie.wirehub.net/news/allfeeders/`. Accessed 6 November 2003.

[9] KANTOR, B., AND LAPSLEY, P. Network news transfer protocol. RFC 977, Network Working Group, Feb. 1986.

[10] KARGER, D. Private communication, Nov. 2003.

[11] NETWIN. Dnews: Unix/windows usenet news server software. `http://netwinsite.com/dnews.htm`. Accessed 9 November 2003.

[12] SAITO, Y., MOGUL, J. C., AND VERGHESE, B. A usenet performance study. `http://www.research.digital.com/wrl/projects/newsbench/usenet.ps`, Nov. 1998.

[13] STOICA, I., MORRIS, R., KARGER, D., KAASHOEK, M. F., AND BALAKRISHNAN, H. Chord: A scalable peer-to-peer lookup service for Internet applications. In *ACM SIGCOMM* (San Diego, Aug. 2001). An extended version appears in ACM/IEEE Trans. on Networking.