UsenetDHT: A Low Overhead Usenet Server

Emil Sit, Frank Dabek and James Robertson
MIT Computer Science and Artificial Intelligence Laboratory
{sit, fdabek, jsr}@csail.mit.edu

Abstract

UsenetDHT is a system that reduces the storage and bandwidth resources required to run a Usenet server by spreading the burden of data storage across participants. UsenetDHT distributes data using a distributed hash table. The amount of data that must be stored on each node participating in UsenetDHT scales inversely with the number of participating nodes. Each node's bandwidth requirements are proportional to the fraction of articles read rather than to the total number posted.

1 Introduction

Usenet is a large, distributed, messaging service that serves thousands of sites world wide. Since its introduction in 1981, the Usenet has been growing exponentially. The daily size of Usenet postings doubles approximately every ten months. In early 2004, users created approximately 1.4 TB of new Usenet data, corresponding to about 4 million articles, 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 is more than a 100 Mbps connection can support. To provide many days of articles for readers becomes an expensive proposition.

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 (DHT) that stores all article data. This approach obviates the need for articles to be replicated to all participating servers. Instead of storing article data at each server, UsenetDHT stores article data on a small number of nodes in the DHT. The DHT distributes articles geographically, and can locate them quickly. It ensures high availability by re-replicating as necessary, maintaining the benfits of full replication without incurring the bandwidth and storage costs.

Using this approach, for an n-host system, each host will be able to offer a full Usenet feed while only storing $2/n$ as much data as it would have had to using a traditional system. Instead of using local storage to hold a large number of articles for a short period of time, UsenetDHT allows each server to hold fewer articles but retain them for longer periods of time.

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 current size Usenet and known readership patterns, we expect this optimization to translate into a significant reduction in the bandwidth required to host a full feed.

2 Background

2.1 Usenet

Usenet is a distributed mesh of servers that are connected in a mostly ad-hoc topology that has evolved since its creation in 1981. Servers are distributed world-wide 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 [10] and RFC2980 [1] describes the NetNews Transfer Protocol in more detail.

Articles are organized into a hierarchy of newsgroups. Upon receiving each article, each peer determines which newsgroups the article is in (based on meta-data included in the article) and updates an index for the group. The group indices are sent to news reading software and is

This research was conducted as part of the IRIS project `http://project-iris.net/`, supported by the National Science Foundation under Cooperative Agreement No. ANI-0225660.

used to summarize and organize displays of messages.

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. Control messages can create or remove newsgroups, and cancel news postings (i.e. remove them from local indices 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 [13] 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" — 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 multi-media files. In 2001, there was an average of 300 GB of "binary" articles posted per day, but the volume in early 2004 is already approaching 2 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 400,000 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 [16] or Tapestry [17]) 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 strategy also allows DHTs to optimize reads by using nearby servers.

3 Architecture

In this section we present the high-level design of UsenetDHT, trace through the life of an article after it is posted, and discuss some tradeoffs of our design.

3.1 Design

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 for articles — while mostly preserving the major features of Usenet. UsenetDHT accomplishes this goal 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.

All indexing of articles remains local to servers. In order to support this, UsenetDHT nodes exchange meta-data and control-data using the same flood-fill network as Usenet. In essence, the article bodies are elided from the news feed between servers. This approach allows cancellation and moderation of articles to work much as they do now; for example, cancellation simply updates the local group index to exclude the canceled article and no DHT storage needs to be touched. 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.

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

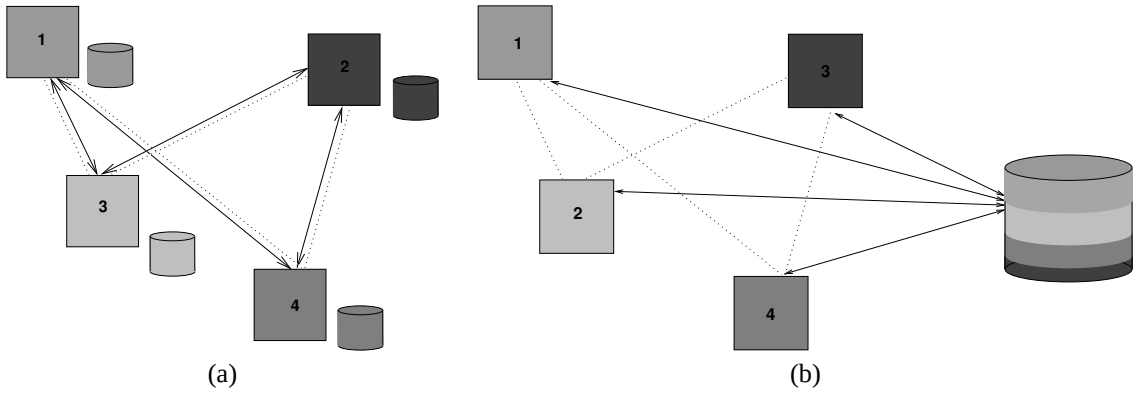

Figure 1: The design of Usenet (a) and UsenetDHT (b). Usenet servers are connected to form a mesh network and exchange article data (solid lines) and meta-data such as article announcements and cancellations (dotted lines). Each Usenet server provides enough storage for an entire feed. UsenetDHT servers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

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. Each server will need to determine an appropriate cache size that will allow them to serve their readership efficiently.

3.2 Article lifetime

Consider how an article 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 verify the integrity of data received over the network.

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.

3.3 Tradeoffs

The primary drawback of UsenetDHT 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 design as presented above.

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, sites can run virtual nodes [4] to match DHT storage requirements with available capacity: each virtual node allows a physical node to claim responsibility for an additional fraction of the blocks stored in the DHT.

Article expiration must also be handled on a system-

wide level. Blocks associated with articles can only be removed when the article has expired from the indices of all servers. This global parameter is set out-of-band.

More troubling are issues relating to content filtration. One common policy that UsenetDHT would like to support is the ability for servers to decide not to carry particular newsgroups, such as erotic newsgroups. Normally, such servers would request not to receive articles associated with unwanted groups from their upstream server. In UsenetDHT, a fraction of the articles from *all* groups will be randomly distributed to all servers. One possible solution may be to make use of recent work that enables a Chord ring to be organized into subrings such that a lookup could be targeted at an item's successor in a given subset of nodes [11]. Subrings could be used so that storage for content such as adult material would be limited to a single subring: lookups for articles in adult groups would be confined within the subring. We have not fully investigated this solution, but plan to do so in the future.

A different challenge is effectively filtering spam; the previous solution would not work since spam appears in all newsgroups. In total, the use of UsenetDHT should reduce the storage impact of spam since each spam message will only be stored once. However, administrators will still want to reduce the presence of spam in newsgroup indices — naively, any node wishing to do this would have to retrieve every complete article to determine if it is spam. This approach would lose all the bandwidth benefits of UsenetDHT. Fortunately, existing techniques such as spam cancellation can still work in UsenetDHT — a small number of sites determine what messages are spam (or are otherwise undesirable) and publish cancel messages. Other sites process the cancel messages to filter out spam locally. Sites would continue to filter local posts to prevent spam from originating locally as well.

4 Evaluation

We have implemented a prototype UsenetDHT server that can accept small news feeds, insert articles into the DHash DHT [4], and make them available to clients. However, the system has yet to be tested with a full feed. In this section, we quantify the potential bandwidth and storage requirements of a UsenetDHT server compared to a Usenet server based on available statistics about the current Usenet. We also describe what performance we require from the underlying DHT. We hope that a future deployment of our implementation running on the PlanetLab test bed will verify these calculations experimentally.

	Total Bytes Transferred	**Storage**
Usenet	$2\overline{b}$	$\overline{b}$
UsenetDHT	$2\overline{b}r + 2 \cdot 512\overline{a}$	$2\overline{b}/n + 512\overline{a}$

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, where $\overline{b}$ represents the total number of bytes for articles injected each day. A single peer is assumed.

4.1 Analysis model

The resource usage of a UsenetDHT server depends on the behavior of readers; we parameterize our analysis based on a simplistic model of the input rate to the system and estimated readership. Let n be the number of servers in the system. Let a represent the average number of articles injected into the system per second. Correspondingly, let b represent the average number of bytes injected per second. For example, based on recent statistics from a well-connected news server (`newsfeed.wirehub.nl`), $a \approx 77$ articles per second and $b \approx 21$ MB per second. Statistics from other large news servers are similar. When we examine the storage requirements of a UsenetDHT server it will be convenient to consider data stored per day: we will write $\overline{a}$ for the total number of articles injected on average each day ($\overline{a} = 86400a$) and $\overline{b}$ for total bytes injected per day.

To model the readership patterns of Usenet, we introduce r, the average percentage of unique articles read per site. Unfortunately, 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 [15]. Today, because of large traffic volume, many networks outsource their news service to large, well-connected providers, such as GigaNews. Byte-transfer statistics for the month of January 2004 from a small ISP that outsources news service to GigaNews suggest that the ISP's approximately 5000 customers read approximately 1% of the total monthly news. In fact, the trends from that ISP show that the number of bytes downloaded has remained relatively constant around 300 GB per month over the past year. This may mean that r will decrease over time if the growth of Usenet remains exponential.

For the purposes of this analysis, we will treat these parameters as constants, though of course, they will change over time.

4.2 Storage

UsenetDHT reduces system-wide storage by storing articles once in the DHT instead of copying them to each site. The storage requirements of a node are proportional to the amount of data in the system and the replication overhead of the underlying DHT and inversely proportional to the number of participating servers.

We assume that UsenetDHT is based on 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 assumption 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. Because each server must dedicate a factor of n less storage, UsenetDHT should allow articles to be retained longer within 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 $512\overline{a}$ bytes per day to the cost of supporting a full feed. In early 2004, this corresponds to approximately 3.2 GB daily, which is barely 0.1% of the total data stored daily.

Sites must also provide some storage space for caching locally read articles. The sizing and effectiveness of this cache is dependent on the size of the reader population and the diversity of articles that they retrieve.

The total daily storage requirement for a UsenetDHT server is $2\overline{b}/n + C + 512\overline{a}$ where C is the size of the server's article cache. This differs from the cost of a traditional Usenet server by roughly a factor of $2/n$. A Usenet server requires $\overline{b} + 512\overline{a}$ bytes of storage daily.

4.3 Bandwidth

UsenetDHT servers save bandwidth by not downloading articles that are never going to be read at their site. Unlike Usenet, the bandwidth used by a UsenetDHT server is proportional to readership, not to the volume of the feed the server carries. In this evaluation, we are only interested in wide-area data transfer. When comparing UsenetDHT's bandwidth requirements to Usenet, we will not consider the cost of sending articles to readers for Usenet servers since these readers are likely to be on the same network as the server.

Servers must download articles from the DHT to satisfy requests by local readers. If the average actual fraction of read articles is r, 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 header overhead may need to be multiplied by a small factor, corresponding to the overhead of communicating with multiple peers.

A server must also send data it stores in its role as part of the DHT. As we showed above, each server requests rb bytes per second from the DHT. This load will be spread out over all n servers. Correspondingly, each participating server must send rb/n bytes per second to satisfy read requests from each other server in the system. Thus, in aggregate, each site must have rb bytes per second of upstream bandwidth. Additionally, each site must have sufficient upstream bandwidth to inject locally-authored articles into the system.

A Usenet server's wide-area bandwidth requirements are equal simply to the size of the feed it serves. In our notation, a Usenet server is required to read b bytes per second from the network.

4.4 DHT performance

The DHT used by UsenetDHT 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 (more stable than, for instance, the membership of the Gnutella network). A stable membership makes it easier for the DHT to provide highly reliable storage [2].

Recent DHTs have demonstrated performance scaling properties that suggest they can meet the demands of UsenetDHT [5]. An improved version of the DHash DHT can locate and download an 8K block in approximately 100ms when run on the PlanetLab test bed [14]. The read latency of the DHT will be visible to end-users when they request an uncached article: reading a small text article requires the DHT to perform a single block download. The time required by the DHT to perform the fetch is likely to be significantly higher than the time to perform a similar operation on a local server. This delay can be partially masked by pre-fetching articles in a newsgroup when the indexing information is requested but before they are actually read. We believe that the bandwidth and storage savings of UsenetDHT are worth the delay imposed by the DHT.

The throughput requirement for a UsenetDHT node is rb. Thus, a full-feed deployment with $r = 0.01$ in early 2004 would require that each DHT node provide 200 KB/s of symmetric bandwidth. A node participating in the

DHash DHT and located at MIT was able to read data at 1 MB/s from the DHT. Such performance should be sufficient to support a full feed for the near future. An experimental deployment of the system will allow us to verify that DHash is able to meet the performance requirements of the system.

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.

NewsCache [9] reduces resource consumption by caching news articles. It is designed to replace traditional Usenet servers that are leaf nodes in the news feed graph, allowing them to only retrieve and store articles that are requested by readers. In addition to filling the cache on demand, it can also pre-fetch articles for certain newsgroups. These features are also available as a mode of DNews [12], a commercial high-performance server, which adds the ability to dynamically determine the groups to pre-fetch. Thus, both NewsCache and DNews reduce local bandwidth requirements to be proportional to readership as well as more optimally using local storage. However, they only do this for servers that are leaf nodes and does not help reduce the requirements on upstream servers. By comparison, UsenetDHT also reduces resource requirements for those servers that are not leaf nodes. Also, standard caches will continue to work with UsenetDHT servers.

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 readership patterns, but UsenetDHT provides storage savings that scale with the number of participants. A planned deployment on PlanetLab will help to quantify the benefits of UsenetDHT.

Acknowledgments

We would like to thank Richard Clayton for his assistance in gathering statistics about Usenet's recent growth and status, and the anonymous reviewers and the members of the PDOS research group for their helpful comments.

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 *Proc. of the 9th Workshop on Hot Topics in Operating Systems* (May 2003).

[3] BORMANN, C. The Newscaster experiment: Distributing Usenet news via many-to-more multicast. `http://citeseer.nj.nec.com/251970.html`.

[4] DABEK, F., KAASHOEK, M. F., KARGER, D., MORRIS, R., AND STOICA, I. Wide-area cooperative storage with CFS. In *Proc. of the 18th ACM Symposium on Operating Systems Principles* (Oct. 2001).

[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 *Proc. of the 1st Symposium on Networked System Design and Implementation (to appear)* (Mar. 2003).

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

[7] GRADWELL.COM. Diablo statistics for news-peer.gradwell.net. `http://news-peer.gradwell.net/`. Accessed 12 February 2004.

[8] GRIMM, B. Diablo statistics for newsfeed.wirehub.nl (all feeders). `http://informatie.wirehub.net/news/allfeeders/`. Accessed 12 February 2004.

[9] GSCHWIND, T., AND HAUSWIRTH, M. NewsCache: A high-performance cache implementation for Usenet news. In *Proc. of the 1999 USENIX Annual Technical Conference* (June 1999), pp. 213–224.

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

[11] KARGER, D., AND RUHL, M. An augmented Chord protocol supporting heterogeneous subgroup formation in peer-to-peer networks. In *Proc. of the 3rd International Workshop on Peer-to-Peer Systems* (Feb. 2004).

[12] NETWIN. DNews: Unix/Windows Usenet news server software. `http://netwinsite.com/dnews.htm`. Accessed 9 November 2003.

[13] NIXON, J., AND D'ITRI, M. Cleanfeed: Spam filter for Usenet news servers. `http://www.exit109.com/~jeremy/news/cleanfeed/`. Accessed on 15 February 2004.

[14] PlanetLab. `http://www.planet-lab.org`.

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

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

[17] ZHAO, B. Y., HUANG, L., STRIBLING, J., RHEA, S. C., JOSEPH, A. D., AND KUBIATOWICZ, J. D. Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications 22*, 1 (Jan. 2004).