MotusNet: A Content Network

H. T. Kung, Pai-Hsiang Hsiao, Adon Hwang, Conrad Nobili, Michael Vernal, Dario Vlah, Salimah Addetia, Cy Chan, Arvin Chang, David Chen, James Chen, Chen-Mou Cheng, Chris Cho, Matthew Clark, Anthony Delvecchio, Alexandra Fedorova, Mohammed Hakky, Nithin Iyengar, Bryan Kim, Denise Kim, Khawla Konyna, Molly Krause, Jason LeeKeenan, Kitty Leung, Hsin-Ting Liu, Qing Liu, Josiah Madigan, Georgi Matev, Jay Moorthi, Jared Morgenstern, Dan Myung, Adriana Rincon, Eben Scanlon, Adam Scheuer, Marcin Strojwas, Octavian Timaru, Dafina Toncheva, Eric Tulla, David Tuttle, Ami Vora, and Zhiyu Yang

Division of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138, U.S.A.

Abstract – MotusNet is a content network in which the addressable objects, contents, are decoupled from the physical nodes that carry them. The content names in MotusNet are hierarchical, so that they can be aggregated in the routing tables to support a larger number of contents. Further, the process of content replication improves availability, performance, and defends against denial of service attacks by localizing their effect. Additionally, using content migration, similar contents can be placed closer together in order to improve the degree of aggregation. A networking class at Harvard University has built a prototype of MotusNet.

1 Introduction

A content network is a network in which the addressing and routing of content is based on content description rather than location. Thus, in a content network, content is free to move around and be replicated while it remains accessible. In addition, since content location is arbitrary and need not necessarily be associated with a particular host, content can be anonymous. The content network is based upon the assumption that in the future, spacious servers and high bandwidth connections between nodes on the network will be commonplace, so that frequent content replication and migration are feasible. A content network is a form of peer-to-peer computing, as nodes along the network communicate with their peers using peer or content identities, not their locations [17] [18].

The motivation for building a content network stems from multiple sources. IP networks, while they have been extremely useful, as seen by the tremendous growth of the Internet in the past few years, have certain functional limitations that are not easily fixed. The content network is designed to address some of the limitations found in current IP networks. For instance, the lack of location addresses on a content network means less vulnerability to denial of service attacks. The content network is also designed to be capable of easily supporting new types of client applications that require flexible addressing or no addressing at all, such as peer-to-peer or mobile computing. In addition, content networks are designed to better utilize the impending increase in bandwidth and large-scale storage capacity suggested, by current hardware developments, to make content more available, to deliver it more quickly, and distribute it more efficiently. The content network is thus designed to simplify and reduce the network's administrative overhead currently required to implement certain types of features such as high availability and defense from attacks, and making content available across administrative domains.

Section 2 lists the goals and constraints of the project. Section 3 describes the architecture of our content network. Section 4 provides some of the highlights of the implementation, starting from the bottom of the protocol stack and working up. Section 5 provides an assessment of the systems implemented and suggests some topics for future work. Section 6 gives a quantitative analysis of some aspects of network performance. Section 7 lists work related to our content network implementation.

2 Project Requirements, Goals, and Constraints

The content network we describe above was designed and implemented by the course staff and the students of the CS 244 class at Harvard University, during the spring semester of 2001. The 36 students in the class (mainly Computer Science undergraduates) were divided into 14 groups of two or three people each. These groups worked to create a testbed platform with which the basic ideas and viability of a content network could be explored. Each group was given responsibility for a component of the content network project. The course instructor and teaching fellows (mostly Computer Science graduate students) defined objectives, supervised the work, conducted weekly meetings with groups, and coordinated dependencies between the various groups' components. The course staff was also responsible for integrating the implemented components into a functional system. The work was split into two parts, Phase I and Phase II,

with the first phase concerned with building a functional content network testbed, and the second phase designed to extend network capabilities in directions deemed interesting by the course staff and students.

The requirements of the project were to build a working content network system; to address design challenges such as header layout and routing methods, understand the design tradeoffs that need to be made when building a large system supporting massive amounts of contents, and to use Internet content tools such as Web browsers. Beyond these requirements, the project organization had the following educational goals. First, the project was intended to expose students with varying levels of software engineering experience to the process of developing a reasonably complex software system, from initial conception, through specification, detailed design, testing, integration, and documentation. Second, the division of tasks among the groups was targeted at allowing components to be developed in parallel. This meant that the students gained valuable experience at not only working in groups, but also in coordinating their own groups with those of others.

One constraint to the project was the strict one-semester timetable. Within this time period, we were able to cover relevant literature, architectural designs and projects, in addition to working on our own design and implementation.

3 Content Network Architecture

There are three basic types of nodes in the content network: a content source, which has some type of content it makes available to the rest of the network; a content requestor, which is looking for a piece of content; and a router, which passes messages between the previous two types of nodes as well as other routers. These types of nodes can be co-located on a single physical host. All messages are exchanged without requiring either party to have a network address.

Instead of flat addresses, the content network uses hierarchical descriptors, consisting of sequences of increasingly general categories. For example, a content about a Red Sox baseball game may be described by the sequence (“Today’s Game”, “Red Sox”, “Baseball”, “Sports”).

Traffic across the network consists of three types of messages: registrations (REG), requests (REQ), and deliveries (DLV), as depicted in Figure 1. When a content source wishes to advertise some content, it sends an REG message containing the content descriptor for that content. Similarly, a content requestor sends out

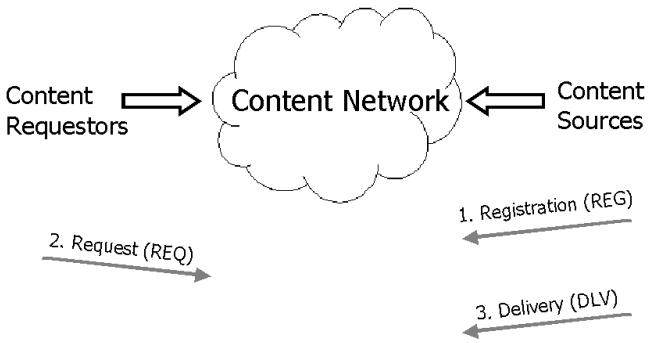

Figure 1. REG, REQ, DLV messages in the Content Network.

an REQ to request a content with a particular content descriptor. The routers then forward this REQ toward the closest content source with the requested content. The REQ may however be dropped if it reaches a router that cannot forward the message, or its Hops-to-live field decreases to zero. When a content source receives a REQ for content it provides, it responds with a DLV. This DLV contains the requested content and is forwarded back through the network to the requestor using a trail set up by the request.

The Phase I design is content over IP, while Phase II is content over both IP, which is more general, and directly over Ethernet, which is more efficient (see Figure 2). In order to simplify the implementation of the protocol stack, for the Phase II design we emulated connectivity among nodes at a *connectivity server* made to switch all traffic. The connectivity server proved to be a convenient way to simulate large, arbitrary network topologies with hundreds of nodes and also to gather traffic statistics, such as average hop count for content access.

4 Design Highlights

In this section, we examine the major components of the content network. We discuss the functionality of each component, along with interesting design and implementation decisions we have made.

4.1 Topic Directory

A network whose nodes are denoted by flat addresses does not scale well if routing tables are to list

	Internet	Content/IP	Content/Ethernet
Network Layer	IP	Content	Content
Link Layer	MAC	IP	MAC

Figure 2. Protocol stacks for the Internet, content over IP, and content over Ethernet. Corresponding layers are shown in the same row.

every available node. Traditional IP networks take advantage of hierarchy in host addresses by aggregating routing table entries, thus avoiding enumeration of every Internet host [21]. Analogously, hierarchically named contents can let content networks scale to a large degree.

In order to provide a large test set of hierarchically named contents, we constructed a topic directory (see Figure 3) mapping URLs of existing Internet content to hierarchical content descriptors. We populated the topic directory using part of the taxonomy from the Open Directory Project [16]; however, any content classification would have worked.

4.2 Message Headers

The goal of the content message header (see Figure 4) is to strike a balance between expressiveness and efficiency. The header needs to allow for arbitrarily long content descriptors, which describe any possible content. At the same time, it must also allow routers to route packets quickly, as in IP networks. To achieve this balance, the message header is divided into two parts: a fixed-length section for efficient routing, followed by a variable-length section for storing the content descriptor. This resembles the design of the IPv6 header, which too consists of fixed- and variable-length sections.

The fixed-length section of the message header contains standard fields such as version number, header length, and checksum. Instead of a destination address field, the content message header contains a summarized content descriptor field with cumulatively hashed values of multiple *elements* of the descriptor (see Figure 5). However, only a fixed-size segment of the content descriptor resides in this field at once; the level field denotes the first element of that segment. Using this summary, packets can be forwarded quickly without examining the variable-length section of the header. The manner in which elements are occasionally

Figure 3. (a) A sample topic directory. The leaf, www.racquets.com, is a content. (b) The associated content descriptor, which consists of nodes on the path to the root of the tree.

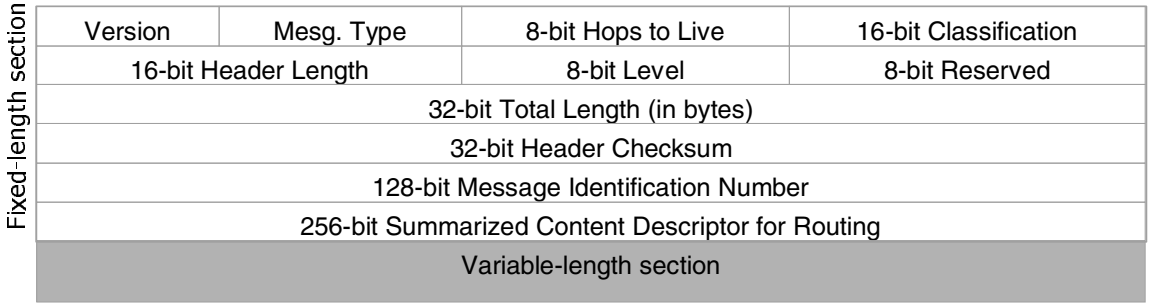

Figure 4. Content Message Header

swapped into the fixed-length section is described in Section 4.3, and resembles header processing for loose source routing [20] in IPv4 networks.

The variable-length section contains the full content descriptor in XML [12]. We chose XML for the encoding because of its inherent extensibility. Using XML to encode the entire content descriptor, the variable-length section of the message header may become quite large. However, routers do not need to parse this section too frequently because the fixed-length part of the header is usually sufficient to make a forwarding decision. To provide additional flexibility in routing, content descriptors encoded in XML could be chained together using Boolean logic operators. Our system does not yet implement this functionality.

4.3 Forwarding

As stated earlier, a unique feature of the content network is that message forwarding is based on content descriptors instead of node addresses. Therefore, a routing table entry consists of a content descriptor prefix and an associated next hop router. The routing table is looked up with the summarized content descriptor field—thus, the lookup has access only to a fixed-size portion of the full content descriptor. The lookup

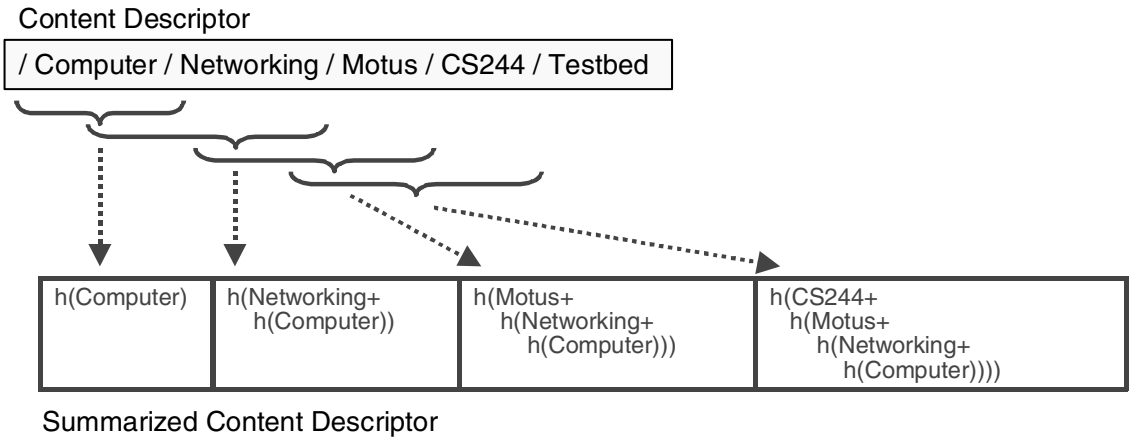

Figure 5. Summarizing the content descriptor by cumulative hashing. h(x) denotes a hash function.

returns a next-hop router if a match is found, and a flag to indicate whether more specific entries exist. In order to guarantee a longest prefix match on the full descriptor, if this flag is turned on, the summary of a more specific portion of the descriptor is computed and the lookup repeated until the returned flag is off. The summary that resulted in the longest match remains in the fixed-length section, and the packet is forwarded to the next hop. Figure 6 describes this process with an example.

In order to facilitate the forwarding of DLV messages, a trail table is maintained. When a router receives an REQ, it records the globally unique 128-bit message ID and the previous hop in the trail table. Thus, when a DLV with the same message ID arrives, it can be forwarded on the correct path to the source. Trail table entries are associated with a time-to-live value, and deleted after they expire.

4.4 Route Aggregation

The routes in our content network are disseminated using a distance vector routing protocol. The protocol takes advantage of content locality by performing aggregation of routing table entries. In many possible applications of a content network, it is likely that similar contents will be close together. Aggregation of routing table entries does not affect routing far away from the content source, and can significantly reduce the number of entries. For example, suppose that content that is specific to Boston, such as the Red Sox web pages, tends to be clustered near Boston. A far away router only needs to store one routing entry for all such Boston content (see Figure 7).

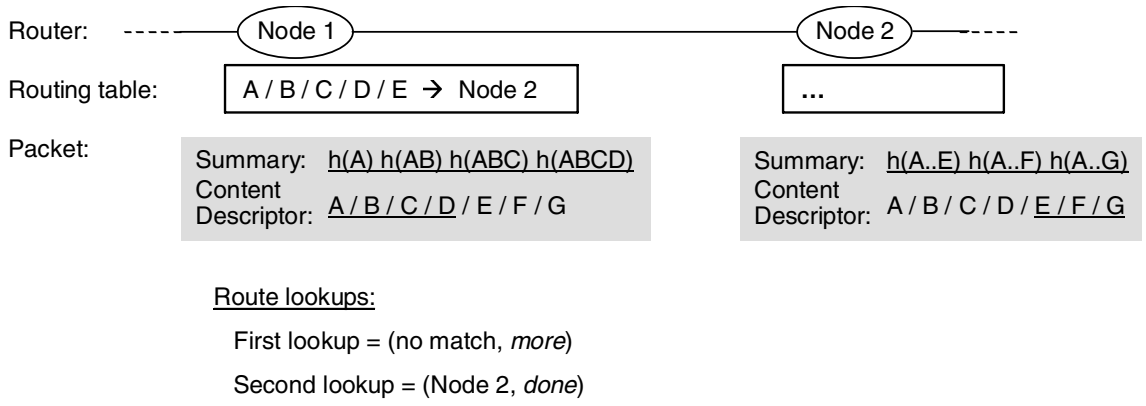

Figure 6. Illustration of incremental lookup. A packet with first four elements of the descriptor in its summary field arrives at Node 1. The first lookup finds no match; however, the result indicates existence of longer entries, so the next three elements are hashed and placed into the summary field. The second lookup now returns the longest prefix match. The resulting packet, with the updated summary field, is shown at Node 2.

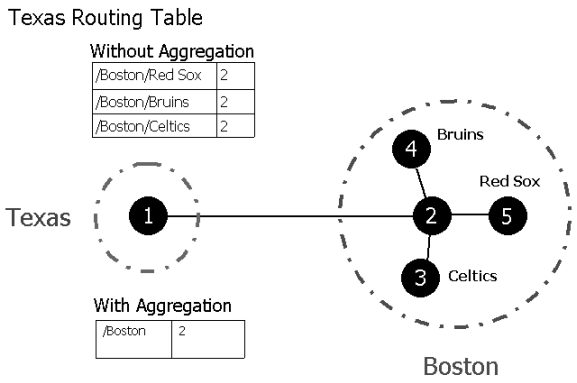

Figure 7. Routing table aggregation

4.5 Transport Layer and API

The transport layer interfaces directly with an API, whose interface is based on transaction identifiers, or sockets, and is analogous to the BSD Socket API. While transport layer design scope usually includes flow control, packetization, retransmission, and congestion control, we chose to implement only packetization for simplicity.

4.6 Applications

We have developed several applications to work on top the content network in order to demonstrate the convenience and power it can provide to application developers. In particular, peer-to-peer applications are easy to create, due to the fact that most of the needed functionality is already provided at the network layer.

A proxy server has been designed to enable interfacing between content on the Internet and our content network. The proxy server translates a request in the form of an ordinary URL, made by a standard Web browser, into a content request. The proxy server caches content internally and uses a simple FIFO scheme to expire cached material.

In order to use content on the Internet, data from the content network returning to a browser is bundled into a content *container*. Since the content network deals with contents and not specific files, the container allows related files to appear as a single content. In a container, the files reside within *tar* archives, each in its own directory, and are accessed through a programming interface that allows reading and writing. For

instance, a container may consist of multiple HTML files, CGI scripts, and image files from a single Web page.

We have an Internet-to-content-network gateway that directly transmits web pages through the content network and a proxy server, to the end user's browser. Given pre-configured URLs from the topic directory, each content-network gateway router REGisters the URLs as available to the content network. They then translate content network REQuests into HTTP requests, acting as WWW proxy servers from the point of view of WWW servers in the Internet. The HTTP replies are packaged into content containers as described above. While pages with frames are obtained by recursively downloading referenced HTML files, parsing JavaScript proved too difficult, so media loaded by JavaScript are ignored.

5 System Assessment

Section 5.1 discusses the basic infrastructure and functionality we implemented in Phase I. Section 5.2 describes the more ambitious and advanced functionality we started in Phase II. Section 5.3 addresses areas we considered but did not implement due to time constraints, and which are planned in future work.

5.1 Phase I Implementation

The basic functionality implemented in Phase I includes content routing, allowing access both to content on the web from the content network, and content in the content network from a normal web browser.

Even with the basic implementation, we were able to see potential advantages over current networks. One of the content network's major features is the fact that its messages are routed by content and not by location. This makes management of content extremely easy. For example, a content provider can easily improve the availability and performance of a particular content by placing multiple copies in the network (see analysis in Section 6). Since the content network already routes by content, no further alteration is necessary to have requests automatically routed to the closest copy. This type of adaptive functionality is also possible on an IP network, but generally requires a great deal of manual management to maintain, whereas the content network is directly suited for this.

In addition to the ease of content management, the content network may allow for easier defense against disruptions such as distributed denial of service attacks. Content can be duplicated and placed in several

locations in the network, and the burden of the attack is diffused among the several copies. When a particular content is requested, the request is routed to the nearest content source automatically. Unlike in an IP network, there is no need for a load-balancing gateway or a complex manual redirection mechanism, both of which requires explicit management and could possibly introduce either new security vulnerabilities that might be exploited or new points of attack.

5.2 Phase II Implementation

In the last few weeks of the class we augmented the basic content network with more advanced functionality. In particular, we began implementing migration and content-clustering based routing.

The content network can be augmented to allow content migration in order to improve performance and network efficiency. There are two possible forms of content migration: request-based and clustering-based. Request-based migration moves content in the direction of high frequency requests to decrease the average latency of content delivery [11]. This type of migration is under the assumption that storage space is limited, so that choosing where to place the single copy of the content becomes important.

We have studied two content placement approaches for reducing the size of routing tables in routers. The first approach places contents based on their paths. That is, contents that have the same initial portions of their paths are placed in the same region so that these contents can be reached via the same external network-layer connections for the region. Consider, for example, Figure 7. Contents related to the sports teams in Boston, such as the Bruins, Celtics and Red Sox, are all placed in the same region so that they can be reached via the same external links connected to node 2. Thus, for nodes outside the region, such as node 1 of Figure 7, to forward content messages related to these teams, their route tables need only have one entry corresponding to the path "Boston/sports teams", rather than separate entries for each team. We have investigated methods of dynamically migrating contents sharing the same initial paths to the same region, in order to reduce the size of routing tables.

The second approach places contents based on their semantics, as described in [11]. Using content clustering analysis, a content search tree is built for a class of contents of interest. By searching over the tree from the root, any given content will end up at one of the tree leaves. The content is then placed at the leaf, or a node known to the leaf. Routing a request for a content follows the same search process over the content

search tree. A node need only have one routing entry for each of the cluster centroids in the layer below. Since a centroid represents all semantically near contents under some metric, a node will just need to track a small number of them. For example, for the content cluster of Figure 8, each node needs to track no more than two centroids. Thus its routing table has no more than two entries.

The basic mechanisms of our content network, such as the content protocol stack, can support both approaches. Since nodes of the current implementation forward content messages based on content paths, it supports the first approach naturally. To support the second approach, routing tables need to be loaded with the content search tree, and the route lookup will involve content classification.

5.3 Future Work

In addition to the issues explored and covered in our two phases of implementation, there are a variety of issues that we have not yet addressed sufficiently due to time constraints, and still need to be dealt with in the future. In particular, there are several important security concerns to consider in the content network.

For instance, while content-based addressing is convenient because individual content sources cannot be easily targeted for attack, there is also a potential downfall. The anonymity of the network can be exploited if a content source spoofs content, since this false content cannot be bypassed easily in the network, even when detected. Therefore, some type of authentication scheme should be developed and added to system, preferably one that does not have a single point of failure.

In addition, malicious sources can flood routing tables with invalid entries. Likewise, content requestors can overload the system by sending large amounts of requests. Deregistration of content and privatization of routing tables [22] could be possible solution to these problems, but have yet to be explored in depth. It may

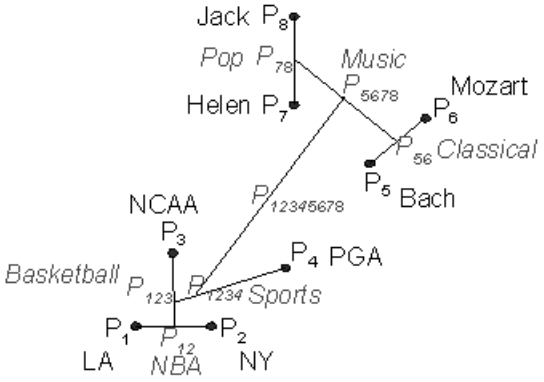

Figure 8. Content clustering

be possible to build a trust management system into the network to help control some of these security issues.

6 System Analysis

Two simulations were conducted to analyze the availability of content on a content network and the benefits of migrations. These simulations were performed separately from our content network testbed.

The content network makes "mirroring" content very easy. Mirroring increases content *availability*—the fraction of nodes, which can reach any copy of a given content. The purpose of the first simulation was to determine the availability of content in a mobile ad hoc network. Multiple copies of the same content were placed randomly in a 1000m x 1000m field with *n* nodes, each with a radio range of 250 m. As expected, availability improved as either the number of copies of the content, or the node density increased (see Figure 9). For example, in a 20-node topology with a single copy of the content in our network, approximately 50% of the nodes have access to the content, whereas with 15 copies of the content, approximately 94% have access to the particular piece of content. In a somewhat denser network of 35 nodes, just two copies of the content were sufficient to provide 95% of the nodes with access to the content network.

The second simulation evaluates the benefits of migration with a simple network of *n* nodes, arranged in a ring topology. We assume each node has space to store two pieces of content—one to store its original copy, and one as temporary space for contents being migrated. The ring topology allows all migration to move in the same direction to avoid deadlock. In addition to being content sources, nodes act as requestors of content. The content requests are modeled using a Zipf distribution, which has been shown to model the

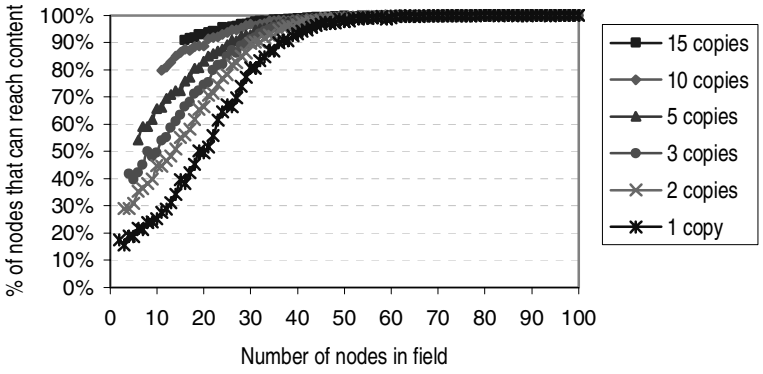

Figure 9. Availability of contents vs. node density

frequency of requests to web pages and other contents [2] [5].

The Zipf distribution is a *heavy-tailed* distribution, where even as items become less popular, they still occur with significant probability. Therefore, we assume that caches with small, fixed storage space end up thrashing—in such a system migration becomes a better alternative. The more negative the slope of the Zipf distribution, the less heavy the tail—which means that the requests exhibit more locality (see Figure 10).

We compared the number of hops to deliver the content from random locations versus from optimal locations. As expected, when contents are placed randomly, the results are not affected by the request distribution (see Figure 11). In addition, the number of hops increases as the number of nodes increases. However, the number of hops in optimal placement is lower for requests exhibiting greater locality (see Figure 12). For distributions with extremely high locality such as the Zipf distribution with a slope of –3, the average number of hops is nearly independent of the size of the network topology.

7 Related Work

Gritter and Cheriton [8] recently proposed an architecture for content routing support in the Internet, where the key idea is to *route* DNS requests, instead of performing recursive lookups using the current out-of-band DNS hierarchy. To do this, they propose a domain name lookup protocol to run alongside IP routing protocols. In order to allow scalability, content names are bundled into volumes called *aggregates*. The membership of aggregates is determined by administrative relationships, such as when all content providers, belonging to a certain ISP, advertise their content in the same aggregate. The idea of routing on content names is similar to what is presented in this paper; here, our key distinction is the hierarchical naming for contents, which lets arbitrary levels of aggregation happen automatically. Beyond that, we discuss content

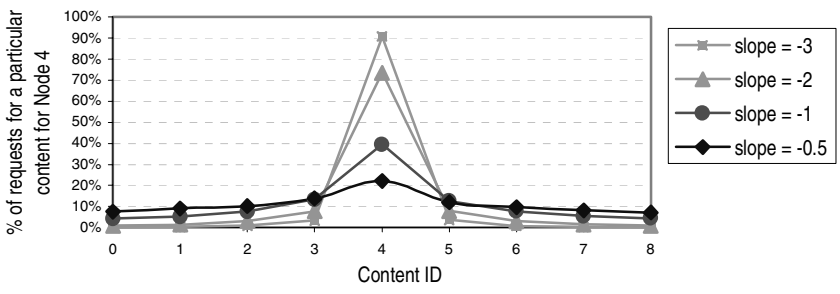

Figure 10. Request Model Based on Zipf Distribution (Node 4 in 9-node network)

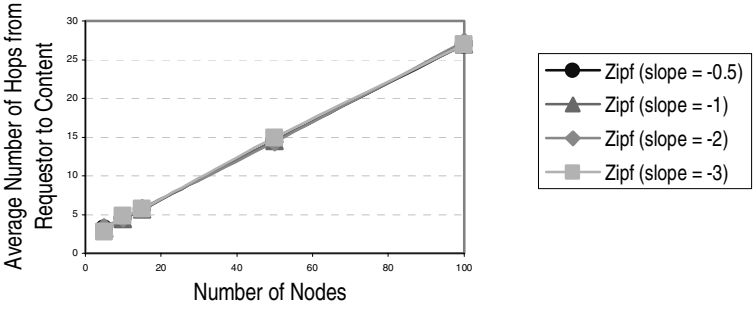

Figure 11. Latency of content requests (random placement)

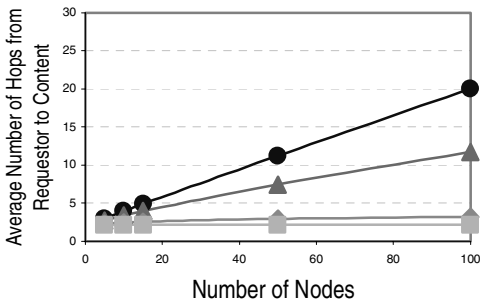

Figure 12. Latency of content requests (optimal placement)

migration and clustering techniques, enabled by the contents' independence of location.

In general, P2P systems decouple content from location, emphasize anonymity, only grant information to nodes about their peers, and allow duplication of content or some other form of redundancy to increase content availability. The Gnutella [6] and Freenet [4] are examples of projects in peer-to-peer computing. Freenet maintains routing tables of hashed content-identifiers—an approach that is functionally similar to our hash-based content routing. Gnutella, on the other hand, uses an application broadcast protocol that floods the network with requests for content. Our use of content routing does not suffer from traffic scalability problems seen in Gnutella networks. Gnutella does not address content availability, making it vulnerable to denial of service (DoS) attacks, like host-addressed networks (i.e., IP) [15]. Freenet performs a hybrid of caching and migration (similar to our scheme) to manage demand and can thus withstand DoS attacks.

These systems are all built at the application level and build approximations to content-addressing on top of host-based routing with a great deal of effort. Our architecture brings content routing to the network layer, making the features of these systems easier to implement, while providing a framework for building more advanced applications such as protection of DoS attacks or content migration.

Distributed data storage systems are still targets of conceptual research and preliminary implementations. One example of such a system is the OceanStore project [10], which describes a redundancy and scattering system similar to Mojo Nation, caching and migration similar to Freenet, and elaborates on many other storage and consistency concerns. A great deal of research is currently pending in the areas of distributed caching [3] [13] and cache update propagation [1].

IP Anycast routing [9] [19] presents semantics that are similar to our high-level content routing mechanism. IP Anycast is a service for transmitting messages to any host within a group of equivalent hosts, called an anycast group. When an REQ is sent using our content network, it reaches exactly one host—the closest host carrying the content specified in the REQ. This differs from IP Anycast in that the addresses target contents, and not physical hosts.

Many design features of P2P and distributed data systems are intended to avoid DoS attacks. The majority of these systems nonetheless suffer from other security and/or usability concerns, such as the inauthenticity of data and the possibility of publishing content under inaccurate descriptors. These concerns either belong to the application scope or are limitations of the protocols being used. Our content network architecture can alleviate the damage caused by some of these vulnerabilities by allowing dynamic routing to localize access to malicious content.

8 Conclusion

Our content network has several advantages over traditional networks, including scaling flexibility, resistance to denial of service attacks, and ease of implementation of peer-to-peer applications. However, we have also pointed out several areas deserving further investigation, particularly security. We have given demonstrations of content network usage, and suggested future areas in which it might be improved.

Technical contributions of this paper include a novel content routing network layer with three primitive message types: REQ, REG, and DLV. The addressable objects in this network layer were contents whose hierarchical names were assigned from an existing taxonomy, or derived from content descriptions. Further, we proposed to take advantage of the contents' location independence by actively migrating them. In one application of migration, access times were reduced when requests exhibited some degree of locality. Lastly, we described a routing framework where messages were forwarded hop-by-hop, using longest prefix matching to compute next hops. Content descriptor prefixes were disseminated and shortest paths computed by means of a distance vector routing algorithm. We also discussed an alternative design where the classification hierarchy is explicitly embedded into the network topology.

9 References

[1] Almeida, J., Broder, A., Cao, P., and Fan, L., "Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol," in Proceedings of ACM SIGCOMM 1998, Vancouver, Canada, September 1998

[2] Breslau, L., Cao, P., Fan, L., Phillips, G., and Shenker, S., "Web Caching and Zipf-like Distributions: Evidence and Implications," in Proceedings of IEEE INFOCOM 1999, New York, NY, USA, March 1999

[3] Cao, P., and Irani, S., "Cost-Aware WWW Proxy Caching Algorithms," in Proceedings of the USENIX Symposium on Internet Technologies and Systems, pp. 193--206, Monterey, CA, USA, December 1997

[4] Clarke, I., Sandberg, O., Wiley, B., and Hong, T. W., "Freenet: A Distributed Anonymous Information Storage and Retrieval System," in ICSI Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, July 2000

[5] Cunha, C. A., Bestavros, A., and Crovella, M. E., "Characteristics of WWW Client-based Traces," Technical Report TR-95-010, Boston University, Department of Computer Science, Boston, MA, USA, April 1995

[6] "The Gnutella Protocol 1.0 Specification," http://dss.clip2.com/GnutellaProtocol04.pdf

[7] "Gnutella: To the Bandwidth Barrier and Beyond," http://dss.clip2.com/gnutella.html

[8] Gritter, M., and Cheriton, D. R., "An Architecture for Content Routing Support in the Internet," in Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems (USITS'01), March 2001, San Francisco, CA, USA

[9] Katabi, D., and Wroclawski, J., "A Framework for Global IP-Anycast (GIA)," In Proceedings of ACM SIGCOMM 2000, Stockholm, Sweden, August 2000

[10] Kubiatowicz, J., Bindel, D., Chen, Y., Czerwinski, S., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Weimer, W., Wells, C., and Zhao, B., "OceanStore: An Architecture for Global-Scale Persistent Storage," in Proceeedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000), Cambridge, MA, USA, November 2000

[11] Kung, H., and Wu, C., "Hierarchical Peer-to-Peer Networks," April 2001 (submitted for publication)

[12] Markup Languages and Ontologies Homepage, http://www.semanticweb.org/knowmarkup.html

[13] Michel, S., Nguyen, K., Rosenstein, A., and Zhang, L., "Adaptive Web Caching: Toward a New Global Caching Architecture," in 3rd International Web Caching Workshop, Manchester, England, June 1998

[14] Mojo Nation Homepage, http://www.mojonation.net/

[15] Needham, R. M., "Denial of service: an example," Communications of the ACM, November 1994, pp. 42-46

[16] Open Directory Project Homepage, http://dmoz.org/

[17] Oram, A., ed., Peer-to-Peer: Harnessing the Power of Disruptive Technologies, Sebastopol, CA: O'Reilly & Associates, March 2001

[18] The O'Reilly Peer-to-Peer Conference Homepage, http://conferences.oreilly.com/p2p/

[19] Partridge, C., Mendez, T., and Milliken, W., "Host Anycasting Service," IETF RFC 1546, November 1993

[20] Postel, J., "Internet Protocol," IETF RFC 791, September, 1981

[21] Rekhter, Y. and Li, T., "An Architecture for IP Address Allocation with CIDR," IETF RFC 1518, September 1993

[22] Turner, R. M., "The Tragedy of the Commons and Distributed AI Systems," in Proceedings of the 12th International Workshop on Distributed Artificial Intelligence, Hidden Valley, PA, USA, 1993

[23] Waldvogel, M., Varghese, G., Turner, J., and Plattner, B., "Scalable High Speed IP Routing Table Lookups," in Proceedings of ACM SIGCOMM '97, Cannes, September 1997