Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 13, 2011

R. Alimi
Yale University
Z. Lu
Fudan University
P. Tao
China Telecom
Y. Yang
Yale University

July 12, 2010

A Survey of In-network Storage Systems
draft-song-decade-survey-04

Abstract

This document describes existing in-network storage systems and their applicability for DECADE.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on January 13, 2011.

Table of Contents

1. Introduction

DECADE (DECoupled Application Data Enroute) is an architecture that provides applications with access to in-network storage.

A major motivation for DECADE is the substantial increase on capacity and reduction in cost offered by storage systems. In particular, over the last decade, capacity of solid-state storage has increased 100-fold, while cost dropped to $50/GB; capacity of magnetic storage devices has increased 100-fold, while cost dropped to $0.50/GB.

High-capacity and low-cost in-network storage devices introduce substantial opportunities. One example of in-network storage is content caches supporting Web and P2P content. Different from existing content caches whose control fully reside at the owners of the caching devices, DECADE also allows applications to control access to their allocated in-network storage, as well as the resources consumed while accessing that storage (bandwidth, connections, storage space). While designed in the context of P2P applications, it may be useful to other applications as well. This document provides details on existing in-network storage solutions, and evaluates their suitability for DECADE.

We note that the techniques presented in this section are only representative of the research in this area. Rather than trying to enumerate an exhaustive list, we have chosen some typical techniques that lead to derivative works.

2. Survey Overview

2.1. Terminology and Concepts

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

This document uses terms defined in [I-D.song-decade-problem-statement].

2.2. Historical Context

In-network storage has been used previously in numerous scenarios to reducing network traffic and enable more efficient content distribution. Systems have been developed with particular use cases in mind. Thus, this survey is not meant to point out shortcomings of existing solutions, but rather to indicate where certain capabilities required in DECADE are not provided by existing systems.

In the early stage of Internet development, most Web content was stored at a central server and clients requested Web content from the central server. In this architecture, the central server was required to provide a large amount of bandwidth. Web browsing is still a primary activity on today's Internet. As more and more users access Web content, a central server can become overloaded. The use of web caches is one technique to reduce load on a central server. Web caches store frequently-requested content, and provide bandwidth for serving the content to clients.

The ongoing growth of broadband technology in the worldwide market has been driven by the hunger of customers for new multimedia services as well as Web content. In particular, the use of audio and video streaming formats has become common for delivery of rich information to the public - both residential and business.

To overcome this challenge of massive multimedia consumption, only installing more Web cache will not be enough. Moving content closer to the consumer results in greater network efficiency, improved QoS, and lower latency, while facilitating personalization of content through broadband content applications. In these edge technologies, CDN is a representative technique. Content Delivery Networks (CDN) is based on a large-scale distributed network of servers located closer to the edges of the Internet for efficient delivery of digital content including various forms of multimedia content.

Although CDN is an effective means of information access and delivery, there are two barriers to making CDN a more common service: cost and replication integrity. Deploying a CDN for publicly available content is expensive. It requires administrative control over nodes with large storage capacity at geographically dispersed locations with adequate connectivity. CDN can be scalable, but due to this administrative and cost overhead, not rapidly deployable for the common user.

The emergence and maturity of Peer to Peer (P2P) has allowed improvements to many network applications. P2P allows the use of client resources, such as CPU, memory, storage, and bandwidth, for serving content. This can reduce the amount of resources required by a content provider. Multimedia content delivery using various peer-to-peer or peer-assisted frameworks has been shown to greatly reduce the dependence on CDN and central content servers. However, popularity of P2P applications has resulted in increased traffic on ISP networks.

DECADE aims to provide a standard protocol allowing P2P applications (including Content Providers) to make use of in-network storage to reduce the traffic burden on ISP networks, while enabling P2P

applications to control access to content they have placed in in-network storage.

2.3. In-network Storage System Components

Before surveying individual technologies, we describe the basic components of in-network storage systems used to evaluate them in the context of DECADE.

Note that the network protocol(s) used by a storage system are also an important part of the design. We omit details of particular protocol choices in the current version of this document.

2.3.1. Data Access Interface

A set of operations are available to a user for accessing data in the in-network storage. Solutions typically allow both read and write, though the mechanisms for doing so can differ drastically.

2.3.2. Data Management Operations

Storage systems may provide users the ability to manage stored content. For example, operations such as delete and move can be provided to users. In this survey, we focus on data management operations that are provided to client users and omit those provided to system administrators.

2.3.3. Data Search Capability

Some storage systems may provide the capability to search or enumerate content that has been stored. In this survey, we focus on search capabilities that are provided to client users and omit those provided to system administrators.

2.3.4. Access Control Authorization

A user is able to authorize individual users to retrieve the content stored on its In-network storage. In-network storage can check the authorization of a client before it stores or retrieves content. In-network storage only permits the users with authorization to access the corresponding contents. The admission could be based on user, content, time period, etc.

2.3.5. Resource Control Interface

This is the interface through which users manage the resources on in-network storage that can be used by other peers, e.g., the bandwidth or connections. The storage system may also allow users to indicate

a time for which resources are granted.

2.3.6. Discovery Mechanism

Users use the discovery mechanism to find location of in-network storage, find access interface or resource control interface or other interfaces of in-network storage.

2.3.7. Storage Mode

The data managed by the in-network storage could be of various types. Example storage modes are file-based, object-based, or block-based.

3. P2P Cache

Caching of P2P traffic is a useful approach to reduce P2P network traffic, because objects in P2P systems are mostly immutable and the traffic is highly repetitive . In addition, making use of P2P caches do not require changes to P2P protocols and can be deployed transparently from clients.

P2P caches operate similarly to web caches, in that they temporarily store frequently-requested content. Requests for content already stored in the cache can be served from local storage instead of requiring the data to be transmitted over expensive network links.

Two types of P2P caches exist: non-transparent P2P caches and transparent P2P caches. A non-transparent cache appears as a super peer; it explicitly peers with other P2P clients. For a transparent cache, once a P2P cache is established, the network will transparently redirect P2P traffic to the cache, which either serves the file directly or passes the request on to a remote P2P user and simultaneously caches that data. Transparency is typically implemented using deep packet inspection (DPI). DPI products identify and pass P2P packets to the P2P caching system so it can cache the traffic and accelerate it.

To enable operation with existing P2P software, P2P caches directly support P2P application protocols. A large number of P2P protocols are used by P2P software, and hence are supported by caches, leading to higher complexity. Additionally, these protocols evolve over time, and new protocols are introduced.

3.1. Transparent P2P Caches

3.1.1. Data Access Interface

Data Access Interface allows P2P content to be cached (stored) and supplied (retrieved) locally such that network traffic is reduced, but it is transparent to P2P users, and P2P users implicitly use the data-access interface (in the form of their native P2P application protocol) to store or retrieve content.

3.1.2. Data Management Operations

Not provided.

3.1.3. Data Search Capability

Not provided.

3.1.4. Access Control Authorization

Not provided.

3.1.5. Resource Control Interface

Not provided.

3.1.6. Discovery Mechanism

Use of Deep Packet Inspection means no discovery mechanism provided to P2P users, it is transparent to P2P users. Since DPI is used to recognize P2P applications private protocols, P2P Cache is getting more and more complicated as the P2P applications keep evolving.

3.1.7. Storage Mode

Object-based. Chunks (typically, the unit of transfer amongst P2P clients) of content are stored in the cache.

3.2. Non-transparent P2P Caches

3.2.1. Data Access Interface

Data Access Interface allows P2P content to be cached (stored) and supplied (retrieved) locally such that network traffic is reduced. P2P users implicitly store and retrieve from the cache using the P2P application's native protocol.

3.2.2. Data Management Operations

Not provided.

3.2.3. Data Search Capability

Not provided.

3.2.4. Access Control Authorization

Not provided.

3.2.5. Resource Control Interface

Not provided.

3.2.6. Discovery Mechanism

Cache pretends to be normal peers to join the P2P overlay network. Other P2P users can find these cache nodes through overlay routing mechanism, just looking them as normal neighbor nodes.

3.2.7. Storage Mode

Object-based. Chunks (typically, the unit of transfer amongst P2P clients) of content are stored in the cache.

4. Web Cache

Web cache is a well-built technology since the late 1990s, which has been widely deployed by many ISPs to reduce bandwidth consumption and web access latency. A web cache can cache the web documents (e.g., HTML pages, images) between server and client to reduce bandwidth usage, server load, and perceived lag. A web cache server is typically shared by many clients, and stores copies of documents passing through it; subsequent requests may be satisfied from the cache if certain conditions are met.

Another form of cache is a client-side cache, typically implemented in web browsers. A client side cache can keep a local copy of all pages recently displayed by browser, and when the user returns to one of these web pages, the local cached copy is reused.

A related protocol for P2P applications to use web cache is HPTP (HTTP based Peer to Peer). It proposes to share chunks of P2P files/ streams using HTTP protocol with cache-control headers.

4.1. Data Access Interface

Users explicitly read from a web cache by making requests, but they cannot explicitly write data into it. Data is implicitly stored into the web cache by requesting content that not already cached and meets policy restrictions of the cache provider.

4.2. Data Management Operations

Not provided.

4.3. Data Search Capability

Not provided.

4.4. Access Control Authorization

Not provided.

4.5. Resource Control Interface

Not provided.

4.6. Discovery Mechanism

Web Caches can be transparently deployed between Web Server and Web Clients, employing DPI for discovery. Alternatively, web caches could be explicitly discovered by clients using techniques such as DNS or manual configuration.

4.7. Storage Mode

Object based. Web content is keyed within the cache by HTTP Request fields, such as Method, URI, and Headers.

5. CDN

Pathan et al. introduced the main idea and function of Content Delivery Networks (CDN) [PR07]. CDN provides services that improve network performance by maximizing bandwidth, improving accessibility and maintaining correctness through content replication. They offer fast and reliable applications and services by distributing content to cache or edge servers located close to users.

A CDN has some combination of content-delivery, request-routing, distribution and accounting infrastructure. The content-delivery infrastructure consists of a set of edge servers (also called

surrogates) that deliver copies of content to end-users. The request-routing infrastructure is responsible to directing client request to appropriate edge servers. It also interacts with the distribution infrastructure to keep an up-to-date view of the content stored in the CDN caches. The distribution infrastructure moves content from the origin server to the CDN edge servers and ensures consistency of content in the caches. The accounting infrastructure maintains logs of client accesses and records the usage of the CDN servers. This information is used for traffic reporting and usage-based billing.

In practice, CDN typically host static content including images, video, media clips, advertisements, and other embedded objects for dynamic Web content. A focus for CDNs is the ability to publish and deliver content to end-users in a reliable and timely manner. A CDN focuses on building its network infrastructure to provide the following services and functionalities: storage and management of content; distribution of content among surrogates; cache management; delivery of static, dynamic and streaming content; backup and disaster recovery solutions; and monitoring, performance measurement and reporting.

Examples of existing CDNs are Akamai, Limelight, and CloudFront.

The following description uses the term Content Provider to refer to the entity purchasing CDN service, and the term Client to refer to the subscriber requesting content via the CDN from the Content Provider.

5.1. Data Access Interface

CDN is typically an internal closed system, and CDN just provide read (retrieve) access interface to clients but they don't provide write(store) access interface to clients. Content provider can access to network edge servers and store content to them, or edge servers retrieve content from content provider, but client nodes just can retrieve content from edge servers.

5.2. Data Management Operations

Content Provider can manage the data distributed in different cache nodes, such as moving one hot data from one cache node to another cache node, or deleting one rarely-accessed data in one cache node, but client user nodes have no right to perform these operations.

5.3. Data Search Capability

Content provider can search or enumerate what data each cache node hold, but client user nodes have no right to perform these operations.

5.4. Access Control Authorization

Content Providers typically cannot control per-client access to content accessed via a CDN.

5.5. Resource Control Interface

Not provided.

5.6. Discovery Mechanism

Content providers can directly find internal CDN cache nodes to store content, since they typically have an explicit business relationship. Clients can locate CDN nodes through DNS or other redirection mechanism.

5.7. Storage Mode

A file-based storage mode is typically used. In most cases, CDN cache nodes cache the entire file from content provider, and sometimes they also can only cache some objects, such as file prefix or file suffix.

5.8. Comments

6. NFS

The Network File System is designed to allow users to access files over a network in a manner similar to how local storage is accessed. NFS is typically used in local area network or enterprise settings, though changes made in later versions of NFS make it easier to operate over the Internet.

6.1. Data Access Interface

Traditional file-system operations such as read, write, and update (overwrite) are provided. Locking is provided to support concurrent access by multiple clients.

6.2. Data Management Operations

Traditional file-system operations such as move and delete are provided.

6.3. Data Search Capability

User has the ability to list contents of directories to find filenames matching desired criteria.

6.4. Access Control Authorization

Files and directories can be protected using read, write, and execute permissions for the files owner, group, and the public (others). Also, NFSv4.1 has a rich ACL model allowing a list of Access Control Entries (ACEs) to be configured for each file or directory. The ACEs can specify per-user read/write access to file data, file/directory attributes, creation/deletion of files in a directory, etc.

6.5. Resource Control Interface

While disk space quotas can be configured, it typically limits the total amount of storage allocated to a particular user. User control of bandwidth and connections used by remote peers is not provided.

6.6. Discovery Mechanism

Manual configuration is typically used. Clients address NFS servers by providing a hostname and a directory that should be mounted.

6.7. Storage Mode

File-based storage, allowing files to be organized into directories.

6.8. Comments

The efficiency and scalability of the NFS access control method is a concern in the context of DECADE. In particular, Section 6.2.1 of [RFC5661] states that:

> Only ACEs that have a "who" that matches the requester are considered.

Thus, in the context of DECADE, to specify per-peer access control policies for an object, a client would need to explicitly configure the ACL for the object for each individual peer. A concern with this approach is scalability when a client's peers may change frequently and ACLs for many small objects need to be updated constantly during

participation in a swarm.

Note that NFS v4.1's usage of RPCSEC_GSS provides support for multiple security mechanisms. Kerberos V5 is required, but others such as X.509 Certificates are also supported by way of GSS-API. Note, however, that NFSv4.1's usage of such security mechanisms is limited to linking a requesting user to a particular account maintained by the NFS server.

7. WebDAV

WebDAV [RFC4918] is a protocol designed for Web content authoring. It is developed as an extension to HTTP/1.1, meaning it can be simpler to integrate into existing software. WebDAV supports traditional operations for reading/writing from storage, as well as other constructs such as locking and collections which are important when multiple users collaborate to author or edit a set of documents.

7.1. Data Access Interface

Traditional read and write operations are supported (using HTTP GET and PUT methods, respectively). Locking is provided to ease concurrent access by multiple clients.

7.2. Data Management Operations

WebDAV supports traditional file-system operations such as move, delete and copy. Objects are organized into collections, and these operations can also be performed on collections. WebDAV also allows objects to have user-defined properties.

7.3. Data Search Capability

User has the ability to list contents of collections to find objects matching desired criteria. A SEARCH extension [RFC5323] has also been specified allowing listing of objects matching client-defined criteria.

7.4. Access Control Authorization

An ACL extension [RFC3744] is provided for WebDAV. ACLs allow both user- and group-based access control policies (relating to reading, writing, properties, locking, etc) to be defined for objects and collections.

A ticketing extension [I-D.ito-dav-ticket] has also been proposed, but has not progressed passed an Internet Draft. This extension

allows a client to request the WebDAV server to create a "ticket" (e.g., for reading an object) that can be distributed to other clients. Tickets may be given expiration times, or may only allow for a fixed number of uses. The proposed extension requires the server to generate tickets and maintain state for outstanding tickets.

7.5. Resource Control Interface

An extension [RFC4331] allows disk space quotas to be configured for Collections. The extension also allows WebDAV clients to query current disk space usage. User control of bandwidth and connections used by remote peers is not provided.

7.6. Discovery Mechanism

Manual configuration is typically used. Clients address WebDAV servers by providing a hostname.

7.7. Storage Mode

File-based storage (a non-collection resource can typically be thought of as a "file"). Files may be organized into collections, which typically map on to the HTTP Path hierarchy.

7.8. Comments

The efficiency and scalability of the WebDAV access control method is a concern in the context of DECADE, for similar reasons as stated in Section 6.8 for NFS. The proposed WebDAV ticketing extension partially alleviates this concern, but the particular technique may need further evaluation before being applied to DECADE. In particular, since DECADE clients may continuously upload/download a large number of small-size objects, and a single DECADE server may need to scale to many concurrent DECADE clients, requiring the server to maintain ticket state and generate tickets may not be the best design choice. Server-generated tickets can also increase latency for data transport operations depending on the message flow used by DECADE.

8. iSCSI

SCSI is a set of protocols enabling communication with storage devices such as disk drives and tapes; iSCSI [RFC3720] is a protocol enabling SCSI commands to be sent over TCP. As in SCSI, iSCSI allows an Initiator to send commands to a Target. These commands operate on the device level as opposed to individual data objects stored on the

device.

8.1. Data Access Interface

Read and write commands indicate which data is to be read or written by specifying the offset (using Logical Block Addressing) into the storage device. The size of data to be read or written is an additional parameter in the command.

8.2. Data Management Operations

Since commands operate at the device level, management operations are different than with traditional file systems. Management commands for SCSI/iSCSI including explicit device control such as starting and stopping the device and formatting the device.

8.3. Data Search Capability

SCSI/iSCSI does not provide the ability to search for particular data within a device. Note that such capabilities can be implemented outside of iSCSI.

8.4. Access Control Authorization

Since SCSI/iSCSI operates at the device level, neither authentication nor authorization are provided for individual data objects. However, iSCSI does use CHAP [RFC1994] to authenticate initiators and targets when accessing storage devices. Note that such capabilities can be implemented outside of iSCSI.

8.5. Resource Control Interface

Not provided.

8.6. Discovery Mechanism

Manual configuration may be used. An alternative is the Internet Storage Name Service (iSNS) [RFC4171] provides the ability to discover available storage resources.

8.7. Storage Mode

Block-based. SCSI/iSCSI provides an Initiator block-level access to the storage device.

9. OAuth

OAuth [I-D.hammer-oauth] is a protocol that richens the traditional client-server authentication model for web resources. In particular, OAuth distinguishes the "client" from the "resource owner", thus enabling a resource owner to authorize a particular client for access (e.g., for a particular lifetime) to private resources.

Note that OAuth is a protocol for authentication and purposefully does not itself provide access to network storage. We include it in the survey so that its authentication model can be evaluated in the context of DECADE.

9.1. Data Access Interface

Not applicable.

9.2. Data Management Operations

Not applicable.

9.3. Data Search Capability

Not applicable.

9.4. Access Control Authorization

While similar in spirit to the WebDAV ticketing extensions [I-D.ito-dav-ticket], OAuth instead uses the following process: (1) a client constructs a delegation request, (2) the client forwards the request to the resource owner for authorization, (3) the resource owner authorizes the request, and finally (4) a callback is made to the client indicating that its request has been authorized.

Once the process is complete, the client has a set of token credentials that grant it access to the protected resource. The token credentials may have an expiration time, and they can also be revoked by the resource owner at any time.

9.5. Resource Control Interface

Not applicable.

9.6. Discovery Mechanism

Not applicable.

9.7. Storage Mode

Not applicable.

9.8. Comments

The ticketing mechanism requires server involvement and the discussion relating to WebDAV's proposed ticketing mechanism (see Section 7.8) applies here as well.

10. Amazon S3

Amazon S3 [AmazonS3] provides an online storage service. Users create buckets, and each bucket can contain stored objects. Users are provided an interface through which they can manage their buckets. Amazon S3 is popular backend storage for other services. Another related storage service is the Blob Service provided by Windows Azure [Azure].

10.1. Data Access Interface

Users can read, and write objects.

10.2. Data Management Operations

Users can delete previously-stored objects.

10.3. Data Search Capability

Users can list contents of buckets to find objects matching desired criteria.

10.4. Access Control Authorization

Access to stored objects can be restricted by owner, a list of other Amazon Web Service users, all Amazon Web Service Users, or open to all users (anonymous access). Another option is for the owner to generate and sign a query (e.g., a query to read an object) that can be used by any user until an owner-defined expiration time.

10.5. Resource Control Interface

Not provided.

10.6. Discovery Mechanism

Users are provided a well-known DNS name (either a default provided by Amazon, or one customized by a particular user). Users accessing S3 storage use DNS to discover an IP address where S3 requests can be sent.

10.7. Storage Mode

Object-based, with the extension that objects can be organized into user-defined buckets.

11. OceanStore

OceanStore is a storage platform developed at UC Berkeley that provides globally-distributed storage. OceanStore implements a model where multiple storage providers can pool resources together. Thus, a major focus is on resiliency and self-organization and self-maintenance.

The protocol is resilient to some storage nodes being compromised by utilizing Byzantine agreement and erasure codes to store data at primary replicas.

11.1. Data Access Interface

Users may read and write objects

11.2. Data Management Operations

Objects may be replaced by newer versions, and multiple versions of an object may be maintained.

11.3. Data Search Capability

Not provided.

11.4. Access Control Authorization

Provided, but specifics are unclear from published paper.

11.5. Resource Control Interface

Not provided.

11.6. Discovery Mechanism

Users require an entry-point into the system in the form of one storage node that is part of OceanStore.

11.7. Storage Mode

Object-based, though interfaces have been provided for NFS and HTTP.

12. Cache-and-Forward Architecture

Cache-and-Forward [PRDW08] is an architecture content delivery services in the future Internet. In this architecture, storage can be exploited at nodes with the network, either directly at routers or deployed nearby routers. CNF is based on the concept of store-and-forward routers with large storage, providing for opportunistic delivery to occasionally disconnected mobile users and for in-network caching of content. The proposed CNF protocol uses reliable hop-by-hop transfer of large data files between CNF routers in place of an end-to-end transport protocol like TCP.

12.1. Data Access Interface

Users implicitly store content at Cache-and-forward routers by requesting files. End hosts read content from in-network storage by submitting queries for content.

12.2. Data Management Operations

Not provided.

12.3. Data Search Capability

Not provided.

12.4. Access Control Authorization

Not provided.

12.5. Resource Control Interface

Not provided.

12.6. Discovery Mechanism

A query including a location-independent content ID is sent to the network, and routed to a Cache-and-forward router, which handles

retrieval of the data and forwarding to the end host.

12.7. Storage Mode

Object-based (with objects representing individual files). The architecture proposes to cache large files at storage within the network, though files could be made to represent smaller chunks of larger files.

13. Network Traffic Redundancy Elimination

Another form of in-network storage is Redundancy Elimination (RE), or identifying and removing repeated content from network transfers. This technique has been proposed to improve network performance in many types of networks, such as ISP backbones and enterprise access links. One example redundancy elimination proposal is SmartRE, proposed by Anand et al., which focuses on network-wide redundancy elimination. In packet-level redundancy elimination, forwarding elements are equipped with additional storage which can be used to cache data from forwarded packets. Upstream routers may replace packet data with a fingerprint that tells a downstream router how to decode and reconstruct the packet based on cached data.

13.1. Data Access Interface

Redundancy-elimination are typically transparent to the user. Writing into the storage is done by transferring data that has not already been cached. Storage is read when users transmit data identical to previously-transmitted data.

13.2. Data Management Operations

Not provided.

13.3. Data Search Capability

Not provided.

13.4. Access Control Authorization

Not provided. However, note that the content provider still retains control over which peers receive the requested data. The returned data is simple "compressed" as it is transferred within the network.

13.5. Resource Control Interface

Not provided. The content provider still retains control over the rate at which packets are sent to a peer. The packet size within the network may be reduced.

13.6. Discovery Mechanism

No discovery mechanism is necessary. Routers can use redundancy-elimination without the users' knowledge.

13.7. Storage Mode

Object-based, with "objects" being data from packets transmitted within the network.

14. BranchCache

BranchCache [BranchCache] is a feature integrated into Windows (Windows 7 and Windows Server 2008R2) that aims to optimize enterprise branch office file access over the WAN links. The main goals are to reduce WAN link utilization and improve application responsiveness by caching and sharing content within a branch while still maintaining end-to-end security. BranchCache allows files retrieved from the web servers and file servers located in headquarters or datacenters to be cached in remote branch offices, and shared among users in the same branch accessing the same content. BranchCache operates transparently by instrumenting the HTTP and SMB components of the networking stack. It provides two modes of operation: Distributed Cache and Hosted Cache.

In both modes, a client always contacts a BranchCache-enabled content server first to get the content identifiers for local search. If the content is cached locally, the client then retrieves the content within the branch. Otherwise, the client will go back to the original content server to request the content. The two modes differ in how the content is shared.

In the Hosted Cache mode, a locally provisioned server acts as a cache for files retrieved from the servers. After getting the content identifiers, the client first consults the cache for the desired file. If it is not present in the cache, the client retrieves it from the content server and sends it to the cache for storage.

In the Distributed Cache mode, a client first queries other clients in the same network using the Web Services Discovery multicast

protocol. As in the Hosted Cache mode, the client retrieves the file from the content server it is not available locally. After retrieving the file (either from another client or the content server), the client stores the file locally.

The original content server always authorizes requests from clients. Cached content is encrypted, and clients can only decrypt the data using keys derived from metadata returned by the content server. In addition to instrumenting the networking stack at clients, content servers must also support BranchCache.

14.1. Data Access Interface

Clients transparently retrieve (read) data from a cache (other clients or a Hosted Cache) since it operates by instrumenting the networking stack. In Hosted Cache mode, clients write data to the Hosted Cache once it is retrieved from the content server.

14.2. Data Management Operations

Not provided.

14.3. Data Search Capability

Not provided.

14.4. Access Control Authorization

Transferred content is encrypted, and can only be decrypted by keys derived from data received from the original content server. Though data may be transferred to unauthorized clients, end-to-end security is maintained by only allowing authorized clients to decrypt the data.

14.5. Resource Control Interface

The storage capacity of caches on the clients and Hosted Caches are configurable by system administrators. The Hosted Cache further allows configuration of the maximum number of simultaneous client accesses. In the Distributed Caching mode, exponential back-off and throttling mechanisms are utilized to prevent reply storms of popular content requests. The client will also spread data block access among multiple serving clients that have the content (complete or partial) to improve latency and provide some load balancing.

14.6. Discovery Mechanism

The Distributed Cache mode uses multicast for discovery of other clients and content within a local network. Currently, the Hosted Cache mode uses policy provisioning or manual configuration of the server used as the Hosted Cache.

14.7. Storage Mode

Object-based.

15. Conclusions

Though there have been many successful in-network storage systems, they have been designed for use cases different from those defined in DECADE. As a result, their functionality and feature set does not meet the requirements defined for DECADE. DECADE aims to provide a standard protocol for P2P applications and content providers to access and control in-network storage, resulting in increased network efficiency while retaining control over content shared with peers. Additionally, defining a standard protocol can reduce complexity of in-network storage since multiple P2P application protocols no longer need to be implemented by in-network storage systems.

16. Security Considerations

This draft is a survey of existing in-network storage systems, and does not introduce any security considerations beyond those of the surveyed systems.

For more information on security considerations of DECADE, see [I-D.song-decade-problem-statement].

17. IANA Considerations

This document does not have any IANA Considerations.

18. Acknowledgments

The authors would like to thank Haibin Song, Yu-Shun Wang and Ning Zong for comments and contributions to this document.

19. References

19.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

19.2. Informative References

[I-D.song-decade-problem-statement]
Yongchao, S., Zong, N., Yang, Y., and R. Alimi, "DECoupled Application Data Enroute (DECADE) Problem Statement", draft-song-decade-problem-statement-02 (work in progress), July 2010.

[I-D.gu-decade-reqs]
Yingjie, G., Yongchao, S., Yang, Y., and R. Alimi, "DECADE Requirements", draft-gu-decade-reqs-04 (work in progress), March 2010.

[I-D.hammer-oauth]
Hammer-Lahav, E., "The OAuth 1.0 Protocol", draft-hammer-oauth-10 (work in progress), February 2010.

[RFC1994] Simpson, W., "PPP Challenge Handshake Authentication Protocol (CHAP)", RFC 1994, August 1996.

[RFC3720] Satran, J., Meth, K., Sapuntzakis, C., Chadalapaka, M., and E. Zeidner, "Internet Small Computer Systems Interface (iSCSI)", RFC 3720, April 2004.

[RFC4171] Tseng, J., Gibbons, K., Travostino, F., Du Laney, C., and J. Souza, "Internet Storage Name Service (iSNS)", RFC 4171, September 2005.

[RFC5661] Shepler, S., Eisler, M., and D. Noveck, "Network File System (NFS) Version 4 Minor Version 1 Protocol", RFC 5661, January 2010.

[RFC4918] Dusseault, L., "HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV)", RFC 4918, June 2007.

[RFC5323] Reschke, J., Reddy, S., Davis, J., and A. Babich, "Web Distributed Authoring and Versioning (WebDAV) SEARCH", RFC 5323, November 2008.

[RFC4331] Korver, B. and L. Dusseault, "Quota and Size Properties fo r Distributed Authoring and Versioning (DAV) Collections", RFC 4331, February 2006.

[RFC3744] Clemm, G., Reschke, J., Sedlar, E., and J. Whitehead, "Web Distributed Authoring and Versioning (WebDAV) Access Control Protocol", RFC 3744, May 2004.

[I-D.ito-dav-ticket] Ito, K., "Ticket-Based Access Control Extension to WebDAV", draft-ito-dav-ticket-00 (work in progress), October 2001.

[HYAL08] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. Liu, and A. Silberschatz., "P4P: Provider Portal for Applications", In ACM SIGCOMM 2008.

[MCM08] M. Hefeeda, C. Hsu, and K. Mokhtarian., "pCache: A Proxy Cache for Peer-to-Peer Traffic", In ACM SIGCOMM'08 Technical Demonstration.

[JZL08] Jie Wu, ZhiHui Lu, BiSheng Liu, et al., "PeerCDN: A Novel P2P Network Assisted Streaming Content Delivery Network Scheme", In 8th IEEE International Conference on Computer and Information Technology (CIT2008).

[GYZ07] G. Shen, Y. Wang, Y. Xiong, B.Y. Zhao, Z.-L. Zhang, "HPTP: Relieving the tension between isps and p2p", In 6th International workshop on Peer-To-Peer Systems (IPTPS2007).

[JCL09] Jiajun Wang, Cheng Huang, Jin Li., "On ISP-friendly rate allocation for peer-assisted VoD", In ACM Multimedia 2008.

[GH09] Geoff Huston, Telstra., "Web Caching", In The Internet Protocol Journal Volume 2, No. 3.

[McGraw02] Scott Hull et al., "Content Delivery Networks: Web Switching for Security, Availability, and Speed".

[PR07] Pathan, A.K., Buyya, R., "A Taxonomy and Survey of Content Delivery Networks", In Grid Computing and Distributed Systems Laboratory in University of Melbourne, Technology Report, Feb. 2007.

[AmazonS3] Amazon, "Amazon Simple Storage Service (Amazon S3)", http://aws.amazon.com/s3/.

[Azure] Microsoft Corporation., "Windows Azure Blob - Programming Blob Storage".

[OceanStore]
S. Rhea, P. Eaton, D. Geels, H. Weatherspoon, B. Zhao, and J. Kubiatowicz., "Pond: the OceanStore Prototype", In FAST 2003.

[AVA09] A. Anand, V. Sekar, A. Akella., "SmartRE: An Architecture for Coordinated Network-wide Redundancy Elimination", In SIGCOMM 2009.

[PRDW08] S. Paul, R. Yates, D. Raychaudhuri, J. Kurose., "The Cache-and-Forward Network Architecture for Efficient Mobile Content Delivery Services in the Future Internet", In Innovations in NGN: Future Network and Services, 2008.

[BranchCache]
Microsoft Corporation., "BranchCache", http://technet.microsoft.com/en-us/network/dd425028.aspx.

Authors' Addresses

Richard Alimi
Yale University

Email: richard.alimi@yale.edu

ZhiHui Lu
Fudan University

Email: lzh@fudan.edu.cn

Pang Tao
China Telecom

Email: pangt@gsta.com

Yang Richard Yang
Yale University

Email: yry@cs.yale.edu