Internet Engineering Task Force (IETF)
Request for Comments: 6392
Category: Informational
ISSN: 2070-1721

R. Alimi, Ed.
Google
A. Rahman, Ed.
InterDigital Communications, LLC
Y. Yang, Ed.
Yale University

October 2011

A Survey of In-Network Storage Systems

Abstract

This document surveys deployed and experimental in-network storage systems and describes their applicability for the DECADE (DECoupled Application Data Enroute) architecture.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc6392.

Table of Contents

1. Introduction

DECADE (DECoupled Application Data Enroute) is an architecture that provides applications with access to provider-based in-network storage for content distribution (hereafter referred to as only "in-network storage" in this document). With access to in-network storage, content distribution applications can be designed to place less load on network infrastructure. As a simple example, a peer of a Peer-to-Peer (P2P) application may upload to other peers through its in-network storage, saving its usage of last-mile uplink bandwidth. See [1] for further discussion.

A major motivation for DECADE is the substantial increase in capacity and reduction in cost offered by storage systems. For example, over the last two decades, there has been at least a 30-fold increase in the amount of storage that a customer can get for a given price (for flash memory and hard disk drives) [2] [3] [4].

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. DECADE differs from existing content caches whose control fully resides with the owners of the caching devices in that 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, DECADE may be useful to other applications as well. This document provides details on deployed and experimental in-network storage solutions, and evaluates their suitability for DECADE.

We note that the survey presented in this document is 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

This document uses terms defined in [1].

2.2. Historical Context

In-network storage has been used previously in numerous scenarios to reduce network traffic and enable more efficient content distribution. This section presents a brief history of content distribution techniques and illustrates how DECADE relates to past

approaches. 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 [5] 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. 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, just installing more Web caches will not be enough. Moving content closer to the consumer results in greater network efficiency, improved Quality of Service (QoS), and lower latency, while facilitating personalization of content through broadband content applications. In these edge technologies, Content Delivery Networks (CDNs) are a representative technique. CDNs are based on a large-scale distributed network of servers located closer to customers for efficient delivery of digital content, including various forms of multimedia content.

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

The emergence and maturation of 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 P2P or peer-assisted frameworks has been shown to greatly reduce the dependence on CDNs and central content servers. However, the popularity of P2P applications has resulted in increased traffic on ISP networks. P2P

caches (both transparent and non-transparent) have been introduced as a way to reduce the burden. Though they can be effective in reducing traffic in certain areas of ISP networks, P2P caches have their shortcomings. In particular, they are application-dependent and thus difficult to keep up to date with new and evolving P2P application protocols. Second, applications may benefit from explicit control of in-network storage, which P2P caches do not provide. See [1] for further discussion.

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.

3. In-Network Storage System Components

Before surveying individual technologies, we describe the basic components of in-network storage. For consistency and for ease of comparison, we use the same model to evaluate each storage technology in this document.

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

3.1. Data Access Interface

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

3.2. Data Management Operations

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

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 users and omit those provided to system administrators. An example of a search would be to find the list of items stored by a given user over a given period of time.

3.4. Access Control Authorization

Storage systems typically allow a user, content owner, or some other entity to define the access policies for the in-network storage system. The in-network storage system then checks the authorization of a user before it stores or retrieves content. We define three types of access control authorization: public-unrestricted, public-restricted, and private.

"Public-unrestricted" refers to content on an in-network storage system that is widely available to all clients (i.e., without restrictions). An example is accessing Wikipedia on the Web, or anonymous access to FTP sites.

"Public-restricted" refers to content on an in-network storage system that is available to a restricted (though still potentially large) set of clients, but that does not require any confidential credentials from the client. An example is some content (e.g., a TV show episode) on the Internet that can only be viewable in selected countries or networks (i.e., white/black lists or black-out areas).

"Private" refers to content on an in-network storage system that is only made available to one or more clients presenting the required confidential credentials (e.g., password or key). This content is not available to anyone without the proper confidential access credentials.

Note that a combination of access control types may be applicable for a given scenario. For example, the retrieval (read) of content from an in-network storage system may be public-unrestricted, but the storage (write) to the same system may be private.

3.5. Resource Control Interface

This is the interface through which users manage the resources on in-network storage systems 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.

3.6. Discovery Mechanism

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

3.7. Storage Mode

Storage systems may use the following modes of storage: file system, object-based, or block-based.

A file system typically organizes files into a hierarchical tree structure. Each level of the hierarchy normally contains zero or more directories, each with zero or more files. A file system may also be flat or use some other organizing principle.

We define an object-based storage mode as one that stores discrete chunks of data (e.g., IP datagrams or another type of aggregation useful to an application) without a pre-defined hierarchy or meta-structure.

We define a block-based storage mode as one that stores a raw sequence of bytes, with a client being able to read and/or write data at offsets within that sequence. Data is typically accessed in blocks for efficiency. A common example for this storage mode is raw access to a hard disk.

In this survey, we define "storage mode" to refer to how data is structured within the system, which may not be the same as how it is accessed by a client. For example, a caching system may cache objects with hierarchical names, but may internally use an object-based storage mode.

4. In-Network Storage Systems

This section surveys in-network storage systems using the methodology defined above. The survey includes some systems that are widely deployed today, some systems that are just being deployed, and some experimental systems. The survey covers both traditional client-server architectures and P2P architectures. The surveyed systems are listed in alphabetical order. Also, for each system, a brief explanation of the relevance to DECADE is given.

4.1. Amazon S3

Amazon S3 (Simple Storage Service) [6] provides an online storage service using Web (HTTP) interfaces. 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 a popular backend storage service for other services. Other related storage services are the Blob Service provided by Windows Azure [7], Google Storage for Developers [8], and Dropbox [9].

4.1.1. Applicability to DECADE

Amazon S3 is a very widely used (deployed) example of in-network storage. Amazon S3 leases the storage to third-party companies for disparate services. In particular, Amazon S3 has a rich model for authorization (using signed queries) to integrate with a wide variety of use cases. A focus for Amazon S3 is scalability. Particular simplifications that were made are the absence of a general, hierarchical namespace and the inability to update the contents of existing data.

4.1.2. Data Access Interface

Users can read and write objects.

4.1.3. Data Management Operations

Users can delete previously stored objects.

4.1.4. Data Search Capability

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

4.1.5. Access Control Authorization

All methods of access control are supported for clients: public-unrestricted, public-restricted, and private.

For example, access to stored objects can be restricted by an owner, a list of other Amazon S3 Web Service users, or all Amazon S3 Web Service users; or can be 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.

4.1.6. Resource Control Interface

Not provided.

4.1.7. Discovery Mechanism

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

4.1.8. Storage Mode

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

4.2. BranchCache

BranchCache [10] is a feature integrated into Windows (Windows 7 and Windows Server 2008R2) that aims to optimize enterprise branch office file access over 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 data centers 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 Server Message Block (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 [11]. As in the Hosted Cache mode, the client retrieves the file from the content server if 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 such that clients can decrypt the data only 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.

4.2.1. Applicability to DECADE

BranchCache is an example of an in-network storage system primarily targeted at enterprise networks. It supports a P2P-like mode (Distributed Cache) as well as a client-server mode (Hosted Cache). Integration into the Microsoft OS will ensure wide distribution of this in-network storage technology.

4.2.2. Data Access Interface

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

4.2.3. Data Management Operations

Not provided.

4.2.4. Data Search Capability

Not provided.

4.2.5. Access Control Authorization

The access control method for clients is private. For example, 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.

4.2.6. Resource Control Interface

The storage capacity of caches on the clients and Hosted Caches is configurable by system administrators. The Hosted Cache further allows configuration of the maximum number of simultaneous client accesses. In the Distributed Cache 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.

4.2.7. 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. In this mode, the address of the server may be found via DNS.

4.2.8. Storage Mode

Object-based.

4.3. Cache-and-Forward Architecture

Cache-and-Forward (CNF) [12] is an architecture for content delivery services for the future Internet. In this architecture, storage can be exploited on nodes within the network, either directly on routers or deployed near the 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 such as TCP.

4.3.1. Applicability to DECADE

CNF is an example of an experimental in-network storage system that would require storage space on (or near) a large number of routers in the Internet if it was deployed. As the name of the system implies, it would provide short-term caching and not long-term network storage.

4.3.2. Data Access Interface

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

4.3.3. Data Management Operations

Not provided.

4.3.4. Data Search Capability

Not provided.

4.3.5. Access Control Authorization

The access control method is public-restricted (to any client that is part of the CNF network).

4.3.6. Resource Control Interface

Not provided.

4.3.7. Discovery Mechanism

A query including a location-independent content ID is sent to the network and routed to a CNF router, which handles retrieval of the data and forwarding to the end host.

4.3.8. Storage Mode

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

4.4. Cloud Data Management Interface

The Cloud Data Management Interface (CDMI) is a specification to access and manage cloud storage. CDMI is specified by the Storage Networking Industry Association (SNIA).

CDMI is a functional interface that applications can use to create, retrieve, update, and delete data elements from the cloud. As part of this interface, the client will be able to discover the capabilities of the cloud storage offering and use this interface to manage containers and the data that is placed in them. In addition, metadata can be set on containers and their contained data elements through this interface [13].

CDMI follows a traditional client-server model, and operates over an HTTP interface using the Representational State Transfer (REST) model. Similar to Amazon S3 buckets (see Section 4.1), users may create containers in which data objects may be stored. Even though data objects may be accessed via a user-defined name within a container, it is also possible to access data objects via a storage-defined Object ID, which is provided in the response upon creation of a data object.

4.4.1. Applicability to DECADE

CDMI is an important initiative to standardize storage interfaces for cloud services, which are rapidly becoming an important type of storage service. In particular, it specifies a set of operations for creating, reading, writing, and managing data objects at a remote server (or set of servers) via HTTP.

4.4.2. Data Access Interface

Users can read and write data objects, and also update data in existing data objects. CDMI data objects are sent on the wire embedded as a field in a JavaScript Object Notation (JSON) object. The protocol also defines interfaces in which the contents of data objects can be written via simple HTTP GET/PUT operations.

4.4.3. Data Management Operations

Users can delete already-existing data objects. The create operation also supports modes in which the created object is copied or moved from an existing data object.

Data system metadata also allows users to configure policies regarding time-to-live, after which a data object is automatically deleted, as well as the redundancy with which a data object is stored.

4.4.4. Data Search Capability

Users may list the contents of containers to locate data objects matching any desired criteria.

4.4.5. Access Control Authorization

All methods of access control for clients are supported: public-unrestricted, public-restricted, and private.

In particular, CDMI allows access to data objects to be protected by Access Control Lists (ACLs) that can allow or restrict access based on user name, group, administrative status, or whether a user is authenticated or anonymous.

4.4.6. Resource Control Interface

CDMI supports attributes ’cdmi_max_latency’ and ’cdmi_max_throughput’ (set at either the level of containers, or a specific data object), which control the level of service offered to any users accessing a particular data object.

4.4.7. Discovery Mechanism

Users are provided a well-known DNS name. The DNS name is resolved to determine the IP address to which requests may be sent.

4.4.8. Storage Mode

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

4.5. Content Delivery Network

A CDN provides services that improve performance by minimizing the amount of data transmitted through the network, improving accessibility, and maintaining correctness through content replication. CDNs offer fast and reliable applications and services by distributing content to cache or edge servers located close to users. See [14] for an additional taxonomy and survey.

A CDN has some combination of content delivery, request routing, distribution, and accounting infrastructures. 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 for directing client requests 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, a CDN typically hosts static content including images, video, media clips, advertisements, and other embedded objects for Web viewing. 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 a CDN service, and the term "client" to refer to the subscriber requesting content via the CDN from the content provider.

4.5.1. Applicability to DECADE

CDNs are a very widely used (deployed) example of in-network storage for multimedia content. The existence and operation of the storage system are totally transparent to the end user. CDNs typically require a strong business relationship between the content providers and content distributors, and often the business relationship extends to the ISPs.

4.5.2. Data Access Interface

A CDN is typically a closed system, and generally provides only a read (retrieve) access interface to clients. A CDN typically does not provide a write (store) access interface to clients. The content provider can access network edge servers and store content on them, or edge servers can retrieve content from content providers. Client nodes can only retrieve content from edge servers.

4.5.3. Data Management Operations

A content provider can manage the data distributed in different cache nodes, such as moving popular data objects from one cache node to another cache node, or deleting rarely accessed data objects in cache nodes. User nodes, however, have no right to perform these operations.

4.5.4. Data Search Capability

A content provider can search or enumerate the data each cache node stores. User nodes cannot perform search operations.

4.5.5. Access Control Authorization

All methods of access control (for reading) are supported for clients: public-unrestricted, public-restricted, and private. Some CDN edge servers allow usage of HTTP basic authentication with the origin server or restrictions by IP address, or they can use a token-based technique to allow the origin server to apply its own authorization criteria.

As mentioned previously, clients typically cannot write to the CDN. Writing is typically a private operation for the content providers.

4.5.6. Resource Control Interface

Not provided.

4.5.7. 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 mechanisms.

4.5.8. Storage Mode

Though the addressing of objects uses URLs that typically refer to objects in a hierarchical fashion, the storage mode is typically object-based.

4.6. Delay-Tolerant Network

The Delay-Tolerant Network (DTN) [15] is an evolution of an architecture originally designed for the Interplanetary Internet. The Interplanetary Internet is a communication system envisioned to provide Internet-like services across interplanetary distances in support of deep space exploration. The DTN architecture can be utilized in various operational environments characterized by severe communication disruptions, disconnections, and high delays (e.g., a month-long loss of connectivity between two planetary networks because of high solar radiation due to sun spots). The DTN architecture is thus suitable for environments including deep space networks, sensor-based networks, certain satellite networks, and underwater acoustic networks.

A key aspect of the DTN is a store-and-forward overlay layer called the "Bundle Protocol" or "Bundle Layer", which exists between the transport and application layers [16]. The Bundle Layer forms a logical overlay that employs persistent storage to help combat long-term network interruptions by providing a store-and-forward service. While traditional IP networks are also based on store-and-forward principles, the amount of time of a packet being kept in "storage" at a traditional IP router is typically on the order of milliseconds (or less). In contrast, the DTN architecture assumes that most Bundle Layer nodes will use some form of persistent storage (e.g., hard disk, flash memory, etc.) for DTN packets because of the nature of the DTN environment.

4.6.1. Applicability to DECADE

The DTN is an example of an experimental in-network storage system that would require fundamental changes to the Internet protocols.

4.6.2. Data Access Interface

Users implicitly cause content to be stored (until successfully forwarded) at Bundle Layer nodes by initiating/terminating any transaction that traverses the DTN.

4.6.3. Data Management Operations

Users can implicitly cause deletion of content stored at Bundle Layer nodes via a "time-to-live" type of parameter that the user can control (for transactions originating from the user).

4.6.4. Data Search Capability

Not provided.

4.6.5. Access Control Authorization

The access control method is public-restricted (to any client that is part of the DTN) or private.

4.6.6. Resource Control Interface

Not provided.

4.6.7. Discovery Mechanism

A Uniform Resource Identifier (URI) approach is used as the basis of the addressing scheme for DTN transactions (and subsequent store-and-forward routing through the DTN network).

4.6.8. Storage Mode

Object-based. DTN applications send data to the Bundle Layer, which then breaks the data into segments. These segments are then routed through the DTN network, and stored in Bundle Layer nodes as required (before being forwarded).

4.7. Named Data Networking

Named Data Networking (NDN) [17] is a research initiative that proposes to move to a new model of addressing and routing for the Internet. NDN uses "named data"-based routing and forwarding, to replace the current IP-address-based model. NDN also uses name-based data caching in the routers.

Each NDN Data packet will be assigned a content name and will be cryptographically signed. Data delivery is driven by the requesting end. Routers disseminate name-based prefix announcements by using routing protocols such as Intermediate System to Intermediate System (IS-IS) or the Border Gateway Protocol (BGP). The requester will send out an "Interest" packet, which identifies the name of the data that it wants. Routers that receive this Interest packet will remember the interface it came from and will then forward it on a name-based routing protocol. Once an Interest packet reaches a node that has the desired data, a named Data packet is sent back, which carries both the name and content of the data, along with a digital signature of the producer. This named Data packet is then forwarded back to the original requester on the reverse path of the Interest packet [18].

A key aspect of NDN is that routers have the capability to cache the named data. If a request for the same data (i.e., same name) comes to the router, then the NDN router will forward the named data stored locally to fulfill the request. The proponents of NDN believe that the network can be designed naturally, matching data delivery characteristics instead of communication between endpoints, because data delivery has become the primary use of the network.

4.7.1. Applicability to DECADE

NDN is an example of an experimental in-network storage system that would require storage space on a large number of routers in the Internet. Named Data packets would be kept in storage in the NDN routers and provided to new requesters of the same data.

4.7.2. Data Access Interface

Users implicitly store content at NDN routers by requesting content (the named Data packets) from the network. Subsequent requests by different users for the same content will cause the named Data packets to be read from the NDN routers' in-network storage.

4.7.3. Data Management Operations

Users do not have the direct ability to delete content stored in the NDN routers. However, there will be some type of time-to-live parameter associated with the named Data packets, though this has not yet been specified.

4.7.4. Data Search Capability

Not provided.

4.7.5. Access Control Authorization

All methods of access control for clients are supported: public-unrestricted, public-restricted, and private.

The basic security mechanism in NDN is for the sender to digitally sign the content (the named Data packets) that it sends. It is envisioned that a complete access control system can be built on top of NDN, though this has not yet been specified.

4.7.6. Resource Control Interface

Not provided.

4.7.7. Discovery Mechanism

Names are used as the basis of the addressing and discovery scheme for NDN (and subsequent store-and-forward routing through the NDN network). NDN names are assumed to be hierarchical and to be able to be deterministically constructed. This is still an active area of research.

4.7.8. Storage Mode

Object-based. NDN sends named Data packets through the network. These Data packets are routed through the NDN network and stored in NDN routers.

4.8. Network of Information

Similar to NDN (see Section 4.7), Network of Information (NetInf) [19] is another information-centric approach in which the named data objects are the basic component of the networking architecture. NetInf is thus moving away from today's host-centric networking

architecture where the nodes in the network are the primary objects. In today's network, the information objects are named relative to the hosts they are stored on (e.g., http://www.example.com/information-object.txt).

The NetInf naming and security framework builds the foundation for an information-centric security model that integrates security deeply into the architecture. In this model, trust is based on the information itself. Information objects (IOs) are given a unique name with cryptographic properties. Together with additional metadata, the name can be used to verify the data integrity as well as several other security properties, such as self-certification, name persistency, and owner authentication and identification. The approach also gives some benefits over the security model in today's host-centric networks, as it minimizes the need for trust in the infrastructure, including the hosts providing the data, the channel, or the resolution service.

In NetInf, the information objects are published into the network. They are registered with a Name Resolution Service (NRS). The NRS is also used to register network locators that can be used to retrieve data objects that represent the published IOs. When a receiver wants to retrieve an IO, the request for the IO is resolved by the NRS into a set of locators. These locators are then used to retrieve a copy of the data object from the "best" available source(s). NetInf is open to use any type of underlying transport network. The locators can thus be a heterogeneous set, e.g., IPv4, IPv6, Medium Access Control (MAC), etc.

NetInf will make extensive use of caching of information objects in the network and will provide network functionality that is similar to what overlay solutions such as CDNs and P2P distribution networks (e.g., BitTorrent) provide today.

4.8.1. Applicability to DECADE

NetInf is an example of an experimental information-centric network architecture that will require storage space for storage and caching of information objects on a large number of NetInf nodes in the Internet.

4.8.2. Data Access Interface

Users will publish IOs with specific IDs into the network. This is done by the client sending a register message to the NRS stating that the IO with the specific ID is available. When another user wishes to retrieve the IO, they will use the given ID to make a request for the IO. The ID is then resolved by the NRS, and the IO is delivered from a nearby in-network storage location.

4.8.3. Data Management Operations

Users do not have the direct ability to delete content stored in the NetInf nodes. However, there can be some type of time-to-live parameter associated with the information objects, though this has not yet been specified.

4.8.4. Data Search Capability

Not provided.

4.8.5. Access Control Authorization

All methods of access control for clients are supported: public-unrestricted, public-restricted, and private. The basic security mechanism in NetInf is for the publisher to digitally sign the content of the information object that it publishes. It is envisioned that a complete access control system can be built on top of NetInf, though this has not yet been specified.

4.8.6. Resource Control Interface

Not provided.

4.8.7. Discovery Mechanism

NetInf IDs are used for naming and accessing information objects. The IDs are resolved by the NRS into locators that are used for routing and transport of data through the transport networks. This is still an active area of research.

4.8.8. Storage Mode

Object-based. From an application perspective, NetInf can be used for publishing entire files or chunks of files. NetInf is agnostic to the application perspective and treats everything as information objects.

4.9. Network Traffic Redundancy Elimination

Redundancy Elimination (RE) is used for 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 of an RE proposal is SmartRE [20], proposed by Anand et al., which focuses on network-wide RE. In packet-level RE, forwarding elements are equipped with additional storage that 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.

4.9.1. Applicability to DECADE

RE is an example of an experimental in-network storage system that would require a large amount of associated packet processing at routers if it was ever deployed.

4.9.2. Data Access Interface

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

4.9.3. Data Management Operations

Not provided.

4.9.4. Data Search Capability

Not provided.

4.9.5. Access Control Authorization

The access control method is public-restricted (to any client that is part of the RE network). Note that the content provider still retains control over which peers receive the requested data. The returned data is "compressed" as it is transferred within the network.

4.9.6. 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.

4.9.7. Discovery Mechanism

No discovery mechanism is necessary. Routers can use RE without the users' knowledge.

4.9.8. Storage Mode

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

4.10. OceanStore

OceanStore [21] is a storage platform developed at the University of California, 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, 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.

4.10.1. Applicability to DECADE

OceanStore is an example of an experimental in-network storage system that provides a high degree of network resilience to failure scenarios.

4.10.2. Data Access Interface

Users may read and write objects.

4.10.3. Data Management Operations

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

4.10.4. Data Search Capability

Not provided.

4.10.5. Access Control Authorization

Provided, but specifics for clients are unclear from the available references.

4.10.6. Resource Control Interface

Not provided.

4.10.7. Discovery Mechanism

Users require an entry point into the system in the form of one storage node that is part of OceanStore. If a hostname is provided, the address of a storage node may be determined via DNS.

4.10.8. Storage Mode

Object-based.

4.11. 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 does not require changes to P2P protocols and can be deployed transparently from clients.

P2P caches operate similarly to Web caches (Section 4.14) 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: transparent P2P caches and non-transparent P2P caches.

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 and accelerate the traffic.

A non-transparent cache appears as a super peer; it explicitly peers with other P2P clients.

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.

4.11.1. Applicability to DECADE

A P2P cache is an example of in-network storage for P2P systems. However, unlike DECADE, the existence and operation of the storage system are totally transparent to the end user.

4.11.2. Transparent P2P Caches

4.11.2.1. Data Access Interface

The 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.

4.11.2.2. Data Management Operations

Not provided.

4.11.2.3. Data Search Capability

Not provided.

4.11.2.4. Access Control Authorization

The access control method is typically public-restricted (to any client that is part of the P2P channel or swarm).

4.11.2.5. Resource Control Interface

Not provided.

4.11.2.6. Discovery Mechanism

The use of DPI means that no discovery mechanism is provided to P2P users; it is transparent to P2P users. Since DPI is used to recognize P2P applications' private protocols, P2P cache implementations must be updated as new applications are added and existing protocols evolve.

4.11.2.7. Storage Mode

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

4.11.3. Non-Transparent P2P Caches

4.11.3.1. Data Access Interface

The 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.

4.11.3.2. Data Management Operations

Not provided.

4.11.3.3. Data Search Capability

Not provided.

4.11.3.4. Access Control Authorization

The access control method is typically public-restricted (to any client that is part of the P2P channel or swarm).

4.11.3.5. Resource Control Interface

Not provided.

4.11.3.6. Discovery Mechanism

A P2P cache node behaves as if it were a normal peer in order to join the P2P overlay network. Other P2P users can find such a cache node through an overlay routing mechanism and can interact with it as if it were a normal neighbor node.

4.11.3.7. Storage Mode

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

4.12. Photo Sharing

There are a growing number of popular online photo-sharing (storing) systems. For example, the Kodak Gallery system [22] serves over 60 million users and stores billions of images [23]. Other well-known examples of photo-sharing systems include Flickr [24] and ImageShack [25]. There are also a number of popular blogging

services, such as Tumblr [26], that specialize in sharing large numbers of photos as well as other multimedia content (e.g., video, text, audio, etc.) as part of their service. All of these in-network storage systems utilize both free and paid subscription models.

Most photo-sharing systems are based on a traditional client-server architecture. However, a minority of systems also offer a P2P mode of operation. The client-server architecture is typically based on HTTP, with a browser client and a Web server.

4.12.1. Applicability to DECADE

Photo sharing is a very widely used (deployed) example of in-network storage where the end user has direct visibility and extensive control of the system. The typical end-user interface is through an HTTP-based Web browser.

4.12.2. Data Access Interface

Users can read (view) and write (store) photos.

4.12.3. Data Management Operations

Users can delete previously stored photos.

4.12.4. Data Search Capability

Users can tag photos and/or organize them using sophisticated Web photo album generators. Users can then search for objects (photos) matching desired criteria.

4.12.5. Access Control Authorization

The access control method for clients is typically either private or public-unrestricted. For example, writing (storing) to a photo blog is typically private to the owner of the account. However, all other clients can view (read) the contents of the blog (i.e., public-unrestricted). Some photo-sharing Websites provide private access to read photos to allow sharing with a limited set of friends.

4.12.6. Resource Control Interface

Not provided.

4.12.7. Discovery Mechanism

Clients usually log on manually to a central Web page for the service and enter the appropriate information to access the desired information. The address to which the client connects is usually determined by DNS using the hostname from the provided URL.

4.12.8. Storage Mode

File system (file-based). Photos are usually stored as files. They can then be organized into meta-structures (e.g., albums, galleries, etc.) using sophisticated Web photo album generators.

4.13. Usenet

Usenet is a distributed Internet-based discussion (message) system. The Usenet messages are arranged as a set of "newsgroups" that are classified hierarchically by subject. Usenet information is distributed and stored among a large conglomeration of servers that store and forward messages to one another in so-called news feeds. Individual users may read messages from, and post messages to, a local news server typically operated by an ISP. This local server communicates with other servers and exchanges articles with them. In this fashion, the message is copied from server to server and eventually reaches every server in the network [27].

Traditional Usenet as described above operates as a P2P network between the servers, and in a client-server architecture between the user and their local news server. The user requires a Usenet client to be installed on their computer and a Usenet server account (through their ISP). However, with the rise of Web browsers, the Usenet architecture is evolving to be Web-based. The most popular example of this is Google Groups, where Google hosts all the newsgroups and client access is via a standard HTTP-based Web browser [28].

4.13.1. Applicability to DECADE

Usenet is a historically very important and widely used (deployed) example of in-network storage in the Internet. The use of this system is rapidly declining, but efforts have been made to preserve the stored content for historical purposes.

4.13.2. Data Access Interface

Users can read and post (store) messages.

4.13.3. Data Management Operations

Users sometimes have limited ability to delete messages that they previously posted.

4.13.4. Data Search Capability

Traditionally, users could manually search through the newsgroups, as they are classified hierarchically by subject. In the newer Web-based systems, there is also an automatic search capability based on key-word matches.

4.13.5. Access Control Authorization

The access control method is either public-unrestricted or private (to client members of that newsgroup).

4.13.6. Resource Control Interface

Not provided.

4.13.7. Discovery Mechanism

Clients usually log on manually to their Usenet accounts. DNS may be used to resolve hostnames to their corresponding addresses.

4.13.8. Storage Mode

File system. Messages are usually stored as files that are then organized hierarchically by subject into newsgroups.

4.14. Web Cache

Web cache [29] has been widely deployed by many ISPs to reduce bandwidth consumption and Web access latency since the late 1990s. 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 a 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) [30]. It proposes sharing chunks of P2P files/streams using HTTP with cache-control headers.

4.14.1. Applicability to DECADE

Web cache is a very widely used (deployed) example of in-network storage for the key Internet application of Web browsing. The existence and operation of the storage system are transparent to the end user in most cases. The content caching time is controlled by time-to-live parameters associated with the original content. The principle of Web caching is to speed up Web page reading by using (the same) content previously requested by another user to service a new user.

4.14.2. 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 in the Web cache by requesting content that is not already cached and meets policy restrictions of the cache provider.

4.14.3. Data Management Operations

Not provided.

4.14.4. Data Search Capability

Not provided.

4.14.5. Access Control Authorization

The access control method for clients is public-unrestricted. It is important to note that if content is authenticated or encrypted (e.g., HTTPS, Secure Socket Layer (SSL)), it will not be cached. Also, if the content is flagged as private (vs. public) at the HTTP level by the origin server, it will not be cached.

4.14.6. Resource Control Interface

Not provided.

4.14.7. Discovery Mechanism

Web caches can be transparently deployed between a 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.

4.14.8. Storage Mode

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

4.15. Observations Regarding In-Network Storage Systems

The following observations about the surveyed in-network storage systems are made in the context of DECADE as defined by [1].

The majority of the surveyed systems were designed for client-server architectures and do not support P2P. However, there are some important exceptions, especially for some of the newer technologies such as BranchCache and P2P cache, that do support a P2P mode of operation.

The P2P cache systems are interesting, since they do not require changes to the P2P applications themselves. However, this is also a limitation in that they are required to support each application protocol.

Many of the surveyed systems were designed for caching as opposed to long-term network storage. Thus, during DECADE protocol design, it should be carefully considered whether a caching mode should be supported in addition to a long-term network storage mode. There is typically a trade-off between providing a caching mode and long-term (and usually also reliable) storage with regards to some performance metrics. Note that [1] identifies issues with classical caching from a DECADE perspective, such as the fact that P2P caches typically do not allow users to explicitly control content stored in the cache.

Certain components of the surveyed systems are outside of the scope of DECADE. For example, a protocol used for searching across multiple DECADE servers is out of scope. However, applications may still be able to implement such functionality if DECADE exposes the appropriate primitives. This has the benefit of keeping the core in-network storage systems simple, while permitting diverse applications to design mechanisms that meet their own requirements.

Today, most in-network storage systems follow some variant of the authorization model of public-unrestricted, public-restricted, and private. For DECADE, we may need to evolve the authorization model to support a resource owner (e.g., end user) authorization, in addition to the network authorization.

5. Storage and Other Related Protocols

This section surveys existing storage and other related protocols, as well as comments on the usage of these protocols to satisfy DECADE's use cases. The surveyed protocols are listed alphabetically.

5.1. HTTP

HTTP [31] is a key protocol for the World Wide Web. It is a stateless client-server protocol that allows applications to be designed using the REST model. HTTP is often associated with downloading (reading) content from Web servers to Web browsers, but it also has support for uploading (writing) content to Web servers. It has been used as the underlying protocol for other protocols, such as Web Distributed Authoring and Versioning (WebDAV).

HTTP is used in some of the most popular in-network storage systems surveyed previously, including CDNs, photo sharing, and Web cache. Usage of HTTP by a storage protocol implies that no extra software is required in the client (i.e., Web-based client), as all standard Web browsers are based on HTTP.

5.1.1. Data Access Interface

Basic read and write operations are supported (using HTTP GET, PUT, and POST methods).

5.1.2. Data Management Operations

Not provided.

5.1.3. Data Search Capability

Not provided.

5.1.4. Access Control Authorization

All methods of access control for clients are supported: public-unrestricted, public-restricted, and private.

The majority of Web pages are public-unrestricted in terms of reading but do not allow any uploading of content. In-network storage systems range from private or public-unrestricted for photo sharing (described in Section 4.12.5) to public-unrestricted for Web caching (described in Section 4.14.5).

5.1.5. Resource Control Interface

Not provided.

5.1.6. Discovery Mechanism

Manual configuration is typically used. Clients typically address HTTP servers by providing a hostname, which is resolved to an address using DNS.

5.1.7. Storage Mode

HTTP is a protocol; it thus does not define a storage mode. However, a non-collection resource can typically be thought of as a "file". These files may be organized into collections, which typically map onto the HTTP path hierarchy, creating the illusion of a file system.

5.1.8. Comments

HTTP is based on a client-server architecture and thus is not directly applicable for the DECADE focus on P2P. Also, HTTP offers only a rudimentary toolset for storage operations compared to some of the other storage protocols.

5.2. iSCSI

Small Computer System Interface (SCSI) is a set of protocols enabling communication with storage devices such as disk drives and tapes; Internet SCSI (iSCSI) [32] 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.

5.2.1. 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.

5.2.2. Data Management Operations

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

5.2.3. 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.

5.2.4. Access Control Authorization

With respect to access to devices, the access control method is private. iSCSI uses the Challenge Handshake Authentication Protocol (CHAP) [33] to authenticate Initiators and Targets when accessing storage devices. However, since SCSI/iSCSI operates at the device level, neither authentication nor authorization is provided for individual data objects. Note that such capabilities can be implemented outside of iSCSI.

5.2.5. Resource Control Interface

Not provided.

5.2.6. Discovery Mechanism

Manual configuration may be used. An alternative is the Internet Storage Name Service (iSNS) [34], which provides the ability to discover available storage resources.

5.2.7. Storage Mode

As a protocol, iSCSI does not explicitly have a storage mode. However, it provides block-based access to clients. SCSI/iSCSI provides an Initiator with block-level access to the storage device.

5.3. NFS

The Network File System (NFS) 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 networks or in enterprise settings, though changes made in later versions of NFS (e.g., [35]) make it easier to operate over the Internet.

5.3.1. 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.

5.3.2. Data Management Operations

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

5.3.3. Data Search Capability

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

5.3.4. Access Control Authorization

All methods of access control for clients are supported: public-unrestricted, public-restricted, and private. For example, files and directories can be protected using read, write, and execute permissions for the files' owner and group, and for 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.

5.3.5. Resource Control Interface

While disk space quotas can be configured, administrative policy 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.

5.3.6. Discovery Mechanism

Manual configuration is typically used. Clients address NFS servers by providing a hostname and a directory that should be mounted. DNS may be used to look up an address for the provided hostname.

5.3.7. Storage Mode

As a protocol, there is no defined internal storage mode. However, implementations typically use the underlying file-system storage. Note that extensions have been defined for alternate storage modes (e.g., block-based [36] and object-based [37]).

5.3.8. Comments

The efficiency and scalability of the NFS access control method are concerns in the context of DECADE. In particular, Section 6.2.1 of [35] 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 NFSv4.1'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 the Generic Security Service Application Program Interface (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.

5.4. OAuth

Open Authorization (OAuth) [38] is a protocol that enriches 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.

We include OAuth in this survey so that its authentication model can be evaluated in the context of DECADE. OAuth itself, however, is not a network storage protocol.

5.4.1. Data Access Interface

Not provided.

5.4.2. Data Management Operations

Not provided.

5.4.3. Data Search Capability

Not provided.

5.4.4. Access Control Authorization

Not provided. While similar in spirit to the WebDAV ticketing extensions [39], 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.

5.4.5. Resource Control Interface

Not provided.

5.4.6. Discovery Mechanism

Not provided.

5.4.7. Storage Mode

Not provided.

5.4.8. Comments

The ticketing mechanism requires server involvement, and the discussion relating to WebDAV's proposed ticketing mechanism (see Section 5.5.8) applies here as well.

5.5. WebDAV

WebDAV [40] is a protocol designed for Web content authoring. It is developed as an extension to HTTP (described in Section 5.1), meaning that 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, that are important when multiple users collaborate to author or edit a set of documents.

5.5.1. Data Access Interface

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

5.5.2. 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.

5.5.3. Data Search Capability

The user has the ability to list contents of collections to find objects matching desired criteria. A SEARCH extension [41] has also been specified allowing listing of objects matching client-defined criteria.

5.5.4. Access Control Authorization

All methods of access control for clients are supported: public-unrestricted, public-restricted, and private.

For example, an ACL extension [42] is provided for WebDAV. ACLs allow both user-based and group-based access control policies (relating to reading, writing, properties, locking, etc.) to be defined for objects and collections.

A ticketing extension [39] has also been proposed, but has not progressed since 2001. 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.

5.5.5. Resource Control Interface

An extension [43] 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.

5.5.6. Discovery Mechanism

Manual configuration is typically used. Clients address WebDAV servers by providing a hostname, which can be resolved to an address using DNS.

5.5.7. Storage Mode

Though no storage mode is explicitly defined, WebDAV can be thought of as providing file system (file-based) storage to a client. A non-collection resource can typically be thought of as a "file". Files may be organized into collections, which typically map onto the HTTP path hierarchy.

5.5.8. Comments

The efficiency and scalability of the WebDAV access control method are concerns in the context of DECADE, for reasons similar to those stated in Section 5.3.8 for NFS. The proposed WebDAV ticketing extension partially alleviates these concerns, 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.

5.6. Observations Regarding Storage and Related Protocols

The following observations about the surveyed storage and related protocols are made in the context of DECADE as defined by [1].

All of the surveyed protocols were primarily designed for client-server architectures and not for P2P. However, it is conceivable that some of the protocols could be adapted to work in a P2P architecture.

Several popular in-network storage systems today use HTTP as their key protocol, even though it is not classically considered as a storage protocol. HTTP is a stateless protocol that is used to design RESTful applications. HTTP is a well-supported and widely implemented protocol that can provide important insights for DECADE.

The majority of the surveyed protocols do not support low-latency access for applications such as live streaming. This was one of the key general requirements for DECADE.

The majority of the surveyed protocols do not support any form of resource control interface. Resource control is required for users to manage the resources on in-network storage systems, e.g., the bandwidth or connections, that can be used by other peers. Resource control is a key capability required for DECADE.

Nearly all surveyed protocols did, however, support the following capabilities required for DECADE: ability of the user to read/write content, some form of access control, some form of error indication, and the ability to traverse firewalls and NATs.

6. Conclusions

Though there have been many successful in-network storage systems, they have been designed for use cases different from those defined in DECADE. For example, many of the surveyed in-network storage systems and protocols were designed for client-server architectures and not P2P. No surveyed system or protocol has the functionality and features to fully meet the set of 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 the complexity of in-network storage, since multiple P2P application protocols no longer need to be implemented by in-network storage systems.

7. Security Considerations

This document 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 [1].

8. Contributors

The editors would like to thank the following people for contributing to the development of this document:

- ZhiHui Lv

- Borje Ohlman

- Pang Tao

- Lucy Yong

- Juan Carlos Zuniga

9. Acknowledgments

The editors would like to thank the following people for providing valuable comments to various draft versions of this document: David Bryan, Tao Mao, Haibin Song, Ove Strandberg, Yu-Shun Wang, Richard Woundy, Yunfei Zhang, and Ning Zong.

10. Informative References

[1] Song, H., Zong, N., Yang, Y., and R. Alimi, "DECoupled Application Data Enroute (DECADE) Problem Statement", Work in Progress, October 2011.

[2] Storage Search, "Flash Memory vs. Hard Disk Drives -- Which Will Win?", <http://www.storagesearch.com/semico-art1.html>.

[3] Brisken, W., "Hard Drive Price Trends", US VLBI Technical Meeting, May 2008.

[4] Woundy, R., "TSV P2P Efforts -- From an ISP's Perspective", IETF 81, Quebec, Canada, July 2011, <http://www.ietf.org/proceedings/81/slides/tsvarea-3.pdf>.

[5] Gu, Y., Bryan, D., Yang, Y., and R. Alimi, "DECADE Requirements", Work in Progress, September 2011.

[6] Amazon Web Services, "Amazon Simple Storage Service (Amazon S3)", <http://aws.amazon.com/s3/>.

[7] Calder, B., Wang, T., Mainali, S., and J. Wu, "Windows Azure Blob -- Programming Blob Storage", May 2009, <http://www.microsoft.com/windowsazure/whitepapers/>.

[8] Google, "Google Storage for Developers", <http://code.google.com/apis/storage>.

[9] Dropbox, "Dropbox Features", <http://www.dropbox.com/features>.

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

[11] Microsoft Corporation, "Web Services Dynamic Discovery (WS-Discovery)", April 2005, <http://specs.xmlsoap.org/ws/2005/04/discovery/ws-discovery.pdf>.

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

[13] SNIA, "Cloud Data Management Interface (CDMI)", <http://www.snia.org/cdmi>.

[14] Pathan, A.K. and Buyya, R., "A Taxonomy and Survey of Content Delivery Networks", Grid Computing and Distributed Systems Laboratory, University of Melbourne, Technical Report, February 2007.

[15] Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., and H. Weiss, "Delay-Tolerant Networking Architecture", RFC 4838, April 2007.

[16] Scott, K. and S. Burleigh, "Bundle Protocol Specification", RFC 5050, November 2007.

[17] Named Data Networking, "Named Data Networking Home Page", <http://www.named-data.net/>.

[18] Named Data Networking, "Named Data Networking (NDN) Project", <http://www.named-data.net/ndn-proj.pdf>.

[19] Network of Information, "NetInf Overview", <http://www.netinf.org/home/overview/>.

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

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

[22] Kodak, "Kodak Gallery Home Page", <http://www.kodakgallery.com/gallery/welcome.jsp>.

[23] Wikipedia, "Kodak Gallery", <http://en.wikipedia.org/wiki/Kodak_Gallery>.

[24] Flickr, "Flickr Home Page", <http://www.flickr.com>.

[25] ImageShack, "ImageShack Home Page", <http://imageshack.us>.

[26] Tumblr, "Tumblr Home Page", <http://www.tumblr.com>.

[27] Wikipedia, "Usenet", <http://en.wikipedia.org/wiki/Usenet>.

[28] Google, "Google Groups", <http://groups.google.com>.

[29] Huston, G., Telstra, "Web Caching", The Internet Protocol Journal Volume 2, No. 3.

[30] Shen, G., Wang, Y., Xiong, Y., Zhao, B., and Z-L. Zhang, "HPTP: Relieving the Tension between ISPs and P2P", 6th International Workshop on Peer-To-Peer Systems (IPTPS2007).

[31] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

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

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

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

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

[36] Black, D., Fridella, S., and J. Glasgow, "Parallel NFS (pNFS) Block/Volume Layout", RFC 5663, January 2010.

[37] Halevy, B., Welch, B., and J. Zelenka, "Object-Based Parallel NFS (pNFS) Operations", RFC 5664, January 2010.

[38] Hammer-Lahav, E., Ed., "The OAuth 1.0 Protocol", RFC 5849, April 2010.

[39] Ito, K., "Ticket-Based Access Control Extension to WebDAV", Work in Progress, October 2001.

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

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

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

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

Authors' Addresses

Richard Alimi (editor)
Google

EMail: ralimi@google.com

Akbar Rahman (editor)
InterDigital Communications, LLC

EMail: Akbar.Rahman@InterDigital.com

Yang Richard Yang (editor)
Yale University

EMail: yry@cs.yale.edu