DECADE
Internet-Draft
Intended status: Informational
Expires: November 22, 2011

R. Alimi
Google
Y. Yang
Yale University
A. Rahman
InterDigital Communications, LLC
D. Kutscher
NEC
H. Liu
Yale University

May 21, 2011

DECADE Architecture
draft-ietf-decade-arch-01

Abstract

Peer-to-peer (P2P) applications have become widely used on the Internet today and make up a large portion of the traffic in many networks. One technique to improve the network efficiency of P2P applications is to introduce storage capabilities within the networks. The DECADE Working Group has been formed with the goal of developing an architecture to provide this capability. This document presents an architecture, discusses the underlying principles, and identifies core components and protocols supporting the architecture.

This Internet-Draft will expire on November 22, 2011.

Table of Contents

1. Introduction

Peer-to-peer (P2P) applications have become widely used on the Internet today to distribute contents, and they contribute a large portion of the traffic in many networks. The DECADE Working Group has been formed with the goal of developing an architecture to introduce in-network storage to be used by such applications, to achieve more efficient content distribution. Specifically, in many subscriber networks, it is typically more expensive to upgrade network equipment in the "last-mile", because it can involve replacing equipment and upgrading wiring at individual homes, businesses, and devices such as DSLAMs and CMTSs. On the other hand, it can be cheaper to upgrade the core infrastructure, which involves fewer components that are shared by many subscribers. See [I-D.ietf-decade-problem-statement] for a more complete discussion of the problem domain and general discussions of the capabilities to be provided by DECADE.

This document presents a potential architecture of providing in-network storage that can be integrated into content distribution applications. The primary focus is P2P-based content distribution, but the architecture may be useful to other applications with similar characteristics and requirements. In particular, content distribution applications that may split data into smaller pieces for distribution may be able to utilize DECADE.

The design philosophy of the DECADE architecture is to provide only the core functionalities that are needed for applications to make use of in-network storage. With such core functionalities, the protocol may be simpler and easier to support by storage providers. If more complex functionalities are needed by a certain application or class of applications, it may be layered on top of the DECADE protocol.

The DECADE protocol will leverage existing transport and application layer protocols and will be designed to work with a small set of alternative IETF protocols.

This document proceeds in two steps. First, it details the core architectural principles that can guide the DECADE design. Next, given these core principles, this document presents the core components of the DECADE architecture and identifies usage of existing protocols and where there is a need for new protocol development.

This document will be updated to track the progress of the DECADE survey [I-D.ietf-decade-survey] and requirements [I-D.ietf-decade-reqs] drafts.

2. Entities

2.1. DECADE Storage Servers

DECADE storage servers are operated by DECADE storage providers and provide the DECADE functionality as specified in this document, including mechanisms to store, retrieve and manage data. A storage provider may typically operate multiple storage servers.

2.2. DECADE Storage Provider

A DECADE in-network storage provider deploys and/or manages DECADE servers within a network. A storage provider may also own or manage the network in which the DECADE servers are deployed.

A DECADE storage provider, possibly in cooperation with one or more network providers, determines deployment locations for DECADE servers and determines the available resources for each.

2.3. DECADE Content Providers

DECADE content providers access DECADE storage servers (by way of a DECADE client) to upload and manage data. A content provider can access one or more storage servers. A content provider may be a single process or a distributed application (e.g., in a P2P scenario).

2.4. DECADE Content Consumers

DECADE content consumers access storage servers (by way of a DECADE client) to download data that has previously been stored by a content provider. A content consumer can access one or more storage servers. A content consumer may be a single process or a distributed application (e.g., in a P2P scenario). An instance of a distributed application, such as a P2P application, may both provide content to and consume content from DECADE storage servers.

2.5. Content Distribution Application

A content distribution application is a distributed application designed for dissemination of possibly-large data to multiple consumers. Content Distribution Applications typically divide content into smaller immutable blocks for dissemination.

The term Application Developer refers to the developer of a particular Content Distribution Application.

2.6. Application End-Point

An Application End-Point is an instance of a Content Distribution Application that makes use of DECADE server(s). A particular Application End-Point may be a DECADE Content Provider, a DECADE Content Consumer, or both.

An Application End-Point need not be an active member of a "swarm" to interact with the DECADE storage system. That is, an End-Point may interact with the DECADE storage servers as an offline activity.

3. Architectural Principles

We identify the following key principles.

3.1. Decoupled Control/Metadata and Data Planes

The DECADE infrastructure is intended to support multiple content distribution applications. A complete content distribution application implements a set of control and management functions including content search, indexing and collection, access control, ad insertion, replication, request routing, and QoS scheduling. A observation of DECADE is that different content distribution applications can have unique considerations designing the control and signaling functions:

- Metadata Management: Traditional file systems provide a standard metadata abstraction: a recursive structure of directories to offer namespace management; each file is an opaque byte stream. In content distribution, applications may use different metadata management schemes. For example, one application may use a sequence of blocks (e.g., for file sharing), while another application may use a sequence of frames (with different sizes) indexed by time. For example, Apple Live Streaming uses a dynamic playlist to allow switching of frames encoded at different encoding rates.

- Resource and Access Control: For example, a major competitive advantage of many successful P2P systems is their substantial expertise in achieving highly efficient utilization of peer and infrastructural resources. For instance, many live P2P systems have their specific algorithms in constructing topologies to achieve low-latency, high-bandwidth streaming. They continue to fine-tune such algorithms.

Given the diversity of control-plane functions, in-network storage should export basic mechanisms and allow as much flexibility as

possible to the control planes to implement specific policies. This conforms to the end-to-end systems principle and allows innovation and satisfaction of specific business goals.

Specifically, in the DECADE architecture, the control plane focuses on the application-specific, complex, and/or processing intensive functions while the data plane provides storage and data transport functions.

- Control plane: Signals details of where the data is to be downloaded from. The control signals may also include the time, quality of service, and receivers of the download. It also provides higher layer meta-data management functions such as defining the sequence of data blocks forming a higher layer content object. These are behaviors designed and implemented by the Application. By Application, we mean the broad sense that includes other control plane protocols.

- Data plane: Stores and transfers basic data objects as instructed by the Application's Control Plane.

Decoupling control plane and data plane is not new. For example, OpenFlow is an implementation of this principle for Internet routing, where the computation of the forwarding table and the application of the forwarding table are separated. Google File System applies the principle to file system design, by utilizing the Master to handle the meta-data management, and the Chunk Servers to handle the data plane functions (i.e., read and write of chunks of data). NFS4 also implements this principle.

Note that applications may have different Data Plane implementations in order to support particular requirements (e.g., low latency). In order to provide interoperability, the DECADE architecture does not intend to enable arbitrary data transport protocols. However, the architecture may allow for more-than-one data transport protocols to be used.

Also note that although an application's existing control plane functions remain implemented within the application, the particular implementation may need to be adjusted to support DECADE.

3.2. Immutable Data Objects

A property of bulk contents to be broadly distributed is that they typically are immutable -- once a piece of content is generated, it is typically not modified. It is not common that bulk contents such as video frames and images need to be modified after distribution.

Many content distribution applications divide content objects into blocks for two reasons: (1) multipath: different blocks may be fetched from different content sources in parallel, and (2) faster recovery and verification: individual blocks may be recovered and verified. Typically, applications use a block size larger than a single packet in order to reduce control overhead.

Common applications whose content matches this model include P2P streaming (live and video-on-demand) and P2P file-sharing content. However, other additional types of applications may match this model.

DECADE adopts a design in which immutable data objects may be stored at a storage server. Applications may consider existing blocks as DECADE data objects, or they may adjust block sizes before storing in a DECADE server.

Focusing on immutable data blocks in the data plane can substantially simplify the data plane design, since consistency requirements can be relaxed. It also allows effective reuse of data blocks and de-duplication of redundant data.

Depending on its specific requirements, an application may store data in DECADE servers such that each data object is completely self-contained (e.g., a complete, independently decodable video segment). An application may also divide data into chunks that require application level assembly. The DECADE architecture and protocols are agnostic to the nature of the data objects and do not specify a fixed size for them.

Note that immutable content may still be deleted. Also note that immutable data blocks do not imply that contents cannot be modified. For example, a meta-data management function of the control plane may associate a name with a sequence of immutable blocks. If one of the blocks is modified, the meta-data management function changes the mapping of the name to a new sequence of immutable blocks.

3.3. Data Object Identifiers

Objects that are stored in a DECADE storage server can be accessed by DECADE content consumers by a resource identifier that has been assigned within a certain application context.

Because a DECADE content consumer can access more than one storage server within a single application context, a data object that is replicated across different storage servers managed by a DECADE storage provider, can be accessed by a single identifier.

Note that since data objects are immutable, it is possible to support

persistent identifiers for data objects.

3.4. Explicit Control

To support the functions of an application's control plane, applications must be able to know and control which data is stored at particular locations. Thus, in contrast with content caches, applications are given explicit control over the placement (selection of a DECADE server), deletion (or expiration policy), and access control for stored data.

Consider deletion/expiration policy as a simple example. An application may require that a DECADE server store content for a relatively short period of time (e.g., for live-streaming data). Another application may need to store content for a longer duration (e.g., for video-on-demand).

3.5. Resource and Data Access Control through User Delegation

DECADE provides a shared infrastructure to be used by multiple tenants of multiple content distribution applications. Thus, it needs to provide both resource and data access control.

3.5.1. Resource Allocation

There are two primary interacting entities in the DECADE architecture. First, Storage Providers control where DECADE storage servers are provisioned and their total available resources. Second, Applications control data transfers amongst available DECADE servers and between DECADE servers and end-points. A form of isolation is required to enable concurrently-running Applications to each explicitly manage their own content and share of resources at the available servers.

The Storage Provider delegates the management of the resources at a DECADE server to one or more applications. Applications are able to explicitly and independently manage their own shares of resources.

3.5.2. User Delegations

Storage providers have the ability to explicitly manage the entities allowed to utilize the resources at a DECADE server. This capability is needed for reasons such as capacity-planning and legal considerations in certain deployment scenarios.

To provide a scalable way to manage applications granted resources at a DECADE server, we consider an architecture that adds a layer of indirection. Instead of granting resources to an application, the

DECADE server grants a share of the resources to a user. The user may in turn share the granted resources amongst multiple applications. The share of resources granted by a storage provider is called a User Delegation.

A User Delegation may be granted to an end-user (e.g., an ISP subscriber), a Content Provider, or an Application Provider. A particular instance of an application may make use of the storage resources:

- granted to the end-user (with the end-user's permission),
- granted to the Content Provider (with the Content Provider's permission), and/or
- granted to the Application Provider.

4. System Components

The primary focus of the current version of this document is on the architectural principles. The detailed system components will be discussed in the next document revision.

This section presents an overview of the components in the DECADE architecture.

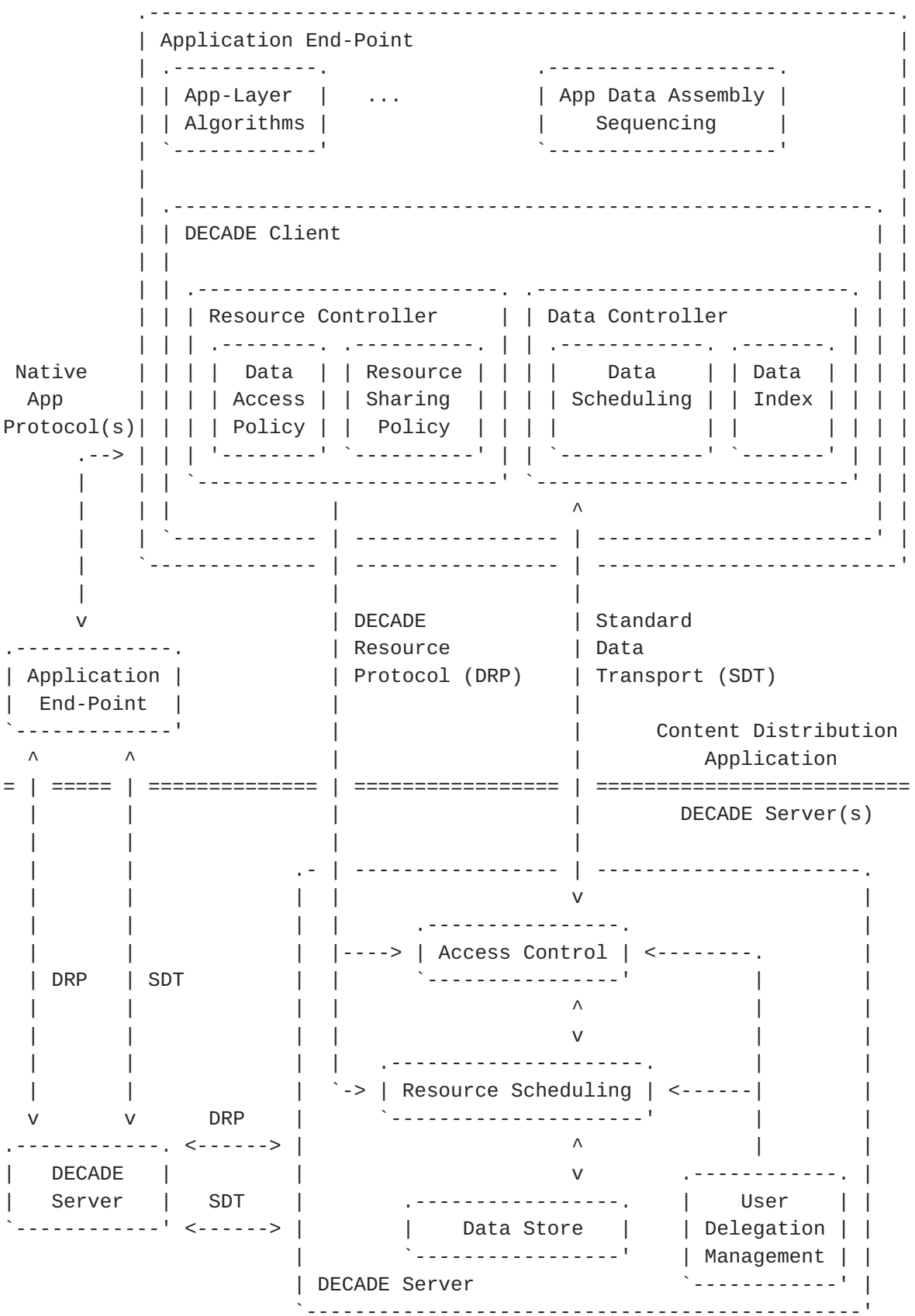

Application End-Point
App-Layer Algorithms
...
App Data Assembly Sequencing
DECADE Client
Resource Controller
Data Access Policy
Resource Sharing Policy
Data Controller
Data Scheduling
Data Index
Native App Protocol(s)
Application End-Point
DECADE Resource Protocol (DRP)
Standard Data Transport (SDT)
Content Distribution Application
DECADE Server(s)
Access Control
Resource Scheduling
DRP
SDT
DECADE Server
SDT
Data Store
User Delegation Management
DECADE Server

Figure 1: DECADE Architecture Components

A component diagram of the DECADE architecture is displayed in Figure 1. The diagram illustrates the major components of a Content Distribution Application related to DECADE, as well as the functional components of a DECADE Server.

To keep the scope narrow, we only discuss the primary components related to protocol development. Particular deployments may require additional components (e.g., monitoring and accounting at a DECADE server), but they are intentionally omitted from the current version of this document.

4.1. Content Distribution Application

Content Distribution Applications have many functional components. For example, many P2P applications have components to manage overlay topology management, piece selection, etc. In supporting DECADE, it may be advantageous to consider DECADE within some of these components. However, in this architecture document, we focus on the components directly employed to support DECADE.

4.1.1. Data Assembly

DECADE is primarily designed to support applications that can divide distributed contents into immutable data objects. To accomplish this, applications include a component responsible for creating the individual data objects before distribution and then re-assembling data objects at the Content Consumer. We call this component Application Data Assembly. The specific implementation is entirely decided by the application.

In producing and assembling the data objects, two important considerations are sequencing and naming. The DECADE architecture assumes that applications implement this functionality themselves.

For example, a Content Distribution Application might divide a single content (e.g., a finite-length file or a live stream) into multiple data objects with names of the form "CONTENT_ID:SEQUENCE_NUMBER" where CONTENT_ID identifies the particular content (e.g., a particular movie or TV channel distributed by the application), and SEQUENCE_NUMBER both identifies an individual data object and determines its order when a client reconstructs individual data objects into the full content.

4.1.2. Native Protocols

Applications may still use existing protocols. In particular, an application may reuse existing protocols primarily for control/signaling. However, an application may still retain its existing data transport protocols, in addition to DECADE as the data transport protocol. This can be important for applications that are designed to be highly robust (e.g., if DECADE servers are unavailable).

4.1.3. DECADE Client

An application may be modified to support DECADE. We call the layer providing the DECADE support to an application the DECADE Client. It is important to note that a DECADE Client need not be embedded into an application. It could be implemented alone, or could be integrated in other entities such as network devices themselves.

4.1.3.1. Resource Controller

Applications may have different Resource sharing policies and Data access policies to control their resource and data in DECADE servers. These policies can be existing policies of applications (e.g., tit-for-tat) or custom policies adapted for DECADE. The specific implementation is decided by the application.

4.1.3.2. Data Controller

DECADE is designed to decouple the control and the data transport of applications. Data transport between applications and DECADE servers uses standard data transport protocols. It may need to schedule the data being transferred according to network conditions, available DECADE Servers, and/or available DECADE Server resources. An index indicates data available at remote DECADE servers. The index (or a subset of it) may be advertised to other Application End-Points.

4.2. DECADE Server

DECADE server is an important functional component of DECADE. It stores data from Application End-Points, and provides control and access of those data to Application End-Points. Note that a DECADE server is not necessarily a single physical machine, it could also be implemented as a cluster of machines.

4.2.1. Access Control

An Application End-Point can access its own data or other Application End-Point's data (provided sufficient authorization) in DECADE servers. Application End-Points may also authorize other End-Points

to store data. If an access is authorized by an Application End-Point, the DECADE Server will provide access.

Note that even if a request is authorized, it may still fail to complete due to insufficient resources by either the requesting Application End-Point or the providing Application End-Point.

4.2.2. Resource Scheduling

Applications may apply their existing resource sharing policies or use a custom policy for DECADE. DECADE servers perform resource scheduling according to the resource sharing policies indicated by Application End-Points as well as configured User Delegations.

Access control and resource control are separated in DECADE server. It is possible that an Application End-Point provides only access to its data without any resources. In order to access this data, another Application End-Point may use the granted access along with its own available resources to store or retrieve data from a DECADE Server.

4.2.3. Data Store

Data from applications may be stored into disks. Data can be deleted from disks either explicitly or automatically (e.g., after a TTL). It may be possible to perform optimizations in certain cases, such as avoiding writing temporary data (e.g., live streaming) to a disk.

4.3. Protocols

The DECADE Architecture uses two protocols. First, the DECADE Resource Protocol is responsible for communicating access control and resource scheduling policies to the DECADE Server. Second, standard data transport protocols (e.g., WebDAV or NFS) are used to transfer data objects to and from a DECADE Server. The DECADE architecture will specify a small number of Standard Data Transport instances.

Decoupling the protocols in this way allows DECADE to both directly utilize existing standard data transports and to evolve independently.

It is also important to note that the two protocols do not need to be separate on the wire. For example, the DECADE Resource Protocol messages may be piggybacked within the extension fields provided by certain data transport protocols. However, this document considers them as two separate, logical functional components for clarity.

4.3.1. DECADE Resource Protocol

The DECADE Resource Protocol is responsible for communicating both access control and resource sharing policies to DECADE Servers used for data transport.

The DECADE architecture specification will provide exactly one DECADE Resource Protocol.

4.3.2. Standard Data Transports

Existing data transport protocols are used to read and write data from a DECADE Server. Protocols under consideration are WebDAV and NFS.

4.4. Data Sequencing and Naming

In order to provide a simple and generic interface, the DECADE Server is only responsible for storing and retrieving individual data objects. Furthermore, DECADE uses its own simple naming scheme that provides uniqueness (with high probability) between data objects, even across multiple applications.

4.4.1. DECADE Data Object Naming Schame

The name of a data object is derived from the hash over the data object's content (the raw bytes), which is made possible by the fact that DECADE objects are immutable. This scheme multiple appealing properties:

- Simple integrity verification
- Unique names (with high probability)
- Application independent, without a new IANA-maintained registry

The DECADE naming scheme also includes a "type" field, the "type" identifier indicates that the name is the hash of the data object's content and the particular hashing algorithm used. This allows the DECADE protocol to evolve by either changing the hashing algorithm (e.g., if security vulernabilities with an existing hashing algorithm are discovered), or move to a different naming scheme altogether.

The specific format of the name (e.g., encoding, hash algorithms, etc) is out of scope of this document, and left for protocol specification.

Another advantage of this scheme is that a DECADE client knows the

name of a data object before it is completely stored at the DECADE server. This allows for particular optimizations, such as advertising data object while the data object is being stored, removing store-and-forward delays. For example, a DECADE client A may simultaneously begin storing an object to a DECADE server, and advertise that the object is available to DECADE client B. If it is supported by the DECADE server, client B may begin downloading the object before A is finished storing the object.

4.4.2. Application Usage

Recall from Section 4.1.1 that an Application typically includes its own naming and sequencing scheme. It is important to note that the DECADE naming format does not attempt to replace any naming or sequencing of data objects already performed by an Application; instead, the DECADE naming is intended to apply only to data objects referenced at the DECADE layer.

DECADE names are not necessarily correlated with the naming or sequencing used by the Application using a DECADE client. The DECADE client is expected to maintain a mapping from its own data objects and their names to the DECADE data objects and names. Furthermore, the DECADE naming scheme implies no sequencing or grouping of objects, even if this is done at the application layer.

Not only does an Application retain its own naming scheme, it may also decide the sizes of data objects to be distributed via DECADE. This is desirable since sizes of data objects may impact Application performance (e.g., overhead vs. data distribution delay), and the particular tradeoff is application-dependent.

4.4.3. Application Usage Example

To illustrate these properties, this section presents multiple examples.

4.4.3.1. Application with Fixed-Size Chunks

Similar to the example in Section 4.1.1, consider an Application in which each individual application-layer segment of data is called a "chunk" and has a name of the form: "CONTENT_ID:SEQUENCE_NUMBER". Furthermore, assume that the application's native protocol uses chunks of size 16KB.

Now, assume that this application wishes to make use of DECADE, and assume that it wishes to store data to DECADE servers in data objects of size 64KB. To accomplish this, it can map a sequence of 4 chunks into a single DECADE object, as shown in Figure 2.

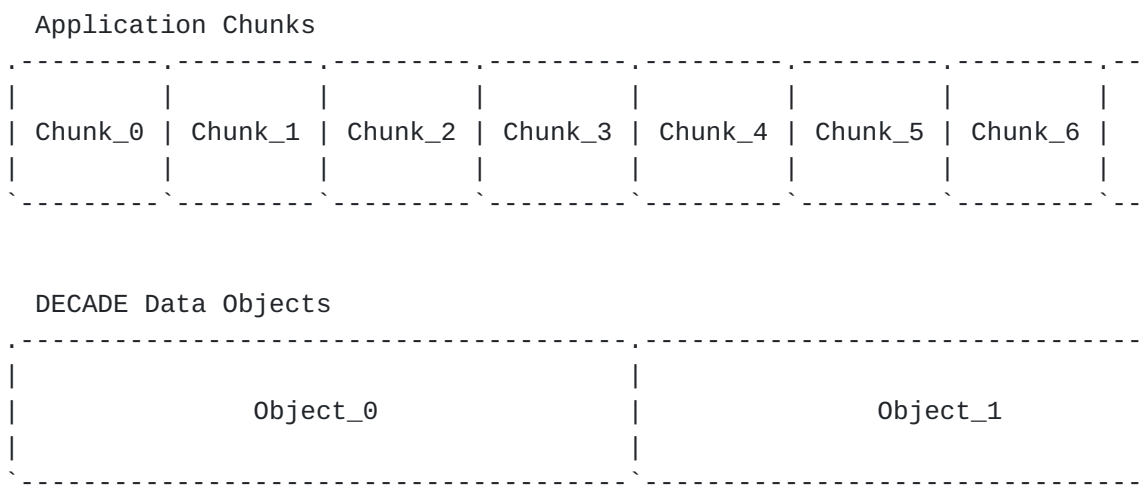


Figure 2: Mapping Application Chunks to DECADE Data Objects

In this example, the Application might maintain a logical mapping that is able to determine the name of a DECADE data object given the chunks contained within that data object. The name might be learned from either the original source, another endpoint with which the it is communicating, a tracker, etc.

It is important to note that as long as the data contained within each sequence of chunks is unique, the corresponding DECADE data objects have unique names. This is desired, and happens automatically if particular Application segments the same stream of data in a different way, including different chunk size sizes or different padding schemes.

4.4.3.2. Application with Continuous Streaming Data

Next, consider an Application whose native protocol retrieves a continuous data stream (e.g., an MPEG2 stream) instead of downloading and redistributing chunks of data. Such an application could segment the continuous data stream to produce either fixed-sized or variable-sized DECADE data objects.

Figure 3 shows how a video streaming application might produce variable-sized DECADE data objects such that each DECADE data object contains 10 seconds of video data.

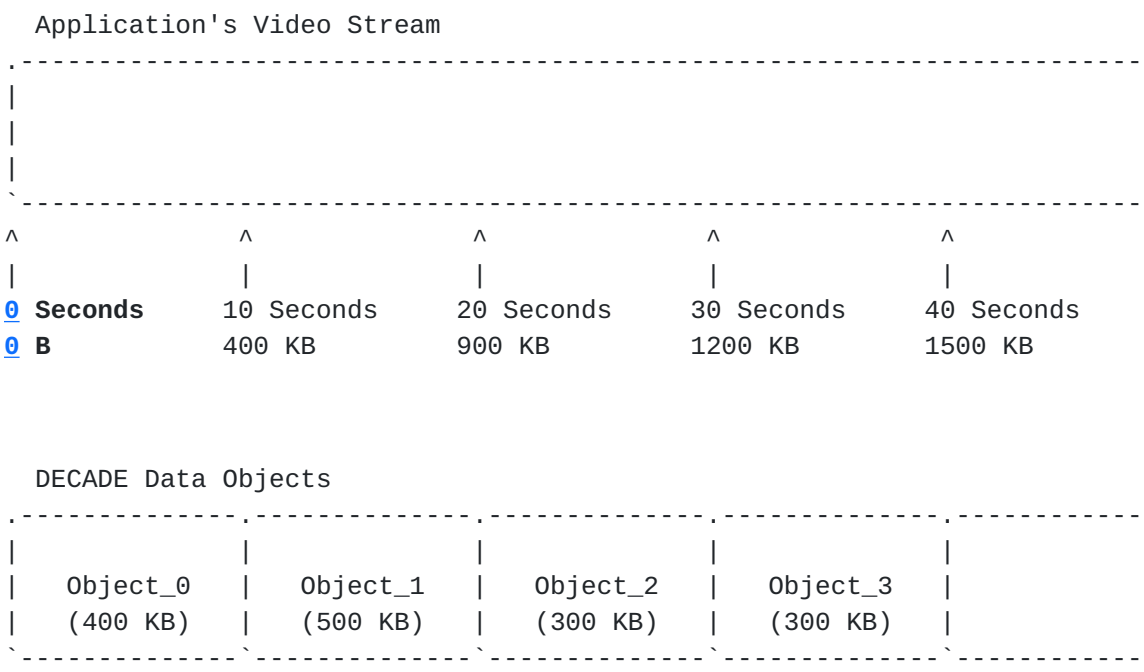


Figure 3: Mapping a Continuous Data Stream to DECADE Data Objects

Similar to the previous example, the Application might maintain a mapping that is able to determine the name of a DECADE data object given the time offset of the video chunk.

4.5. Token-based Authentication and Resource Control

A primary use case for DECADE is a DECADE Client authorizing other DECADE Clients to store or retrieve data objects from its DECADE storage. To support this, DECADE uses a token-based authentication scheme.

In particular, an entity trusted by a DECADE Client generates a digitally-signed token with particular properties (see Section 5.1.2 for details). The DECADE Client distributes this token to other DECADE Clients which then use the token when sending requests to the DECADE Server. Upon receiving a token, the DECADE Server validates the signature and the operation being performed.

This is a simple scheme, but has multiple important advantages over an alternate approach in which a DECADE Client explicitly manipulates an Access Control List (ACL) associated with each DECADE data object. In particular, it has the following advantages when applied to DECADE's target applications:

- Authorization policies are implemented within the Application; an Application explicitly controls when tokens are generated and to

whom they are distributed.

- Fine-grained access and resource control can be applied to data objects; see Section 5.1.2 for the list of restrictions that can be enforced with a token.

- There is no messaging between a DECADE Client and DECADE Server to manipulate data object permissions. This can simplify, in particular, Applications which share data objects with many dynamic peers and need to frequently adjust access control policies attached to DECADE data objects.

- Tokens can provide anonymous access, in which a DECADE Server does not need to know the identity of each DECADE Client that accesses it. This enables a DECADE Client to send tokens to DECADE Clients in other administrative or security domains, and allow them to read or write data objects from its DECADE storage.

It is important to note that, in addition to DECADE Clients applying access control policies to DECADE data objects, the DECADE Server may be configured to apply additional policies based on user, object, geographic location, etc. Defining such policies is out of scope of the DECADE Working Group, but in such a case, a DECADE Client may be denied access even though it possess a valid token.

4.6. In-Network Storage Components Mapped to DECADE Architecture

In this section we evaluate how the basic components of an in-network storage system identified in Section 3 of [I-D.ietf-decade-survey] map into the DECADE architecture.

It is important to note that complex and/or application-specific behavior is delegated to applications instead of tuning the storage system wherever possible.

4.6.1. Data Access Interface

Users can read and write objects of arbitrary size through the DECADE Client's Data Controller, making use of a standard data transport.

4.6.2. Data Management Operations

Users can move or delete previously stored objects via the DECADE Client's Data Controller, making use of a standard data transport.

4.6.3. Data Search Capability

Users can enumerate or search contents of DECADE servers to find objects matching desired criteria through services provided by the Content Distribution Application (e.g., buffer-map exchanges, a DHT, or peer-exchange). In doing so, End-Points may consult their local data index in the DECADE Client's Data Controller.

4.6.4. Access Control Authorization

All methods of access control are supported: public-unrestricted, public-restricted and private. Access Control Policies are generated by a Content Distribution Application and provided to the DECADE Client's Resource Controller. The DECADE Server is responsible for implementing the access control checks.

4.6.5. Resource Control Interface

Users can manage the resources (e.g. bandwidth) on the DECADE server that can be used by other Application End-Points. Resource Sharing Policies are generated by a Content Distribution Application and provided to the DECADE Client's Resource Controller. The DECADE Server is responsible for implementing the resource sharing policies.

4.6.6. Discovery Mechanism

This is outside the scope of the DECADE architecture. However, it is expected that DNS or some other well known protocol will be used for the users to discover the DECADE servers.

4.6.7. Storage Mode

DECADE Servers provide an object-based storage mode. Immutable data objects may be stored at a DECADE server. Applications may consider existing blocks as DECADE data objects, or they may adjust block sizes before storing in a DECADE server.

5. DECADE Protocols

This section specifies the DECADE Resource Protocol (DRP) and the Standard Data Transport (SDT) in terms of abstract protocol interactions that are intended to mapped to specific protocols. Note that while the protocols are logically separate, DRP is specified as being carried through extension fields within an SDT (e.g., HTTP headers).

The DRP is the protocol used by a DECADE client to configure the

resources and authorization used to satisfy requests (reading, writing, and management operations concerning DECADE objects) at a DECADE server. The SDT is used to send the operations to the DECADE server. Necessary DRP metadata is supplied using mechanisms in the SDT that are provided for extensibility (e.g., additional request parameters or extension headers).

5.1. DECADE Resource Protocol (DRP)

DRP provides configuration of access control and resource sharing policies on DECADE servers. A content distribution application, e.g., a live P2P streaming session, MAY employ several DECADE servers, for instance, servers in different operator domains, and DRP allows one instance of such an application, e.g., an application endpoint, to configure access control and resource sharing policies on a set of servers.

5.1.1. Controlled Resources

On a single DECADE server, the following resources may be managed:

communication resources: DECADE servers have limited communication resources in terms of bandwidth (upload/download) but also in terms of number of connected clients (connections) at a time.

storage resources: DECADE servers have limited storage resources.

5.1.2. Token-based Authentication and Resource Control

DECADE uses a token-based scheme that allows a DECADE Client to authorize other DECADE Clients to perform certain actions (e.g., read or write data objects) on the client's DECADE Server. The token includes the following fields:

Permitted operations (e.g., read, write)

Permitted objects (e.g., names of data objects that may be read or written)

Permitted clients (e.g., as indicated by IP address or other identifier) that may use the token

Expiration time

Priority for bandwidth given to requested operation

Amount of data that may be read or written

The particular format for the token is out of scope of this document.

The tokens are generated by a trusted entity at the request of a DECADE Client. It is out of scope of this document to identify which entity serves this purpose, but examples include the DECADE Client itself, a DECADE Server trusted by the DECADE Client, or another server managed by a Storage Provider trusted by the DECADE Client.

Upon generating a token, a DECADE Client may distribute it to another DECADE Client (e.g., via their native Application protocol). The receiving DECADE Client may then connect to the sending DECADE Client's DECADE Server and perform any operation permitted by the token. The token must be sent along with the operation. The DECADE Server validates the token to identify the DECADE Client that issued it and whether the requested operation is permitted by the contents of the token. If the token is successfully validated, the DECADE Server applies the resource control policies indicated in the token while performing the operation.

It is possible for DRP to allow tokens to apply to a batch of operations to reduce communication overhead required between DECADE Clients.

DRP may also define tokens to include a unique identifer to allow a DECADE Server to detect when a token is used multiple times.

5.1.3. Status Information

DRP provides a request service for status information that DECADE clients can use to request information from a DECADE server.

status information per application context on a specific server:
Access to such status information requires client authorization, i.e., DECADE clients need to be authorized to access status information for a specific application context. This authorization (and the mapping to application contexts) is based on the user delegation concept as described in Section 3.5. The following status information elements can be obtained:

* list of associated objects (with properties)

* resources used/available

* list of servers to which objects have been distributed (in a certain time-frame)

* list of clients to which objects have been distributed (in a certain time-frame)

For the list of servers/clients to which objects have been distributed to, the DECADE server can decide on time bounds for which this information is stored and specify the corresponding time frame in the response to such requests. Some of this information can be used for accounting purposes, e.g., the list of clients to which objects have been distributed.

access information per application context on a specific server:
Access information can be provided for accounting purposes, for example, when application service providers are interested to maintain access statistics for resources and/or to perform accounting per user. Again, access to such information requires client authorization based on the user delegation concept as described in Section 3.5. The following access information elements can be requested:

* what objects have been accessed how many times

* access tokens that a server as seen for a given object

The DECADE server can decide on time bounds for which this information is stored and specify the corresponding time frame in the response to such requests.

5.1.4. Object Properties

Objects that are stored on a DECADE server can provide properties (in addition to the object identifier and the actual content). Depending on authorization, DECADE clients may get or set such properties. This authorization (and the mapping to application contexts) is based on the user delegation concept as described in Section 3.5. The DECADE architecture does not limit the set of permissible properties, but rather specifies a set of baseline properties that SHOULD be supported by implementations.

TTL: TTL of the object as an absolute time value

object size: in bytes

MIME type

access statistics: how often the object has been accessed (and what tokens have been used)

5.2. Standard Data Transport (SDT)

A DECADE server provide a data access interface, and SDT is used to write objects to a server and to read (download) objects from a server. Semantically, SDT is a client-server protocol, i.e., the DECADE server always responds to client requests.

5.2.1. Writing/Uploading Objects

For writing objects, a client uploads an object to a DECADE server. The object on the server will be named (associated to an identifier), and this name can be used to access (download) the object later, e.g., the client can pass the name as a reference to other client that can then refer to the object.

DECADE objects can be self-contained objects such as multimedia resources, files etc., but also chunks, such as chunks of a P2P distribution protocol that can be part of a containing object or a stream.

A server MUST accept download requests for an object that is still being uploaded.

The application that originates the objects MUST generate DECADE object names according to the naming specification in Section 4.4. The naming scheme provides that the name is unique. DECADE clients (as parts of application entities) upload a named object to a server, and a DECADE server MUST not change the name. It MUST be possible for downloading clients, to access the object using the original name. A DECADE server MAY verify the integrity and other security properties of uploaded objects.

In the following we provide an abstract specification of the upload operation that we name 'PUT METHOD'. See Appendix A.1 for an example how this could be mapped to HTTP.

Method PUT:

Parameters:

NAME: The naming of the object according to Section 4.4

OBJECT: The object itself. The protocol MUST provide transparent binary object transport.

Description: The PUT method is used by a DECADE client to upload an object with an associated name 'NAME' to a DECADE server.

RESPONSES: The DECADE server MUST respond with one the following response messages:

OK: The object has been uploaded successfully and has replaced an existing object with the same name.

CREATED: The object has been uploaded successfully and is now available under the specified name.

ERRORs: possible error codes later will be specified in a later version of this document

5.2.2. Downloading Objects

A DECADE client can request named objects from a DECADE server. In the following, we provide an abstract specification of the download operation that we name 'GET METHOD'. See Section 4.4 for an example how this could be mapped to HTTP.

Method GET:

Parameters:

NAME: The naming of the object according to Section 4.4.

Description: The GET method is used by a DECADE client to download an object with an associated name 'NAME' from a DECADE server.

RESPONSES: The DECADE server MUST respond with one the following response messages:

OK: The request has succeeded, and an entity corresponding to the requested resource is sent in the response.

ERRORs:

NOTFOUND: The DECADE server has not found anything matching the request object name.

Other Errors: TBD in a future version of this document

6. Server-to-Server Protocols

An important feature of DECADE is the capability for one DECADE server to directly download data objects from another DECADE server. This capability allows Applications to directly replicate data objects between servers without requiring end-hosts to use uplink capacity to upload data objects to a different DECADE server. Similar to other operations in DRP and SDT, replicating data objects between DECADE servers is an explicit operation.

To support this functionality, DECADE re-uses the already-specified protocols to support operations directly between servers. DECADE servers are not assumed to trust each other nor are configured to do so. All data operations are performed on behalf of DECADE clients via explicit instruction, so additional capabilities are needed in the DECADE client-server protocols DECADE clients must be able to indicate to a DECADE server the following additional parameters:

- which remote DECADE server(s) to access;
- the operation to be performed (PUT or GET); and
- Credentials indicating permission to perform the operation at the remote DECADE server.

In this way, a DECADE server is also a DECADE client, and requests may instantiate requests via that client. The operations are performed as if the original requestor had its own DECADE client co-located with the DECADE server. It is this mode of operation that provides substantial savings in uplink capacity.

6.1. Operational Overview

DECADE's server-to-server support is focused on reading and writing data objects between DECADE servers. A DECADE GET or PUT request MAY supply the following additional parameters:

REMOTE_SERVER: Address of the remote DECADE server. The format of the address is out-of-scope of this document.

REMOTE_USER: The account at the remote server from which to retrieve the object (for a GET), or in which the object is to be stored (for a PUT).

TOKEN: Credentials to be used at the remote server.

These parameters are used by the DECADE server to instantiate a request to the specified remote server. It is assumed that the data

object referred to at the remote server is the same as the original request. It is also assumed that the operation performed at the remote server is the same as the operation in the original request. Though explicitly supplying these may provide additional freedom, it is not clear what benefit they might provide.

Note that when a DECADE client invokes a request a DECADE server with these additional parameters, it is giving the DECADE server permission to act on its behalf. Thus, it would be wise for the supplied token to have narrow privileges (e.g., limited to only the necessary data objects) or validity time (e.g., a small expiration time).

In the case of a GET operation, the DECADE server is to retrieve the data object from the remote server using the specified credentials (via a GET request to the remote server), and then return the object to the client. In the case of a PUT operation, the DECADE server is to store the object from the client, and then store the object to the remote server using the specified credentials (via a PUT request to the remote server).

7. Potential Optimizations

As suggestions for the protocol design and eventual implementations, we discuss particular optimizations that are enabled by the DECADE Architecture discussed in this document.

7.1. Pipelining to Avoid Store-and-Forward Delays

DECADE server may choose to not fully store an object before beginning to serve it. For example, in the case of a GET request, a DECADE server may begin to receive a data object from a remote server or DECADE Client, and immediately begin returning it to the DECADE client. This pipelining mode avoids store-and-forward delays, which could be substantial for large objects. A similar behavior could be used for PUT.

7.2. Deduplication

A common concern amongst Storage Providers is the total volume of data that needs to be stored. An optimization frequently applied in existing storage systems is de-duplication techniques which attempt to avoid storing identical data multiple times. DECADE Server implementations may internally perform de-duplication of data on disk, but the DECADE architecture enables other forms of de-duplication.

Note that these techniques may impact protocol design. Discussion of whether or not they should be adopted is out of scope of this document.

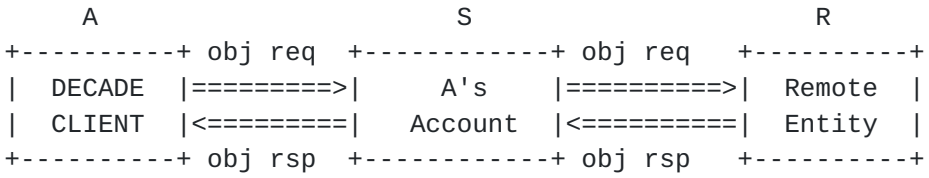
7.2.1. Traffic Deduplication

7.2.1.1. Rationale

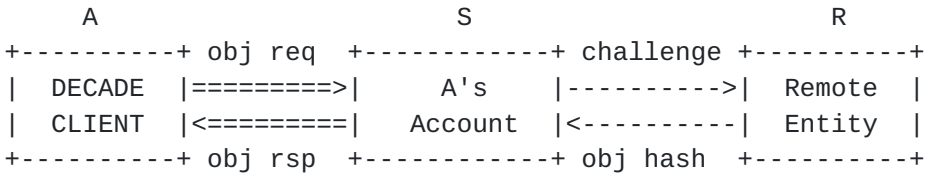
When a DECADE client (A) indicates its DECADE account on a DECADE server (S) to fetch an object from a remote entity (R) (a DECADE server or DECADE client) and if the object is already stored locally in S, S may perform Traffic Deduplication. This means that S does not download the object from R, which saves network traffic. Instead, it performs a challenge to make sure that the remote entity R actually has the object and then replies with its local object copy directly.

7.2.1.2. Example

As shown in Figure 4 , without Traffic Deduplication, redundant traffic flows between S and R will be issued if the server already has the object requested by A. If Traffic Deduplication is enabled, S only needs to challenge R to verify that it does have the data to avoid data-stealing attacks.



(a) Without Traffic Deduplication



(b) With Traffic Deduplication

Figure 4

7.2.1.3. HTTP Compatibility of Challenge

How to integrate traffic deduplication with HTTP is shown in Appendix A.1.3.

7.2.2. Cross-Server Storage Deduplication

The same object might be uploaded for multiple times to different DECADE servers. For storage efficiency, storage providers may desire a single object to be stored on one or a few servers. They might design internal system architecture to achieve that, or simply redirect the requests to proper servers. DECADE protocol support redirections of DECADE client request to support further cross-server storage deduplication.

8. Security Considerations

This document currently does not contain any security considerations beyond those mentioned in [I-D.ietf-decade-problem-statement].

9. IANA Considerations

This document does not have any IANA considerations.

10. Informative References

[RFC2616] 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.

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

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

[RFC4709] Reschke, J., "Mounting Web Distributed Authoring and Versioning (WebDAV) Servers", RFC 4709, October 2006.

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

[I-D.ietf-decade-problem-statement] Song, H., Zong, N., Yang, Y., and R. Alimi, "DECoupled Application Data Enroute (DECADE) Problem Statement", draft-ietf-decade-problem-statement-03 (work in progress), March 2011.

[I-D.ietf-decade-survey]
Alimi, R., Rahman, A., and Y. Yang, "A Survey of In-network Storage Systems", draft-ietf-decade-survey-04 (work in progress), March 2011.

[I-D.ietf-decade-reqs]
Yingjie, G., Bryan, D., Yang, Y., and R. Alimi, "DECADE Requirements", draft-ietf-decade-reqs-02 (work in progress), May 2011.

[GoogleStorageDevGuide]
"Google Storage Developer Guide", <http://code.google.com/apis/storage/docs/developer-guide.html>.

Appendix A. Appendix: Evaluation of Some Candidate Existing Protocols for DECADE DRP and SDT

In this section we evaluate how well the abstract protocol interactions specified in Section 5 for DECADE DRP and SDT can be fulfilled by existing protocols such as HTTP and WEBDAV.

A.1. HTTP

HTTP [RFC2616] is a key protocol for the Internet in general and especially for the World Wide Web. HTTP is a request-response protocol. A typical transaction involves a client (e.g. web browser) requesting content (resources) from a web server. Another example is when a client stores or deletes content from a server.

A.1.1. HTTP Support for DECADE Resource Protocol Primitives

DRP provides configuration of access control and resource sharing policies on DECADE servers.

A.1.1.1. Access Control Primitives

Access control requires mechanisms for defining the access policies for the server, and then checking the authorization of a user before it stores or retrieves content. HTTP supports a rudimentary access control via "HTTP Secure" (HTTPS). HTTPS is a combination of HTTP with SSL/TLS. The main use of HTTPS is to authenticate the server and encrypt all traffic between the client and the server. There is also a mode to support client authentication though this is less frequently used.

A.1.1.2. Communication Resource Controls Primitives

Communications resources include bandwidth (upload/download) and number of simultaneous connected clients (connections). HTTP supports bandwidth control indirectly through "persistent" HTTP connections. Persistent HTTP connections allows a client to keep open the underlying TCP connection to the server to allow streaming and pipelining (multiple simultaneous requests for a given client).

HTTP does not define protocol operation to allow limiting the communciation resources to a client. However servers typically perform this function via implementation algorithms.

A.1.1.3. Storage Resource Control Primitives

Storage resources include amount of memory and lifetime of storage. HTTP does not allow direct control of storage at the server end point. However HTTP supports caching at intermediate points such as a web proxy. For this purpose, HTTP defines cache control mechanisms that define how long and in what situations the intermediate point may store and use the content.

A.1.2. HTTP Support for DECADE Standard Transport Protocol Primitives

SDT is used to write objects and read (download) objects from a DECADE server. The object can be either a self-contained object such as a multimedia file or a chunk from a P2P system.

A.1.2.1. Writing Primitives

Writing involves uploading objects to the server. HTTP supports two methods of writing called PUT and POST. In HTTP the object is called a resource and is identified by a URI. PUT uploads a resource to a specific location on the server. POST, on the other hand, submits the object to the server and the server decides whether to update an existing resource or to create a new resource.

For DECADE, the choice of whether to use PUT or POST will be influenced by which entity is responsible for the naming. If the client performs the naming, then PUT is appropriate. If the server performs the naming, then POST should be used (to allow the server to define the URI).

A.1.2.2. Downloading Primitives

Downloading involves fetching of an object from the server. HTTP supports downloading through the GET and HEAD methods. GET fetches a specific resource as identified by the URL. HEAD is similiar but

only fetches the metadata ("header") associated with the resource but not the resource itself.

A.1.3. Traffic Deduplication Primitives

To challenge a remote entity for an object, the DECADE server should provide a seed number, which is generated by the server randomly, and ask the remote entity to return a hash calculated from the seed number and the content of the object. The server MAY also specify the hash function which the remote entity should use. HTTP supports the challenge message through the GET methods. The message type ("challenge"), the seed number and the hash funtion name are put in URL. In the reply, the hash is sent in an ETAG header.

A.1.4. Other Operations

HTTP supports deleting of content on the server through the DELETE method.

A.1.5. Conclusions

HTTP can provide a rudimentary DRP and SDT for some aspects of DECADE, but will not be able to satisfy all the DECADE requirements. For example, HTTP does not provide a complete access control mechanism, nor does it support storage resource controls at the end point server.

It is possible, however, to envision combining HTTP with a custom suite of other protocols to fulfill most of the DECADE requirements for DRP and SDT. For example, Google Storage for Developers is built using HTTP (with extensive proprietary extensions such as custom HTTP headers). Google Storage also uses OAUTH 2.0 (for access control) in combination with HTTP [GoogleStorageDevGuide].

A.2. WEBDAV

WebDAV [RFC4918] is a protocol for enhanced Web content creation and management. It was developed as an extension to HTTP Appendix A.1. WebDAV supports traditional operations for reading/writing from storage, as well as more advanced features such as locking and namespace management which are important when multiple users collaborate to author or edit a set of documents. HTTP is a subset of WebDAV functionality. Therefore, all the points noted above in Appendix A.1 apply implicitly to WebDAV as well.

A.2.1. WEBDAV Support for DECADE Resource Protocol Primitives

DRP provides configuration of access control and resource sharing policies on DECADE servers.

A.2.1.1. Access Control Primitives

Access control requires mechanisms for defining the access policies for the server, and then checking the authorization of a user before it stores or retrieves content. WebDAV has an Access Control Protocol defined in [RFC3744].

The goal of WebDAV access control is to provide an interoperable mechanism for handling discretionary access control for content and metadata managed by WebDAV servers. WebDAV defines an Access Control List (ACL) per resource. An ACL contains a set of Access Control Entries (ACEs), where each ACE specifies a principal (i.e. user or group of users) and a set of privileges that are granted to that principal. When a principal tries to perform an operation on that resource, the server evaluates the ACEs in the ACL to determine if the principal has permission for that operation.

WebDAV also requires that an authentication mechanism be available for the server to validate the identity of a principal. As a minimum, all WebDAV compliant implementations are required to support HTTP Digest Authentication.

A.2.1.2. Communication Resource Controls Primitives

Communications resources include bandwidth (upload/download) and number of simultaneous connected clients (connections). WebDAV supports communication resource control as described in Appendix A.1.1.2.

A.2.1.3. Storage Resource Control Primitives

Storage resources include amount of memory and lifetime of storage. WebDAV supports the concept of properties (which are metadata for a resource). A property is either "live" or "dead". Live properties include cases where a) the value of a property is protected and maintained by the server, and b) the value of the property is maintained by the client, but the server performs syntax checking on submitted values. A dead property has its syntax and semantics enforced by the client; the server merely records the value of the property.

WebDAV supports a list of standardized properties [RFC4918] that are useful for storage resource control. These include the self-

explanatory "creationdate", and "getcontentlength" properties. There is also an operation called PROPFIND to retrieve all the properties defined for the requested URI.

WebDAV also has a Quota and Size Properties mechanism defined in [RFC4331] that can be used for storage control. Specifically, two key properties are defined per resource: "quota-available-bytes" and "quota-used-bytes".

WebDAV does not define protocol operation for storage resource control. However servers typically perform this function via implementation algorithms in conjunction with the storage related properties discussed above.

A.2.2. WebDAV Support for DECADE Standard Transport Protocol Primitives

SDT is used to write objects and read (download) objects from a DECADE server. The object can be either a self-contained object such as a multimedia file or a chunk from a P2P system.

A.2.2.1. Writing Primitives

Writing involves uploading objects to the server. WebDAV supports PUT and POST as described in Appendix A.1.2.1. WebDAV LOCK/UNLOCK functionality is not needed as DECADE assumes immutable data objects. Therefore, resources cannot be edited and so do not need to be locked. This approach should help to greatly simplify DECADE implementations as the LOCK/UNLOCK functionality is quite complex.

A.2.2.2. Downloading Primitives

Downloading involves fetching of an object from the server. WebDAV supports GET and HEAD as described in Appendix A.1.2.2. WebDAV LOCK/UNLOCK functionality is not needed as DECADE assumes immutable data objects.

A.2.3. Other Operations

WebDAV supports DELETE as described in Appendix A.1.4. In addition WebDAV supports COPY and MOVE methods. The COPY operation creates a duplicate of the source resource identified by the Request-URI, in the destination resource identified by the URI in the Destination header.

The MOVE operation on a resource is the logical equivalent of a COPY, followed by consistency maintenance processing, followed by a delete of the source, where all three actions are performed in a single operation. The consistency maintenance step allows the server to

perform updates caused by the move, such as updating all URLs, other than the Request-URI that identifiesthe source resource, to point to the new destination resource.

WebDAV also supports the concept of "collections" of resources to support joint operations on related objects (e.g. file system directories) within a server's namespace. For example, GET and HEAD may be done on a single resource (as in HTTP) or on a collection. The MKCOL operation is used to create a new collection. DECADE may find the concept of collections to be useful if there is a need to support directory like structures in DECADE.

WebDAV servers can be interfaced from an HTML-based user interface in a web browser. However, it is frequently desirable to be able to switch from an HTML-based view to a persentation provided by a native WebDAV client, directly supporting WebDAV features. The method to perform this in a platform-neutral mechanism is specified in the WebDAV protocol for "mounting WebDAV servers" [RFC4709]. This type of feature may also be attractive for DECADE clients.

A.2.4. Conclusions

WebDAV has a rich array of features that can provide a good base for DRP and SDT for DECADE. An initial analysis finds that the following WebDAV features will be useful for DECADE:

- access control
- properties (and PROPFIND operation)
- COPY/MOVE operations
- collections
- mounting WebDAV servers

It is recommended that the following WebDAV features NOT be used for DECADE:

- LOCK/UNLOCK

Finally, some extensions to WebDAV may still be required to meet all DECADE requirements. For example, defining a new WebDAV "time-to-live" property may be useful for DECADE. Further analysis is required to fully define the potential extensions to WebDAV to meet all DECADE requirements.

Authors' Addresses

Richard Alimi
Google

Email: ralimi@google.com

Y. Richard Yang
Yale University

Email: yry@cs.yale.edu

Akbar Rahman
InterDigital Communications, LLC

Email: akbar.rahman@interdigital.com

Dirk Kutscher
NEC

Email: dirk.kutscher@neclab.eu

Hongqiang Liu
Yale University

Email: hongqiang.liu@yale.edu