DECADE
Internet-Draft
Intended status: Informational
Expires: May 3, 2012

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

October 31, 2011

DECADE Architecture
draft-ietf-decade-arch-04

Abstract

Content Distribution Applications (e.g., P2P applications) are widely used on the Internet and make up a large portion of the traffic in many networks. One technique to improve the network efficiency of these applications is to introduce storage capabilities within the networks; this is the capability to be provided by DECADE (DECoupled Application Data Enroute). This document presents an architecture for DECADE, discusses the underlying principles, and identifies core components and protocols for introducing in-network storage for these applications.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of this Memo

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

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

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

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

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

This Internet-Draft will expire on May 3, 2012.

Copyright Notice

Table of Contents

1. Introduction 5
2. Functional Entities 6
2.1. DECADE Server 6
2.2. DECADE Client 6
2.3. DECADE Storage Provider 6
2.4. Content Provider Using DECADE 6
2.5. Content Consumer Using DECADE 7
2.6. Content Distribution Application 7
3. Protocol Flow 7
3.1. Overview 7
3.2. An Example 9
4. Architectural Principles 9
4.1. Decoupled Control/Metadata and Data Planes 10
4.2. Immutable Data Objects 11
4.3. Data Object Identifiers 12
4.4. Explicit Control 12
4.5. Resource and Data Access Control through User Delegation 12
5. System Components 13
5.1. Content Distribution Application 14
5.2. DECADE Server 16
5.3. Data Sequencing and Naming 18
5.4. Token-based Authentication and Resource Control 21
5.5. Discovery 22
6. DECADE Protocols 23
6.1. DECADE Resource Protocol (DRP) 23
6.2. Standard Data Transport (SDT) 26
7. Server-to-Server Protocols 29
7.1. Operational Overview 29
8. Potential Optimizations 30
8.1. Pipelining to Avoid Store-and-Forward Delays 30
8.2. Deduplication 31
9. Security Considerations 32
10. IANA Considerations 33
11. Acknowledgments 33
12. References 33
12.1. Normative References 33
12.2. Informative References 33
Appendix A. Appendix: Evaluation of Some Candidate Existing Protocols for DECADE DRP and SDT 34
A.1. HTTP 35
A.2. WEBDAV 37
A.3. CDMI 40
Appendix B. In-Network Storage Components Mapped to DECADE Architecture 42
B.1. Data Access Interface 43

B.2. Data Management Operations 43
B.3. Data Search Capability 43
B.4. Access Control Authorization 43
B.5. Resource Control Interface 43
B.6. Discovery Mechanism 43
B.7. Storage Mode 44
Authors' Addresses 44

1. Introduction

Content Distribution Applications are widely used on the Internet today to distribute data, and they contribute a large portion of the traffic in many networks. The DECADE architecture described in this document enables such applications to leverage in-network storage to achieve more efficient content distribution. Specifically, in many subscriber networks, it can be 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 (Digital Subscriber Line Access Multiplexers) and CMTSs (Cable Modem Termination Systems) in remote locations. Therefore, 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 an architecture for 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. See [I-D.ietf-decade-reqs] for a definition of the target applications supported by 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. Focusing on only core functionalities, DECADE may be simpler and easier to support by service providers. If more complex functionalities are needed by a certain application or class of applications, it may be layered on top of DECADE.

DECADE will leverage existing data transport and application-layer protocols. The design is to work with a small set of alternative IETF protocols. In this document, we use "data transport" to refer to a protocol that is used to read data from and write data into DECADE in-network storage.

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

This document does not define any new protocol. Instead, when identifying the need for a new protocol, it presents an abstract design including the necessary functions of that protocol (including

rationale) in order to guide future protocol development.

2. Functional Entities

This section defines the functional entities involved in a DECADE system. Functional entities can be classified as follows:

- A physical or logical component in the DECADE architecture: DECADE Client, DECADE Server, Content Distribution Application and Application End Point;
- Operator of a physical or logical component in the DECADE architecture: DECADE Storage Provider; and
- Source or sink of content distributed via the DECADE architecture: DECADE Content Provider, and DECADE Content Consumer.

2.1. DECADE Server

A DECADE server stores DECADE data inside the network, and thereafter manages both the stored data and access to that data. To reinforce that these servers are responsible for storage of raw data, this document also refers to them as storage servers.

2.2. DECADE Client

A DECADE client stores and retrieves data at DECADE Servers.

2.3. DECADE Storage Provider

A DECADE storage provider deploys and/or manages DECADE storage server(s) within a network. A storage provider may also own or manage the network in which the DECADE servers are deployed, but this is not mandatory.

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.

2.4. Content Provider Using DECADE

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

2.5. Content Consumer Using DECADE

A content consumer using DECADE accesses DECADE storage servers (by way of a DECADE client). A content consumer can access one or more DECADE storage servers. A content consumer may be a single process or a distributed application (e.g., in a P2P scenario), and may either be fixed or mobile. An instance of a distributed application, such as a P2P application, may both provide content to and consume content from DECADE storage servers.

2.6. Content Distribution Application

A content distribution application (as a target application for DECADE as described in [I-D.ietf-decade-reqs]) is a distributed application designed for dissemination of a possibly-large data set to multiple consumers. Content Distribution Applications typically divide content into smaller blocks for dissemination.

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

2.6.1. 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. For example, an Application End-Point may be an instance of a video streaming client, or it may be the source providing the video to a set of clients.

An Application End-Point need not be actively transferring data with other Application End-Points to interact with the DECADE storage system. That is, an End-Point may interact with the DECADE storage servers as an offline activity.

3. Protocol Flow

3.1. Overview

The DECADE Architecture uses two protocols, as shown in Figure 1. First, the DECADE Resource Protocol is responsible for communication of access control and resource scheduling policies from DECADE Client to DECADE Server, as well as between DECADE Servers. The DECADE Architecture includes exactly one DRP for interoperability and a common format through which these policies can be communicated.

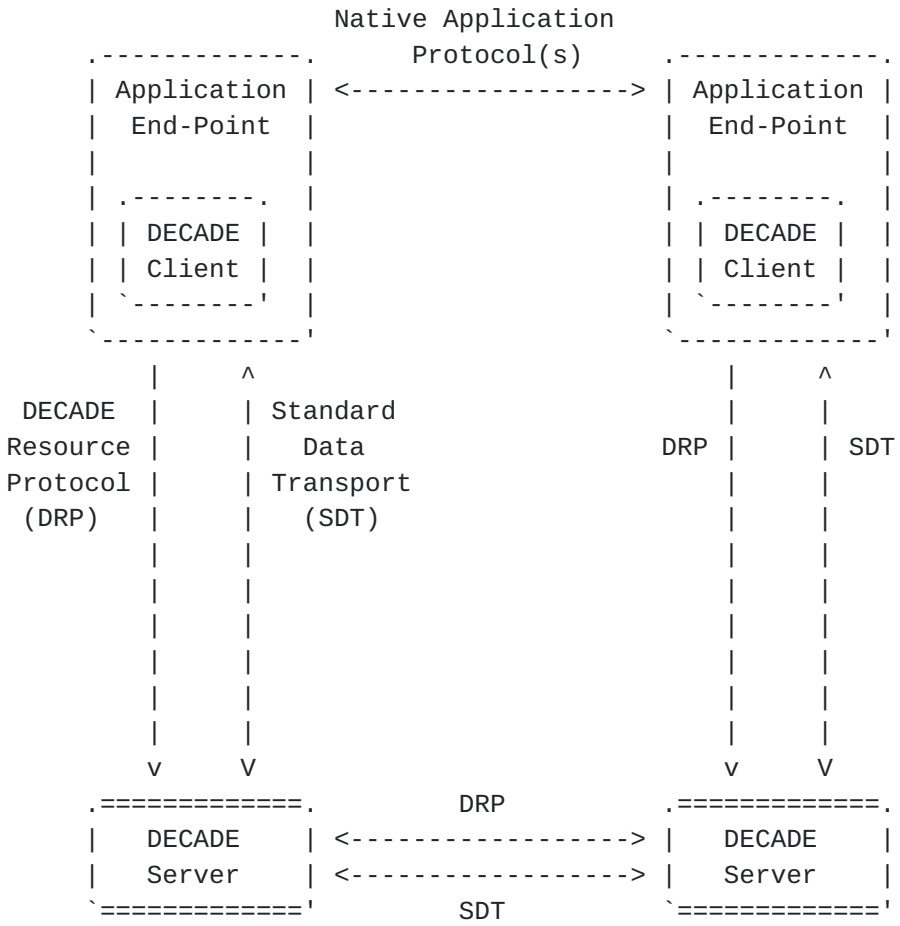


Figure 1: Generic Protocol Flow

Second, Standard Data Transport protocols (e.g., HTTP, WebDAV, or CDMI) are used to transfer data objects to and from a DECADE Server. The DECADE architecture may be used with multiple standard data transports.

Decoupling the protocols in this way allows DECADE to directly utilize existing standard data transports, as well as allowing both DECADE and DRP to evolve independently from data transports.

It is also important to note that the two protocols do not need to be separate on the wire. For example, DRP messages may be piggybacked within some extension fields provided by certain data transport protocols. In such a scenario, DRP is technically a data structure (transported by other protocols), but it can still be considered as a logical protocol that provides the services of configuring DECADE resource usage. Hence, this document considers SDT and DRP as two separate, logical functional components for clarity.

3.2. An Example

Before discussing details of the architecture, this section provides an example data transfer scenario to illustrate how the DECADE Architecture can be applied.

In this example, we assume that Application End-Point B (the receiver) is requesting a data object from Application End-Point A (the sender). Let S(A) denote A's DECADE storage server. There are multiple usage scenarios (by choice of the Content Distribution Application). For simplicity of introduction, we design the example to use only a single DECADE Server; Section 7 details a case when both A and B wish to employ DECADE Servers.

When an Application End-Point wishes to use its DECADE storage server, it provides a token (see Section 6.1.2 for details) to the other Application End-Point. The token is sent using the Content Distribution Application's native protocol.

The steps of the example are illustrated in Figure 2. First, B requests a data object from A using their native protocol. Next, A uses the DECADE Resource Protocol (DRP) to obtain a token from its DECADE storage server, S(A). A then provides the token to B (again, using their native protocol). Finally, B provides the token to S(A) via DRP, and requests and downloads the data object via a Standard Data Transport (SDT).

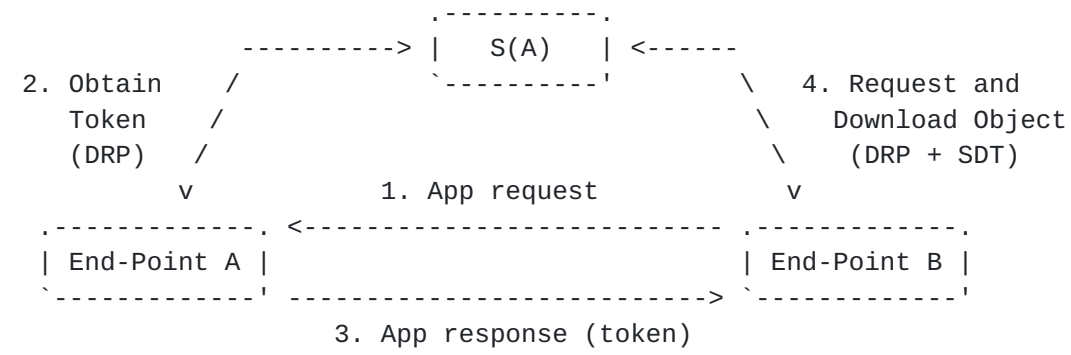


Figure 2: Download from Storage Server

4. Architectural Principles

We identify the following key principles.

4.1. 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. An observation of DECADE is that different content distribution applications can have unique considerations designing the control and signaling functions:

- Metadata Management Scheme: 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.

- Resource Scheduling Algorithms: 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.

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 [GoogleFileSystem] 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). NFSv4.1's pNFS extension [RFC5661] 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.

4.2. 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 using the aforementioned data model include P2P streaming (live and video-on-demand) and P2P file-sharing. 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.

Throughout this document, all the data objects/blocks are referred as immutable data objects/blocks.

4.3. 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.

4.4. 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).

4.5. 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.

4.5.1. 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 its 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.

4.5.2. 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.

As a simple example, a DECADE Server operated by an ISP may be configured to grant each ISP Subscriber 1.5 Mbps of bandwidth. The ISP Subscriber may in turn divide this share of resources amongst a video streaming application and file-sharing application which are running concurrently.

In general, 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.

5. System Components

The primary focus of this document is the architectural principals and the system components that implement them. While certain system components might differ amongst implementations, the document details the major components and their overall roles in the architecture.

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 this document.

5.1. Content Distribution Application

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

Figure 3 illustrates the components discussed in this section from the perspective of a single Application End-Point and their relation to DECADE.

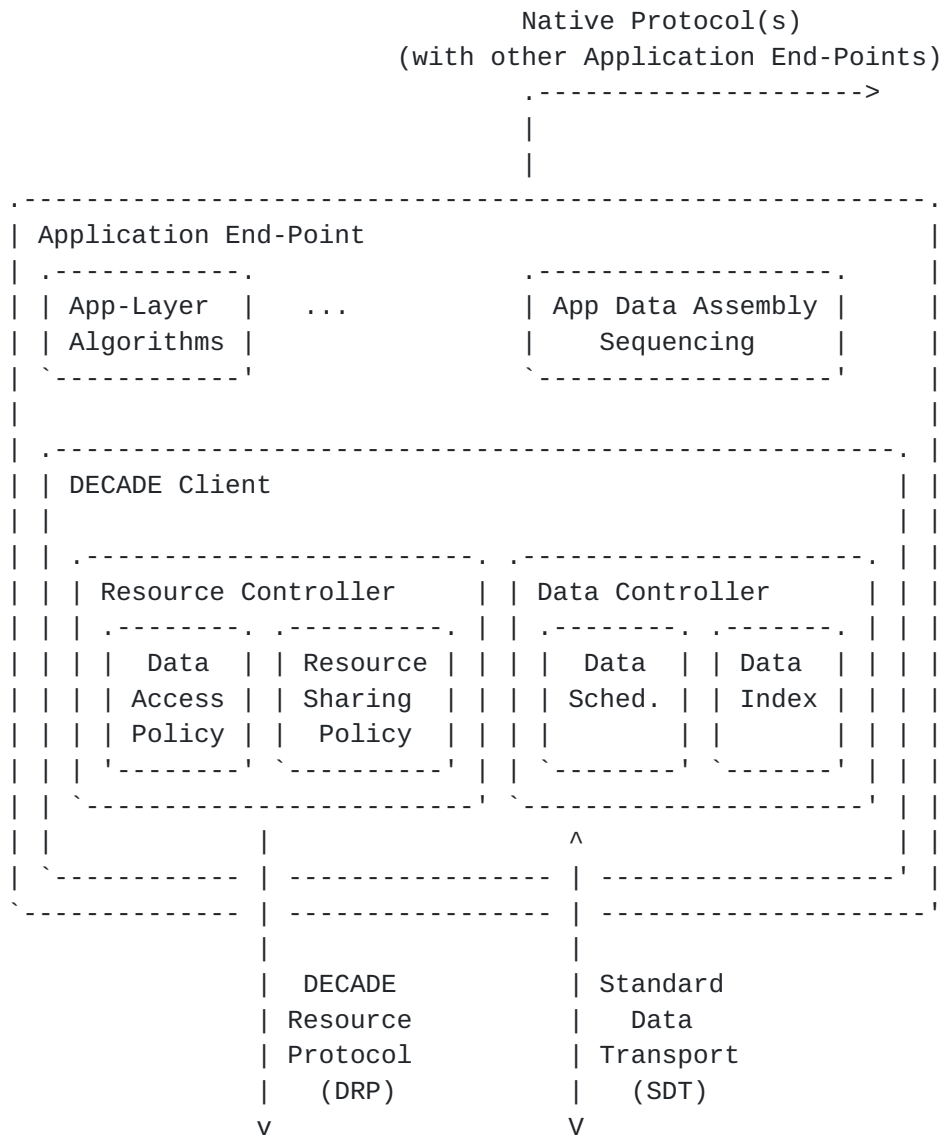


Figure 3: Application Components

5.1.1. Data Assembly

DECADE is primarily designed to support applications that can divide distributed contents into 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.

See Section 5.3 for further discussion.

5.1.2. 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).

5.1.3. 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.

5.1.3.1. 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.

5.1.3.2. 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. A Data Scheduling component schedules data transfers according to network conditions, available DECADE Servers, and/or available DECADE Server resources. The Data Index indicates data available at remote DECADE servers. The Data Index (or a subset of it) may be advertised to other Application End-Points. A common use case for this is to provide the ability to locate data amongst a distributed set of Application End-Points (i.e., a data search mechanism).

5.2. DECADE Server

A DECADE Server 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.

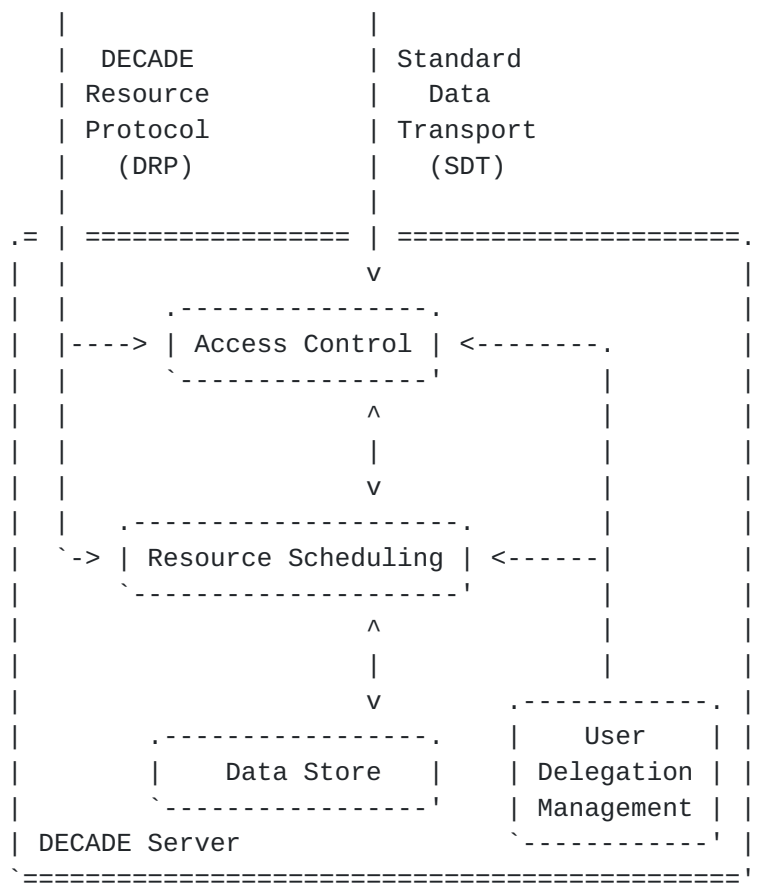


Figure 4: DECADE Server Components

5.2.1. 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, the providing Application End-Point, or the DECADE Server itself.

5.2.2. 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.

5.2.3. Data Store

Data from applications may be stored at a DECADE Server. Data can be deleted from storage either explicitly or automatically (e.g., after a TTL expiration). It may be possible to perform optimizations in certain cases, such as avoiding writing temporary data (e.g., live streaming) to persistent storage, if appropriate storage hints are supported by the SDT.

5.3. 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.

5.3.1. DECADE Data Object Naming Scheme

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 DECADE to evolve by either changing the hashing algorithm (e.g., if security vulnerabilities with an existing hashing algorithm are discovered), or moving 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.

In certain scenarios (e.g., where a DECADE client has limited computational power), it may be advantageous to offload the computation of a data object's name to the DECADE Server. This possibility is not considered in the current architecture, but may be a topic for future extensions.

5.3.2. Application Usage

Recall from Section 5.1.1 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.

5.3.3. Application Usage Example

To illustrate these properties, this section presents multiple examples.

5.3.3.1. Application with Fixed-Size Chunks

Similar to the example in Section 5.1.1, 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 5.

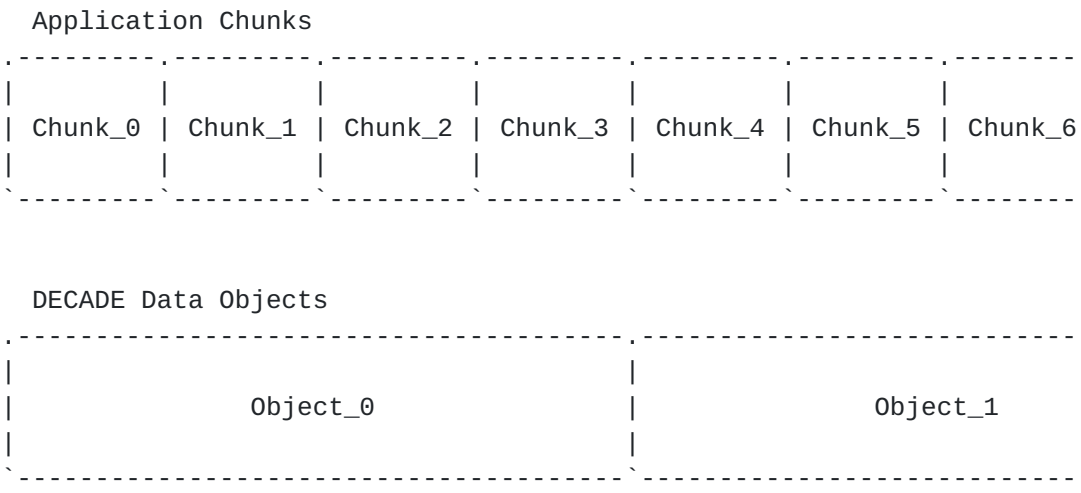


Figure 5: Mapping Application Chunks to DECADE Data Objects

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.

5.3.3.2. 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 6 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.

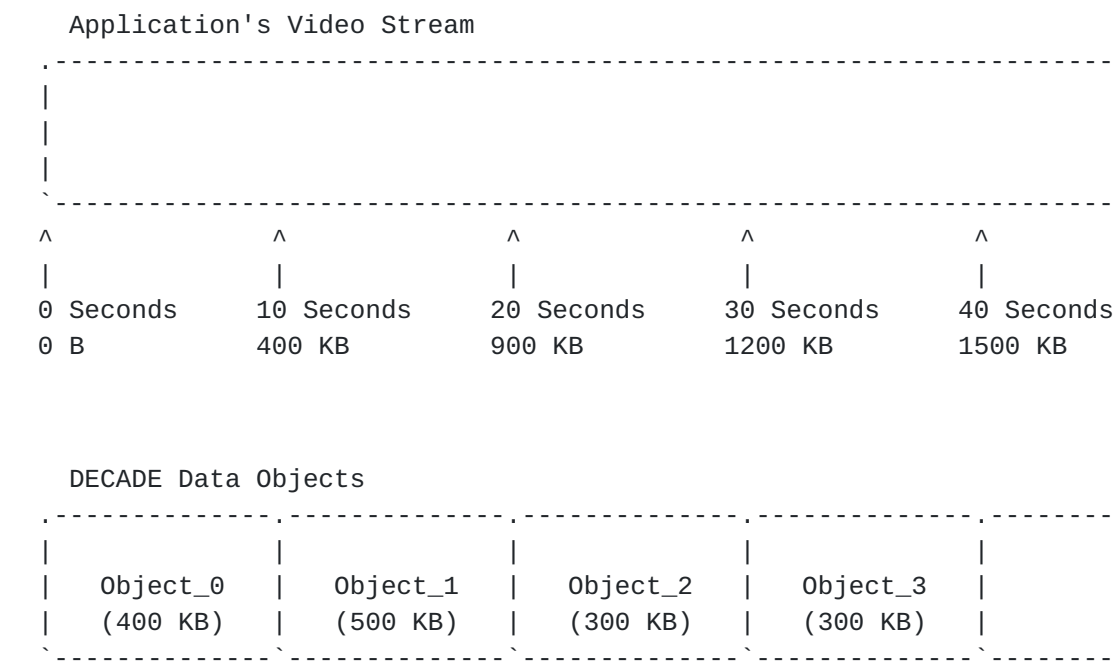


Figure 6: Mapping a Continuous Data Stream to DECADE Data Objects

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.

5.4. 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 6.1.2 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 6.1.2 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 for DECADE, but in such a case, a DECADE Client may be denied access even though it possess a valid token.

5.5. Discovery

DECADE includes a discovery mechanism through which DECADE clients locate an appropriate DECADE Server. [I-D.ietf-decade-reqs] details specific requirements of the discovery mechanism; this section discusses how they relate to other principles outlined in this document.

A discovery mechanism allows a DECADE client to determine an IP address or some other identifier that can be resolved to locate the server for which the client will be authorized to generate tokens (via DRP). (Note that the discovery mechanism may also result in an error if no such DECADE servers can be located.) After discovering one or more DECADE servers, a DECADE client may distribute load and requests across them (subject to resource limitations and policies of the DECADE servers themselves) according to the policies of the Application End-Point in which it is embedded.

The particular protocol used for discovery is out of scope of this document, but any specification will re-use standard protocols wherever possible.

It is important to note that the discovery mechanism outlined here

does not provide the ability to locate arbitrary DECADE servers to which a DECADE client might obtain tokens from others. To do so requires application-level knowledge, and it is assumed that this functionality is implemented in the Content Distribution Application, or if desired and needed, as an extension to this DECADE architecture.

6. DECADE Protocols

This section presents 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).

6.1. 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 apply access control and resource sharing policies on each of them.

6.1.1. 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.

6.1.2. Access and Resource Control Token

A 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 (e.g., a weight used in a weighted bandwidth sharing scheme)

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 identifier to allow a DECADE Server to detect when a token is used multiple times.

6.1.3. 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 4.5. 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 4.5. 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.

6.1.4. Object Attributes

Objects that are stored on a DECADE server may have associated attributes (in addition to the object identifier and the actual content) that relate to the data storage and its management. These attributes may be used by the DECADE server (and possibly the underlying storage system) to perform specialized processing or handling for the data object, or to attach related DECADE server or storage-layer properties to the data object. These attributes have a scope local to a DECADE server. In particular, these attributes are not applied to a DECADE server or client to which a data object is copied.

Depending on authorization, DECADE clients may get or set such attributes. This authorization (and the mapping to application contexts) is based on the user delegation concept as described in Section 4.5. The DECADE architecture does not limit the set of permissible attributes, but rather specifies a set of baseline attributes that SHOULD be supported by implementations.

Suggested attributes are the following:

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)

It is important to note that the Object Attributes defined here are distinct from application metadata (see Section 4.1). Application metadata is custom information that an application may wish to associate with a data object to understand its semantic meaning (e.g., whether it is video and/or audio, its playback length in time, or its index in a stream). If an application wishes to store such metadata persistently within DECADE, it can be stored within data objects themselves.

6.2. 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.

An SDT used in DECADE SHOULD offer a transport mode that provides

confidentiality and integrity.

6.2.1. Writing/Uploading Objects

To write an object, a client first generates the object's name (see Section 5.3), and then uploads the object to a DECADE server and supplies the generated name. The 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 MAY 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 5.3. 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 5.3

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:

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

ERRORs:

VALIDATION_FAILED: The contents of the data object received by the DECADE server did not match the provided name (i.e., hash validation failed).

PERMISSION_DENIED: The DECADE client lacked sufficient permission to store the object.

Specifics regarding error handling, including additional error conditions, precedence for returned errors and its relation with server policy, are deferred to eventual protocol specification.

6.2.2. 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 Appendix A.1 for an example how this could be mapped to HTTP.

Method GET:

Parameters:

NAME: The naming of the object according to Section 5.3.

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:

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

PERMISSION_DENIED: The DECADE client lacked sufficient permission to read the object.

NOT_AVAILABLE: The data object exists but is currently unavailable for download (e.g., due to server policy).

Specifics regarding error handling, including additional error conditions (e.g. overload), precedence for returned errors and its relation with server policy, are defered to eventual protocol specification.

7. Server-to-Server Protocols

An important feature of DECADE is the capability for one DECADE server to directly download 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.

To support this functionality, DECADE uses the DRP and SDT 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. Note, however, that the objects being processed do not necessarily have to originate or terminate at the DECADE client (i.e. the object may be limited to being exchanged between DECADE servers even if the instruction is triggered by the client). DECADE clients thus 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 (e.g. PUT, GET); and
- Credentials indicating permission to perform the operation at the remote DECADE server.

In this way, a DECADE server acts as a proxy for a DECADE client, and a DECADE client may instantiate requests via that proxy. The operations are performed as if the original requester had its own DECADE client co-located with the DECADE server. It is this mode of operation that provides substantial savings in uplink capacity. Note that this mode of operation may also be triggered by an administrative/management application outside the DECADE architecture.

7.1. 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. Note that object attributes (see Section 6.1.4) may also be specified in the request to the remote server.

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 (proxy) 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 optionally return the object to a client. In the case of a PUT operation, the DECADE server is to store the object to the remote server using the specified credentials (via a PUT request to the remote server). The object may optionally be uploaded from the client or may already exist at the proxying server.

8. 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.

8.1. Pipelining to Avoid Store-and-Forward Delays

A DECADE server may choose to not fully store an object before beginning to serve it. For example, when serving a GET request, instead of waiting for the complete data to arrive from a remote server or DECADE client, a DECADE server may forward received data

bytes as they come in. This pipelining mode reduces store-and-forward delays, which could be substantial for large objects. A similar behavior could be used for PUT.

8.2. 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, which attempts to avoid storing identical data multiple times. A DECADE Server implementation may internally perform de-duplication of data on disk. The DECADE architecture enables additional forms of de-duplication.

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

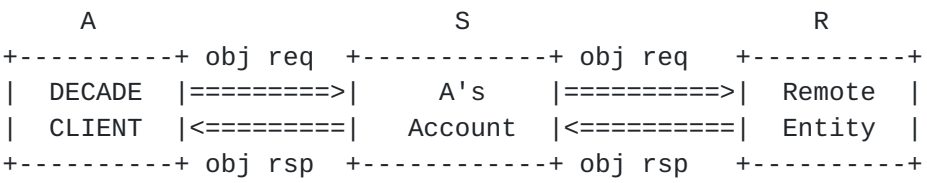
8.2.1. Traffic De-duplication

8.2.1.1. Rationale

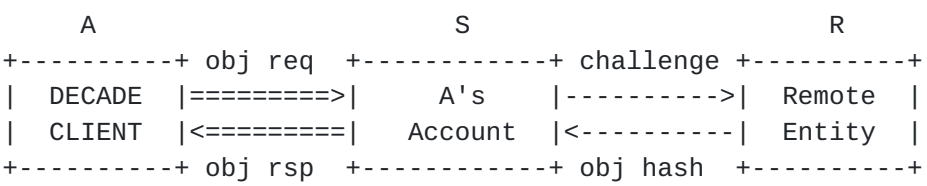
When a DECADE client (A) indicates its DECADE account on a DECADE server (S) to fetch an object from a remote entity (R) (a DECADE server or DECADE client) and if the object is already stored locally in S, S may perform Traffic De-duplication. This means that S does not download the object from R, in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

8.2.1.2. An Example

As shown in Figure 7, without Traffic De-duplication, unnecessary transfer of an object from R to S may happen, if the server S already has the object requested by A. If Traffic De-duplication is enabled, S only needs to challenge R to verify that it does have the data to avoid data-stealing attacks.



(a) Without Traffic De-duplication



(b) With Traffic De-duplication

Figure 7

8.2.1.3. HTTP Compatibility of Challenge

How to integrate traffic de-duplication with HTTP is shown in Appendix A.1.3.

8.2.2. Cross-Server Storage De-duplication

The same object might be uploaded multiple times to different DECADE servers. For storage efficiency, storage providers may desire that a single object be stored on one or a few servers. They might implement an internal mechanism to achieve the goal, for example, by redirecting requests to proper servers. DECADE supports the redirection of DECADE client requests to support further cross-server storage de-duplication.

9. Security Considerations

In general, the security considerations mentioned in [I-D.ietf-decade-problem-statement] apply to this document as well.

In addition, it should be noted that the token-based approach Section 5.4 provides authorization through token delegation. The strength of this authorization depends on several factors:

1. the uniqueness of tokens: tokens should be constructed in a way that minimize the possibilities for collisions;

2. validity of tokens: applications/users should not re-use tokens; and

3. secrecy of tokens: if tokens are compromised to unauthorized entities, access control for the associated resources cannot be provided.

Depending on the specific application, DECADE can be used to access confidential information. Hence DECADE implementations SHOULD provide a secure transport mode that allows for encryption.

10. IANA Considerations

This document does not have any IANA considerations.

11. Acknowledgments

We thank the following people for their contributions to this document:

David Bryan

Yingjie Gu

David McDysan

Borje Ohlman

Haibin Song

Martin Stiemerling

Richard Woundy

Ning Zong

12. References

12.1. Normative References

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

12.2. 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.

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

[RFC6392] Alimi, R., Rahman, A., and Y. Yang, "A Survey of In-Network Storage Systems", RFC 6392, October 2011.

[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-04 (work in progress), October 2011.

[I-D.ietf-decade-reqs] Yingjie, G., Bryan, D., Yang, Y., and R. Alimi, "DECADE Requirements", draft-ietf-decade-reqs-04 (work in progress), September 2011.

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

[GoogleFileSystem] Ghemawat, S., Gobioff, H., and S. Leung, "The Google File System", SOSP 2003, October 2003.

[CDMI] "CDMI", <http://www.snia.org/cdmi>.

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 6 for DECADE DRP and SDT can be

fulfilled by existing protocols such as HTTP, WEBDAV, and CDMI.

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

Communication 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 communication 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 Data 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 similar but only fetches the metadata ("header") associated with the resource but not the resource itself.

A.1.3. Traffic De-duplication 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 function 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 identifies the 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 presentation 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.

A.3. CDMI

The Cloud Data Management Interface (CDMI) specification defines a functional interface through which applications can store and manage data objects in a cloud storage environment. The CDMI interface for reading/writing data is based on standard HTTP requests, with CDMI-specific encodings using JavaScript Object Notation (JSON). CDMI is specified by the Storage Networking Industry Association (SNIA) [CDMI].

A.3.1. CDMI Support for DECADE Resource Protocol Primitives

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

A.3.1.1. Access Control Primitives

Access control includes mechanisms for defining the access policies for the server, and then checking the authorization of a user before it stores or retrieves content. CDMI defines an Access Control List (ACL) per data object, and thus supports access control (read and/or

write) at the data object granularity. 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.

CDMI requires that an HTTP authentication mechanism be available for the server to validate the identity of a principal (client). Specifically, CDMI requires that either HTTP Basic Authentication or HTTP Digest Authentication be supported. CDMI recommends that HTTP over TLS (HTTPS) is supported to encrypt the data sent over the network.

A.3.1.2. Communication Resource Controls Primitives

Communication resources include bandwidth (upload/download) and number of simultaneous connected clients (connections). CDMI supports two key data attributes which provide control over the communication resources to a client: "cdmi_max_throughput" and "cdmi_max_latency". These attributes are defined in the metadata for data objects and indicate the desired bandwidth or delay for transmission of the data object from the cloud server to the client.

A.3.1.3. Storage Resource Control Primitives

Storage resources include amount of quantity and lifetime of storage. CDMI defines metadata for individual data objects and general storage system configuration which can be used for storage resource control. In particular, CDMI defines the following metadata fields:

- cdmi_data_redundancy: desired number of copies to be maintained,

- cdmi_geographic_placement: region where object is permitted to be stored,

- cdmi_retention_period: time interval object is to be retained, and

- cdmi_retention_autodelete: whether object should be auto deleted after retention period.

A.3.2. CDMI 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.3.2.1. Writing Primitives

Writing involves uploading objects to the server. CDMI supports standard HTTP methods for PUT and POST as described in Appendix A.1.2.1.

A.3.2.2. Downloading Primitives

Downloading involves fetching of an object from the server. CDMI supports the standard HTTP GET method as described in Appendix A.1.2.2.

A.3.3. Other Operations

CDMI supports DELETE as described in Appendix A.1.4. CDMI also supports COPY and MOVE operations.

CDMI supports the concept of containers of data objects to support joint operations on related objects. For example, GET may be done on a single data object or on an entire container.

CDMI supports a global naming scheme. Every object stored within a CDMI system will have a globally unique object string identifier (ObjectID) assigned at creation time.

A.3.4. Conclusions

CDMI 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 CDMI features may be useful for DECADE:

- access control
- storage resource control
- communication resource control
- COPY/MOVE operations
- data containers
- naming scheme

Appendix B. 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 [RFC6392] 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.

B.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.

B.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.

B.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.

B.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.

B.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.

B.6. Discovery Mechanism

The particular protocol used for discovery is outside the scope of this document. However, options and considerations have been discussed in Section 5.5.

B.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.

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