DECADE
Internet-Draft
Intended status: Standards Track
Expires: September 8, 2011

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

March 7, 2011

DECADE Architecture
draft-ietf-decade-arch-00

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 September 8, 2011.

Copyright Notice

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.gu-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 and Data Planes

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

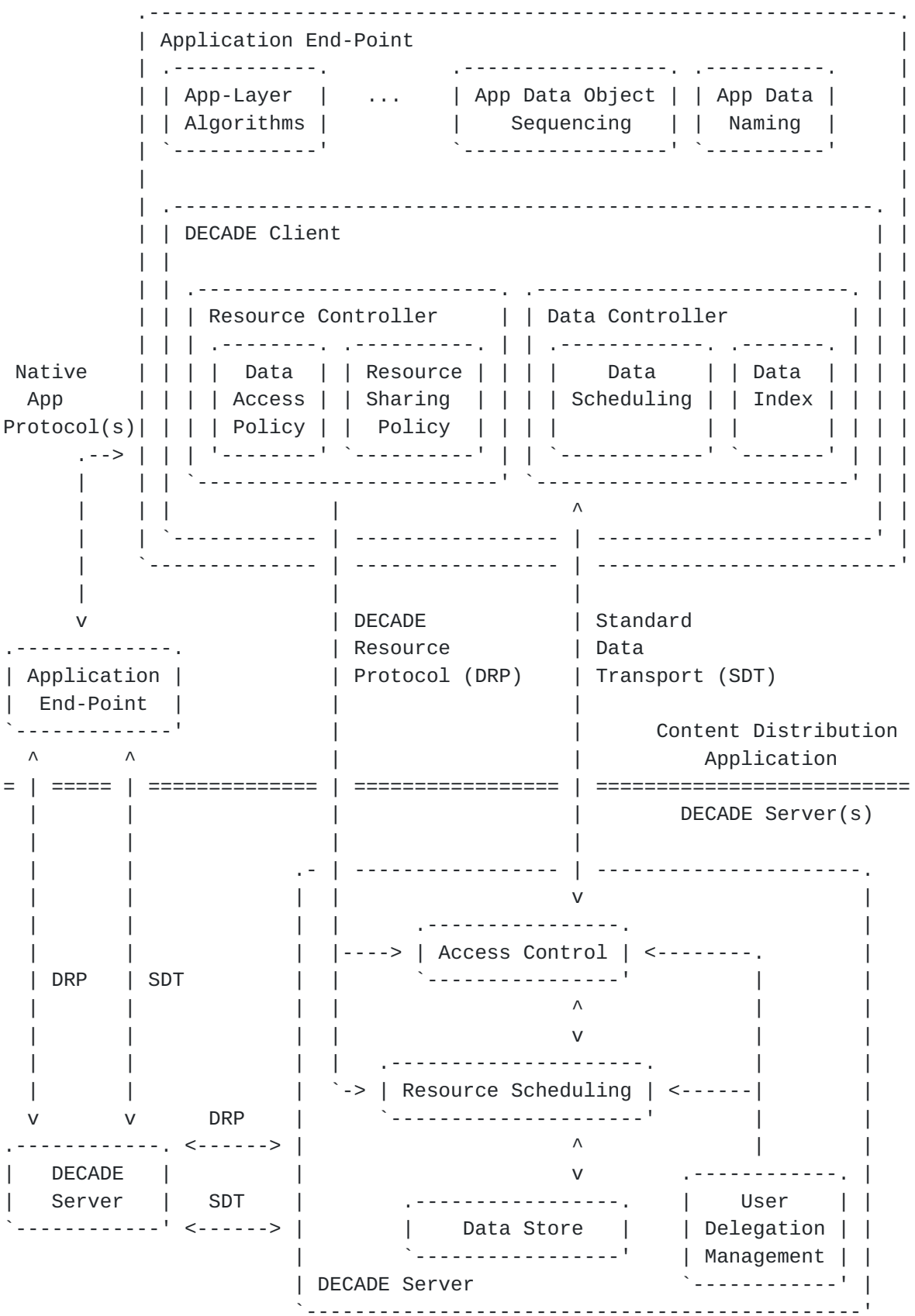
Application End-Point
App-Layer Algorithms
...
App Data Object Sequencing
App Data Naming
DECADE Client
Resource Controller
Data Access Policy
Resource Sharing Policy
Data Controller
Data Scheduling
Data Index
Native App Protocol(s)
Application End-Point
DECADE Resource Protocol (DRP)
Standard Data Transport (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 Sequencing and Naming

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 re-assembling data objects and also creating the individual data objects. We call this component Application Data Sequencing. The specific implementation is entirely decided by the application.

In assembling or producing the data objects, an important consideration is the naming of these objects. We call the component responsible for assigning and interpreting application-layer names the Application Data Naming component. The specific implementation is entirely decided by the application.

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. DECADE Data Sequencing and Naming

We have discussed above that an Application may have its own behavior for both sequencing and naming data objects. In order to provide a simple and generic interface, the DECADE Server is only responsible for storing and retrieving individual data objects.

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 naming to the DECADE naming. Furthermore, the DECADE naming scheme implies no sequencing or grouping of objects, even if this is done at the application layer.

Multiple applications may make use of a DECADE infrastructure, and each Application may employ its own naming scheme. To remain independent of particular applications, DECADE uses a simple, common naming scheme that supports unique naming of individual data objects. This is achieved by deriving object names from hashes of the object content. This scheme is made possible by the fact that DECADE data objects are immutable. Details of the naming scheme (complete syntax, hash algorithms etc.) will be defined in a future version of this document.

By naming data objects directly as the content hash, DECADE names satisfy important objectives:

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

A particular advantage of using the content hash is that it is straightforward for as server or client to validate a data object before storing or transmitting it. While these capabilities could be achieved by supplying the content hash in both read and write requests as metadata, using the content hash as the name satisfies the objectives and is straightforward.

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.

4.5. 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.

4.5.1. 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.

4.5.2. 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.

4.5.3. 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.

4.5.4. 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.

4.5.5. 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.

4.5.6. 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.

4.5.7. 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 such as HTTP. It is possible that a single specific protocol provides both, DRP and SDT functionality.

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). A detailed architectural description of the DRP and SDT is still a work in progress. Important aspects, including details of authorization, will be added in a future version of this document.

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.

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

communication resources: DECADE servers may 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 may limited storage resources.

Note: this list of resources will be extended in a future version of this document.

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).

6.2. Potential Optimizations

As a suggestion to the protocol and eventual implementations, we would like to point out particular optimizations that could be made when making use of the server-to-server protocol.

6.2.1. 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, in the case of a GET request, a DECADE server may begin to receive a data object from a remote server, 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. Security Considerations

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

8. IANA Considerations

This document does not have any IANA considerations.

9. Informative References

[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-02 (work in progress), January 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.gu-decade-reqs]
Yingjie, G., Bryan, D., Yang, Y., and R. Alimi, "DECADE Requirements", draft-gu-decade-reqs-05 (work in progress), July 2010.

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

In this section we illustrate how the abstract protocol interactions specified in Section 5 for DECADE DRP and SDT can be fulfilled by existing protocols such as HTTP and WEBDAV. (This version of the document considers HTTP only, a future version will provide a possible WEBDAV-based illustration as well.)

A.1. HTTP

HTTP is a key protocol for the Internet and specifically the World Wide Web. HTTP is a request-response protocol. A typical transaction is when a client (e.g. web browser) requests content (resources) from a web server. Other examples are when the client puts or deletes content from the 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 access control via "HTTP Secure" (HTTPS). HTTPS is a combination of HTTP with SSL/TLS. The main purpose of HTTPS is to authenticate the server and encrypt all traffic between the client and the server. HTTPS does not support authentication of the client.

A.1.1.2. Communication Resource Controls Primitives

Communications resources include bandwidth (upload/download) and number of simultaneous connected clients (connections). HTTP supports communication resource control 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). HTTP does not support a mechanism to allow limiting the communciation resources to a client.

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 can be identified by a URL. 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.

A.1.2.2. Downloading Primitives

Downloading involves fetching of an object from the server. HTTP supports downloading through the GET method. GET fetches a specific resource as identified by the URL.

A.1.2.3. Other Methods

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

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