DECoupled Application Data Enroute (DECADE)

Abstract

   Content distribution applications, such as those employing peer-to-
   peer (P2P) technologies, are widely used on the Internet and make up
   a large portion of the traffic in many networks. Often, however,
   content distribution applications use network resources
   inefficiently. One way to improve efficiency is to introduce storage
   capabilities within the network and enable cooperation between end-
   host and in-network content distribution mechanisms. This is the
   capability provided by a DECoupled Application Data Enroute (DECADE)
   system, which is introduced in this document. DECADE enables
   applications to take advantage of in-network storage when
   distributing data objects as opposed to using solely end-to-end
   resources. This document presents the underlying principles and key
   functionalities of such a system and illustrates operation through a
   set of examples.

Status of This Memo

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

   This is a contribution to the RFC Series, independently of any other
   RFC stream. The RFC Editor has chosen to publish this document at
   its discretion and makes no statement about its value for
   implementation or deployment. Documents approved for publication by
   the RFC Editor are not a candidate for any level of Internet
   Standard; see Section 2 of RFC 5741.

   Information about the current status of this document, any errata,
   and how to provide feedback on it may be obtained at
   http://www.rfc-editor.org/info/rfc7069.

Table of Contents

1. Introduction

   Content distribution applications, such as peer-to-peer (P2P)
   applications, are widely used on the Internet to distribute data
   objects and make up a large portion of the traffic in many networks.
   Said applications can often introduce performance bottlenecks in
   otherwise well-provisioned networks. In some cases, operators are
   forced to invest substantially in infrastructure to accommodate the
   use of such applications. For instance, 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 and
   devices at individual homes, businesses, DSLAMs (Digital Subscriber
   Line Access Multiplexers), and CMTSs (Cable Modem Termination
   Systems) in remote locations. It may be more practical and
   economical to upgrade the core infrastructure, instead of the "last
   mile" of the network, as this involves fewer components that are
   shared by many subscribers. See [RFC6646] and [RFC6392] for a more
   complete discussion of the problem domain and general discussions of
   the capabilities envisioned for a DECADE system. As a historical
   point, it should be noted that [RFC6646] and [RFC6392] came out of
   the now closed DECADE Working Group. This document aims to advance
   some of the valuable concepts from that now closed Working Group.

   This document presents mechanisms for providing in-network storage
   that can be integrated into content distribution applications. The
   primary focus is P2P-based content distribution, but DECADE may be
   useful to other applications with similar characteristics and
   requirements (e.g., Content Distribution Networks (CDNs) or hybrid
   P2P/CDNs). The approach we adopt in this document is to define the
   core functionalities and protocol functions that are needed to
   support a DECADE system. This document provides illustrative
   examples so that implementers can understand the main concepts in
   DECADE, but it is generally assumed that readers are also familiar
   with the terms and concepts used in [RFC6646] and [RFC6392].

1.1. 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].

2. Terminology

   This document uses the following terminology.

   Application Endpoint
      A host that includes a DECADE client along with other application
      functionalities (e.g., peer-to-peer (P2P) client, video streaming
      client).

   Content Distribution Application
      A specific type of application that may exist in an Application
      Endpoint. A content distribution application is an application
      (e.g., P2P) designed for dissemination of large amounts of content
      (e.g., files or video streams) to multiple peers. Content
      distribution applications may divide content into smaller blocks
      for dissemination.

   Data Object
      A data object is the unit of data stored and retrieved from a
      DECADE server. The data object is a sequence of raw bytes. The
      server maintains metadata associated with each data object, but
      the metadata is physically and logically separate from the data
      object.

   DECADE Client
      A DECADE client uploads and/or retrieves data from a DECADE
      server.

   DECADE Resource Protocol (DRP)
      A logical protocol for communication of access control and
      resource-scheduling policies from a DECADE client to a DECADE
      server, or between DECADE servers. In practice, the functionality
      of the DRP may be distributed over one or more actual protocols.

   DECADE Server
      A DECADE server stores data inside the network for a DECADE client
      or another DECADE server, and thereafter it manages both the
      stored data and access to that data by other DECADE clients.

   DECADE Storage Provider
      A DECADE storage provider deploys and/or manages DECADE servers
      within a network.

   DECADE System
      An in-network storage system that is composed of DECADE clients
      and DECADE servers. The DECADE servers may be deployed by one or
      more DECADE storage providers.

In-Network Storage
   A service inside a network that provides storage to applications.
   In-network storage may reduce upload/transit/backbone traffic and
   improve application performance. In-network storage may, for
   example, be co-located with the border router (network-attached
   storage) or inside a data center. A DECADE system is an example
   of an in-network storage system.

Standard Data Transfer (SDT) Protocol
   A logical protocol used to transfer data objects between a DECADE
   client and DECADE server, or between DECADE servers. The intent
   is that in practice the SDT should map to an existing, well-known
   protocol already in use over the Internet for transporting data.

3. Overview

A DECADE system provides a distributed storage service for content
distribution applications (e.g., P2P). The system consists of
clients and servers. A client first uploads data objects to one or
more selected servers and optionally requests distribution of these
data objects to other servers. The client then selectively
authorizes other clients to download these data objects. Such a
system is employed in an overall application context (e.g., P2P file
sharing), and it is expected that DECADE clients take part in
application-specific communication sessions.

Figure 1 is a schematic of a simple DECADE system with two DECADE
clients and two DECADE servers. As illustrated, a DECADE client,
which is part of an Application Endpoint, uses the DECADE Resource
Protocol (DRP) to convey to a server information related to access
control and resource-scheduling policies. DRP can also be used
between servers for exchanging this type of information. A DECADE
system employs the Standard Data Transfer (SDT) protocol to transfer
data objects to and from a server, as we will explain later.

```
                        Native Application
                          Protocol(s)
                          (e.g., P2P)
 .--------------.                                .--------------.
 | Application  |    <------------------->       | Application  |
 | Endpoint     |                                | Endpoint     |
 |              |                                |              |
 |  .--------.  |                                |  .--------.  |
 |  | DECADE |  |                                |  | DECADE |  |
 |  | Client |  |                                |  | Client |  |
 |  .--------.  |                                |  .--------.  |
 .--------------.                                .--------------.
         |   ^                                          |   ^
 DECADE  |   | Standard                                 |   |
 Resource|   | Data                        DRP          |   |  SDT
 Protocol|   | Transfer                                 |   |
 (DRP)   |   | (SDT)                                    |   |
         |   |                                          |   |
         |   |                                          |   |
         v   v                                          v   v
 .=============.              DRP                 .=============.
 | DECADE      |   <------------------->          | DECADE      |
 | Server      |   <------------------->          | Server      |
 =============.              SDT                  =============.
```
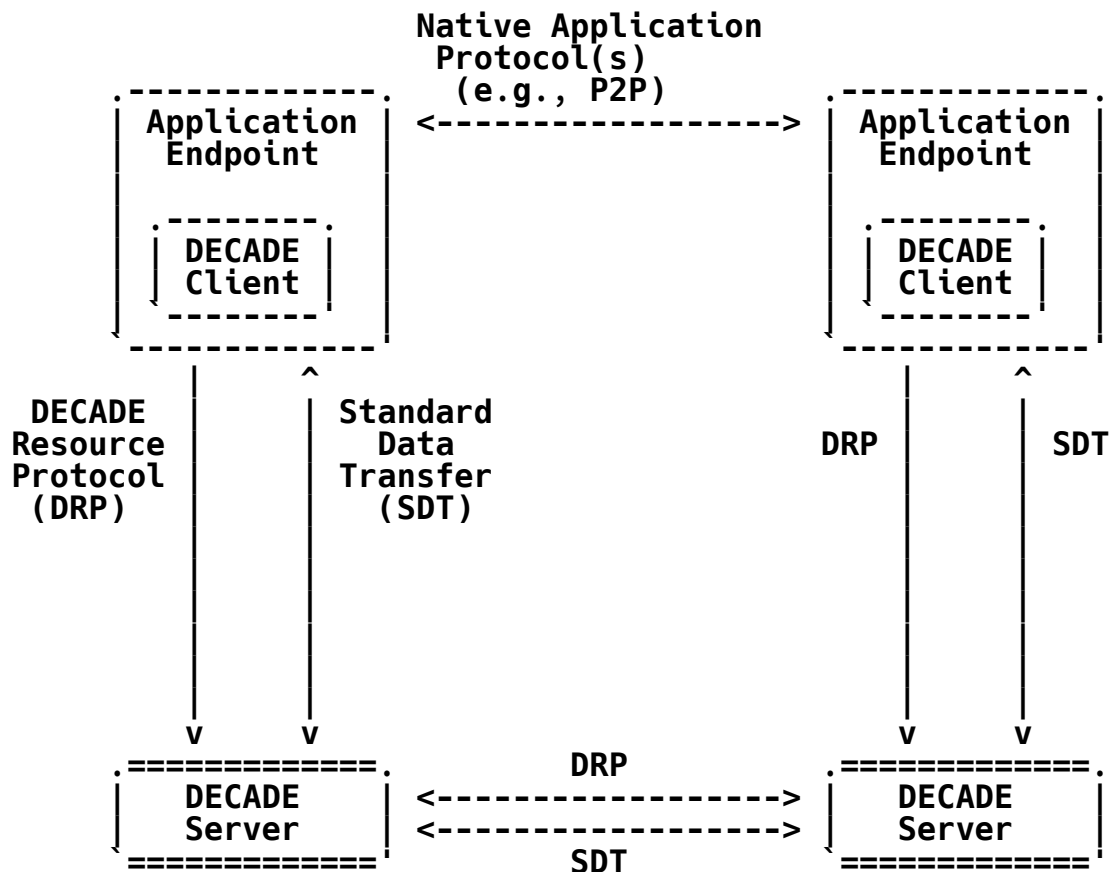
                      Figure 1: DECADE Overview

   With Figure 1 at hand, assume that Application Endpoint B requests a
   data object from Application Endpoint A using their native
   application protocols (e.g., P2P protocol) as in Figure 2. In this
   case, Endpoint A will act as the sender, and Endpoint B as the
   receiver for said data object. S(A) is the DECADE storage server
   which is access controlled. This means, first, that Endpoint A has a
   right to store the data object in S(A). Secondly, Endpoint B needs
   to obtain authorization before being able to retrieve the data object
   from S(A).

   The four steps involved in a DECADE session are illustrated in
   Figure 2. The sequence starts with the initial contact between
   Endpoint B and Endpoint A, where Endpoint B requests a data object
   using their native application protocol (e.g., P2P). Next, Endpoint
   A uses DRP to obtain a token corresponding to the data object that
   was requested by Endpoint B. There may be several ways for Endpoint
   A to obtain such a token, e.g., compute it locally or request one
   from its DECADE storage server, S(A). Once obtained, Endpoint A then

provides the token to Endpoint B (again, using their native
application protocol). Finally, Endpoint B provides the received
token to S(A) via DRP, and subsequently requests and downloads the
data object via SDT. Again, it is assumed that DECADE is employed in
an overall application context (e.g., P2P file-sharing session).

For completeness, note that there is an important prerequisite step
(not shown) to Figure 2, where Endpoint A first discovers and then
stores the data object(s) of interest in S(A).

```
                              .----------.
  2. Obtain      -------->  |   S(A)   | <------
     Token        /          .----------.        \   4. Request and
     (DRP)       /                                 \     Download
     Locally    /                                   \    Data Object
     or From   /                                     \   (DRP + SDT)
     S(A)    v         1. App Request                 v
  .-------------.  <--------------------------  .-------------.
  | Application |                               | Application |
  | Endpoint A  |                               | Endpoint B  |
  .-------------.  -------------------------->  .-------------.
                   3. App Response (token)
```
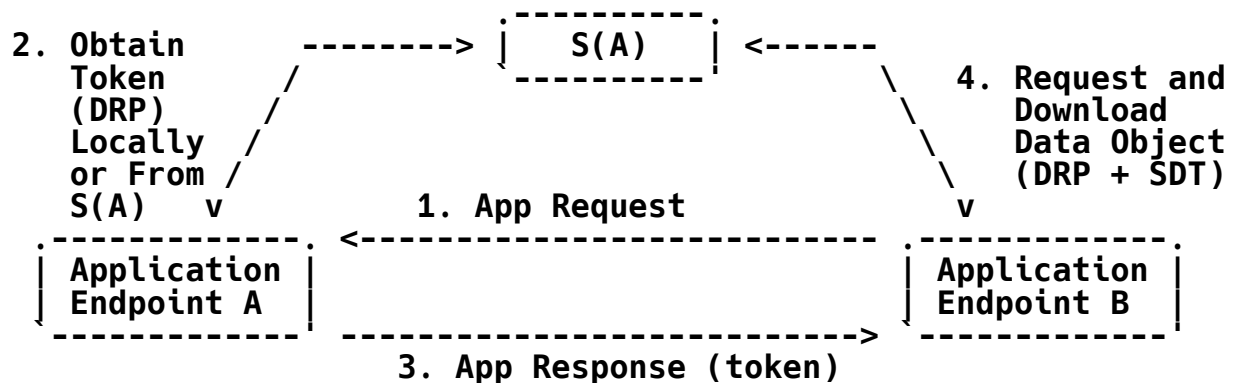
                  Figure 2: Download from Storage Server

4. Architectural Principles

   This section presents the key principles followed by any DECADE
   system.

4.1. Data- and Control-Plane Decoupling

   DECADE SDT and DRP can be classified as belonging to data-plane
   functionality. The algorithms and signaling for a P2P application,
   for example, would belong to control-plane functionality.

   A DECADE system aims to be application independent and should support
   multiple content distribution applications. Typically, a complete
   content distribution application implements a set of control-plane
   functions including content search, indexing and collection, access
   control, replication, request routing, and QoS scheduling.
   Implementers of different content distribution applications may have
   unique considerations when designing the control-plane functions.
   For example, with respect to the metadata management scheme,
   traditional file systems provide a standard metadata abstraction: a
   recursive structure of directories to offer namespace management
   where each file is an opaque byte stream. Content distribution
   applications may use different metadata management schemes. For

instance, one application might use a sequence of blocks (e.g., for
file sharing), while another application might use a sequence of
frames (with different sizes) indexed by time.

With respect to resource-scheduling algorithms, a major advantage of
many successful P2P systems is their substantial expertise in
achieving efficient utilization of peer resources. For instance,
many streaming P2P systems include optimization algorithms for
constructing overlay topologies that can support low-latency, high-
bandwidth streaming. The research community as well as implementers
of such systems continuously fine-tune existing algorithms and invent
new ones. A DECADE system should be able to accommodate and benefit
from all new developments.

In short, given the diversity of control-plane functions, a DECADE
system should allow for as much flexibility as possible to the
control plane to implement specific policies (and be decoupled from
data-plane DRP/SDT). Decoupling the control plane from the data
plane is not new, of course. For example, OpenFlow [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. The Google File System
[GoogleFileSystem] applies the same principle to file system design
by utilizing a Master to handle metadata management and several Chunk
servers to handle data-plane functions (i.e., read and write of
chunks of data). Finally, NFSv4.1's parallel NFS (pNFS) extension
[RFC5661] also adheres to this principle.

4.2. Immutable Data Objects

A common property of bulk content to be broadly distributed is that
it is immutable -- once content is generated, it is typically not
modified. For example, once a movie has been edited and released for
distribution, it is very uncommon that the corresponding video frames
and images need to be modified. The same applies to document
distribution, such as RFCs; audio files, such as podcasts; and
program patches. Focusing on immutable data can substantially
simplify data-plane design, since consistency requirements can be
relaxed. It also simplifies data reuse and the removal of
duplicates.

Depending on its specific requirements, an application may store
immutable data objects 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
data objects that require application-level assembly. Many content
distribution applications divide bulk content into data objects for
multiple reasons, including (a) fetching different data objects from

different sources in parallel and (b) faster recovery and
verification as individual data objects might be recovered and
verified. Typically, applications use a data object size larger than
a single packet in order to reduce control overhead.

A DECADE system should be agnostic to the nature of the data objects
and should not specify a fixed size for them. A protocol
specification based on this architecture may prescribe requirements
on minimum and maximum sizes for compliant implementations.

Note that immutable data objects can still be deleted. Applications
can support modification of existing data stored at a DECADE server
through a combination of storing new data objects and deleting
existing data objects. For example, a metadata management function
of the control plane might associate a name with a sequence of
immutable data objects. If one of the data objects is modified, the
meta-data management function changes the mapping of the name to a
new sequence of immutable data objects.

4.3. Data Object Identifiers

A data object stored in a DECADE server shall be accessed by DECADE
clients via a data object identifier. Each DECADE client may be able
to access more than one storage server. A data object that is
replicated across different storage servers managed by a storage
provider may be accessed through a single identifier. Since data
objects are immutable, it shall be possible to support persistent
identifiers for data objects.

Data object identifiers should be created by DECADE clients when
uploading the corresponding objects to a DECADE server. The scheme
for the assignment/derivation of the data object identifier to a data
object depends as the data object naming scheme and is out of scope
of this document. One possibility is to name data objects using
hashes as described in [RFC6920]. Note that [RFC6920] describes
naming schemes on a semantic level only, but specific SDTs and DRPs
use specific representations.

In particular, for some applications, it is important that clients
and servers be able to validate the name-object binding, i.e., by
verifying that a received object really corresponds to the name
(identifier) that was used for requesting it (or that was provided by
a sender). If a specific application requires name-object binding
validation, the data object identifiers can support it by providing
message digests or so-called self-certifying naming information.

Different name-object binding validation mechanisms may be supported
in a single DECADE system. Content distribution applications can
decide what mechanism to use, or to not provide name-object
validation (e.g., if authenticity and integrity can by ascertained by
alternative means). We expect that applications may be able to
construct unique names (with high probability) without requiring a
registry or other forms of coordination. Names may be self-
describing so that a receiving DECADE client understands, for
example, which hash function to use for validating name-object
binding.

Some content distribution applications will derive the name of a data
object from the hash over the data object; this is made possible by
the fact that DECADE objects are immutable. But there may be other
applications such as live streaming where object names will not based
on hashes but rather on an enumeration scheme. The naming scheme
will also enable those applications to construct unique names.

In order to enable the uniqueness, flexibility and self-describing
properties, the naming scheme used in a DECADE system should provide
a "type" field that indicates the name-object validation function
type (for example, "sha-256" [RFC5754]) and the cryptographic data
(such as an object hash) that corresponds to the type information.
Moreover, the naming scheme may additionally provide application or
publisher information.

4.4. Explicit Control

To support the functions of an application's control plane,
applications should be able to keep track and coordinate which data
is stored at particular servers. Thus, in contrast with traditional
caches, applications are given explicit control over the placement
(selection of a DECADE server), deletion (or expiration policy), and
access control for stored data objects. Consider deletion/expiration
policy as a simple example. An application might require that a
DECADE server stores data objects for a relatively short period of
time (e.g., for live-streaming data). Another application might need
to store data objects for a longer duration (e.g., for video on
demand), and so on.

4.5. Resource and Data Access Control through Delegation

A DECADE system provides a shared infrastructure to be used by
multiple Application Endpoints. Thus, it needs to provide both
resource and data access control, as discussed in the following
subsections.

4.5.1. Resource Allocation

There are two primary interacting entities in a DECADE system.
First, storage providers coordinate DECADE server provisioning,
including their total available resources. Second, applications
coordinate data transfers amongst available DECADE servers and
between servers and clients. A form of isolation is required to
enable each of the concurrently running applications to explicitly
manage its own data objects and share of resources at the available
servers. Therefore, a storage provider should delegate resource
management on a DECADE server to uploading DECADE clients, enabling
them to explicitly and independently manage their own share of
resources on a server.

4.5.2. User Delegation

DECADE storage providers will have the ability to explicitly manage
the entities allowed to utilize the resources available on a DECADE
server. This is needed for reasons such as capacity-planning and
legal considerations in certain deployment scenarios. The DECADE
server should grant a share of the resources to a DECADE client. The
client can in turn share the granted resources amongst its (possibly)
multiple applications. The share of resources granted by a server is
called a User Delegation. As a simple example, a DECADE server
operated by an ISP might be configured to grant each ISP subscriber
1.5 Mbit/s of network capacity and 1 GB of memory. The ISP
subscriber might in turn divide this share of resources amongst a
video-streaming application and file-sharing application that are
running concurrently.

5. System Components

As noted earlier, the primary focus of this document is the
architectural principles and the system components that implement
them. While specific system components might differ between
implementations, this 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 will require additional components (e.g.,
monitoring and accounting at a server), but they are intentionally
omitted from this document.

5.1. Application Endpoint

   Content distribution applications have many functional components.
   For example, many P2P applications have components and algorithms to
   manage overlay topology, rate allocation, piece selection, and so on.
   In this document, we focus on the components directly engaged in a
   DECADE system. Figure 3 illustrates the components discussed in this
   section from the perspective of a single Application Endpoint.

```
                            Native Application Protocol(s)
                           (with other Application Endpoints)
                           .---------------------->
                           |
                           V
       .----------------------------------------------------------------.
       | Application Endpoint                                           |
       | .-------------------.          .--------------------.          |
       | | Application-Layer |   ...    |  App Data Assembly |          |
       | |     Algorithms    |          |     Sequencing     |          |
       | '-------------------'          '--------------------'          |
       |    .===========================================================.
       |    | DECADE Client                                             |
       |    | .-------------------.      .-------------------------.     |
       |    | | Resource Controller|     | Data Controller         |     |
       |    | | .-------..--------.|     | .----------..-------.    |     |
       |    | | | Data  ||Resource||     | |   Data   || Data  |    |     |
       |    | | |Access ||-Sharing||     | |Scheduling|| Index |    |     |
       |    | | |Policy ||  Policy||     | |          ||       |    |     |
       |    | | '-------''--------'|     | '----------''-------'    |     |
       |    | '-------------------'      '--------------^----------'     |
       |  == |  ===========================  |==================== |     |
       '-----|  ---------------------------  |-------------------- |-----'
             |                               |
         DECADE Resource Protocol        Standard Data Transfer
               (DRP)                           (SDT)
             |                               |
             V                               V
```
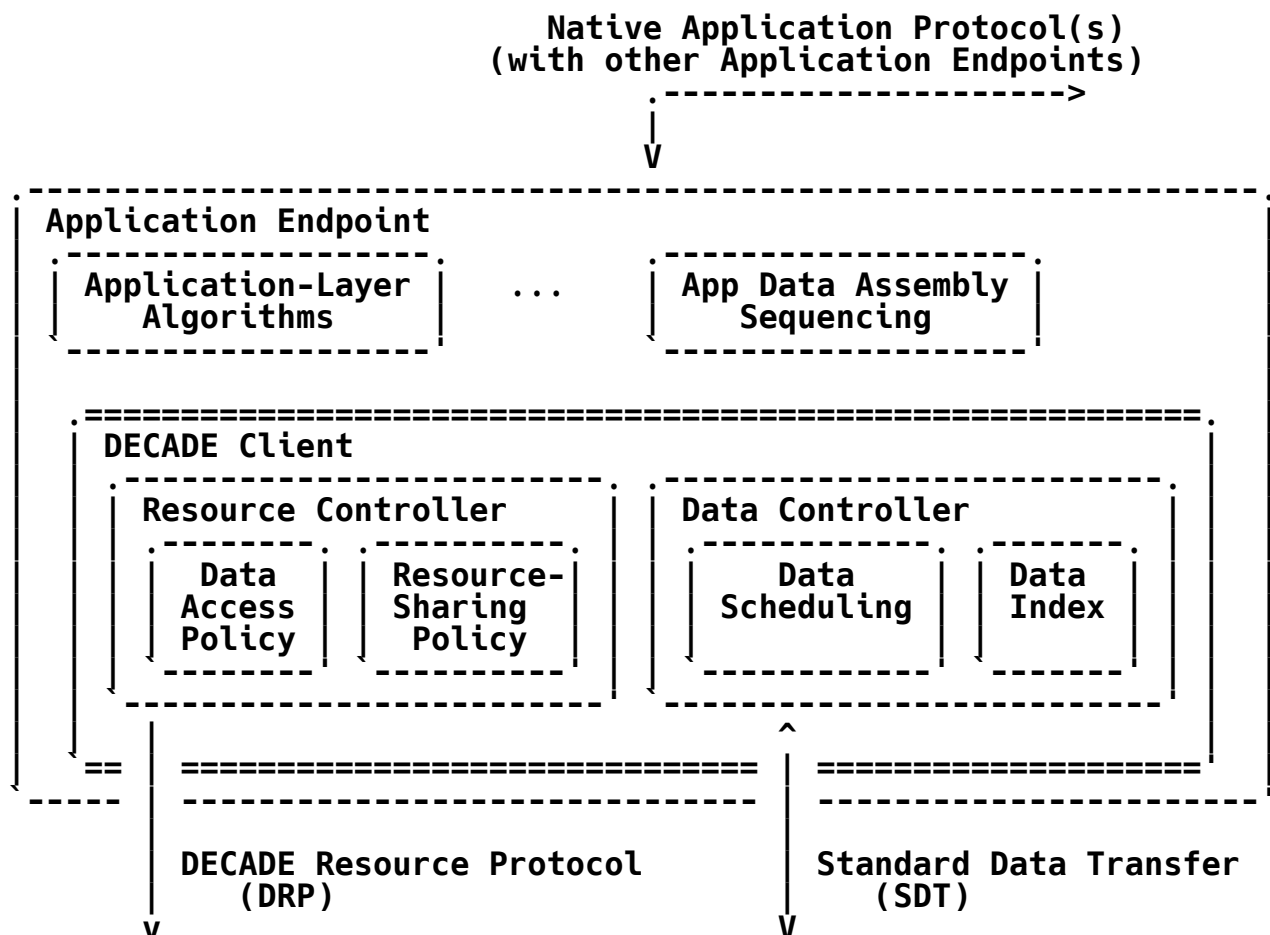
              Figure 3: Application and DECADE Client Components

   A DECADE system is geared towards supporting applications that can
   distribute content using data objects (e.g., P2P). To accomplish
   this, applications can include a component responsible for creating
   the individual data objects before distribution and for reassembling
   them later. We call this component Application Data Assembly. In
   producing and assembling data objects, two important considerations
   are sequencing and naming. A DECADE system assumes that applications

implement this functionality themselves. In addition to DECADE
DRP/SDT, applications will most likely also support other, native
application protocols (e.g., P2P control and data transfer
protocols).

5.2. DECADE Client

The DECADE client provides the local support to an application, and
it can be implemented standalone, embedded into the application, or
integrated in other software entities within network devices (i.e.,
hosts). In general, applications may have different resource-sharing
policies and data access policies with regard to DECADE servers.
These policies may be existing policies of applications or custom
policies. The specific implementation is decided by the application.

Recall that DECADE decouples the control and the data transfer of
applications. A data-scheduling component schedules data transfers
according to network conditions, available servers, and/or available
server resources. The Data Index indicates data available at remote
servers. The Data Index (or a subset of it) can be advertised to
other clients. A common use case for this is to provide the ability
to locate data amongst distributed Application Endpoints (i.e., a
data search mechanism such as a Distributed Hash Table (DHT)).

5.3. DECADE Server

Figure 4 illustrates the primary components of a DECADE server. Note
that the description below does not assume a single-host or
centralized implementation -- a DECADE server is not necessarily a
single physical machine; it can also be implemented in a distributed
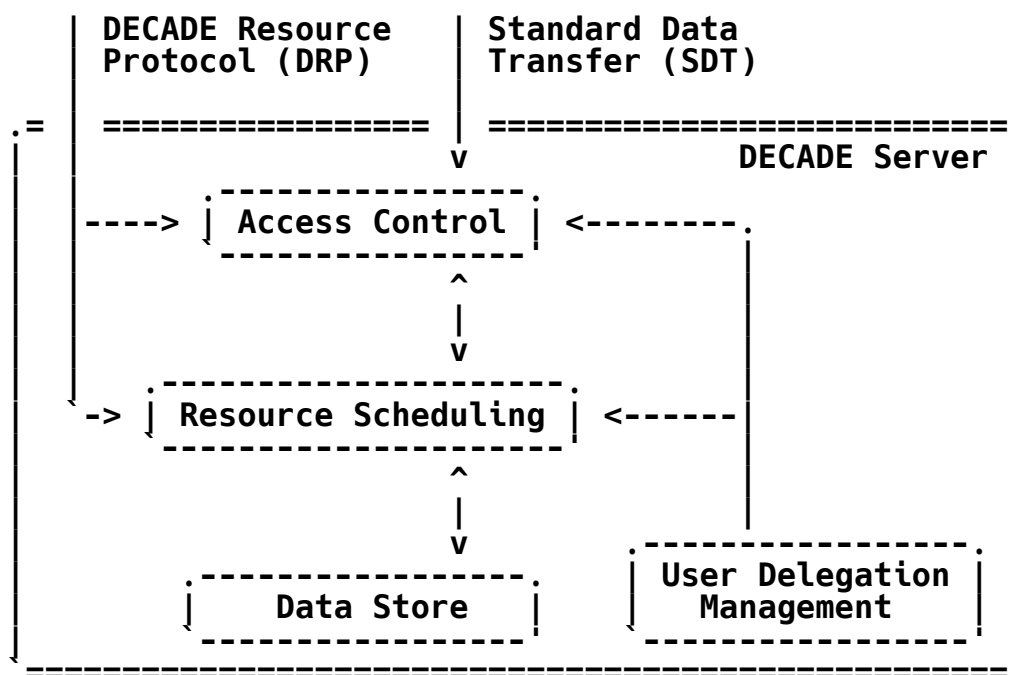manner on a cluster of machines.

```
             DECADE Resource │ Standard Data
             Protocol (DRP)  │ Transfer (SDT)

        .=  │ ===============  │ =========================.
        │   │                  v         DECADE Server   │
        │          .----------------.                    │
        │----->   │ Access Control │ <---------.          │
        │          .----------------.           │          │
        │                 ^                      │          │
        │                 │                      │          │
        │                 v                      │          │
        │          .--------------------.        │          │
        .->       │ Resource Scheduling │ <------│          │
             │     .--------------------.         │          │
             │            ^                       │          │
             │            │                       │          │
             │            v            .------------------.  │
        │          .----------------. │ User Delegation │  │
        │         │   Data Store   │ │   Management    │  │
        │          .----------------.  .------------------.  │
        .=================================================.
```

                Figure 4: DECADE Server Components

   Provided sufficient authorization, a client shall be able to access
   its own data or other client's data in a DECADE server. Clients may
   also authorize other clients to store data. If access is authorized
   by a client, the server should provide access. Applications may
   apply resource-sharing policies or use a custom policy. DECADE
   servers will then perform resource scheduling according to the
   resource-sharing policies indicated by the client as well as any
   other previously configured User Delegations. Data from applications
   will be stored at a DECADE server. Data may be deleted from storage
   either explicitly or automatically (e.g., after a Time To Live (TTL)
   expiration).

5.4. Data Sequencing and Naming

   The DECADE naming scheme implies no sequencing or grouping of
   objects, even if this is done at the application layer. To
   illustrate these properties, this section presents several examples
   of use.

5.4.1. Application with Fixed-Size Chunks

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 16 KB. Now, assume
that this application wishes to store data in a DECADE server in data
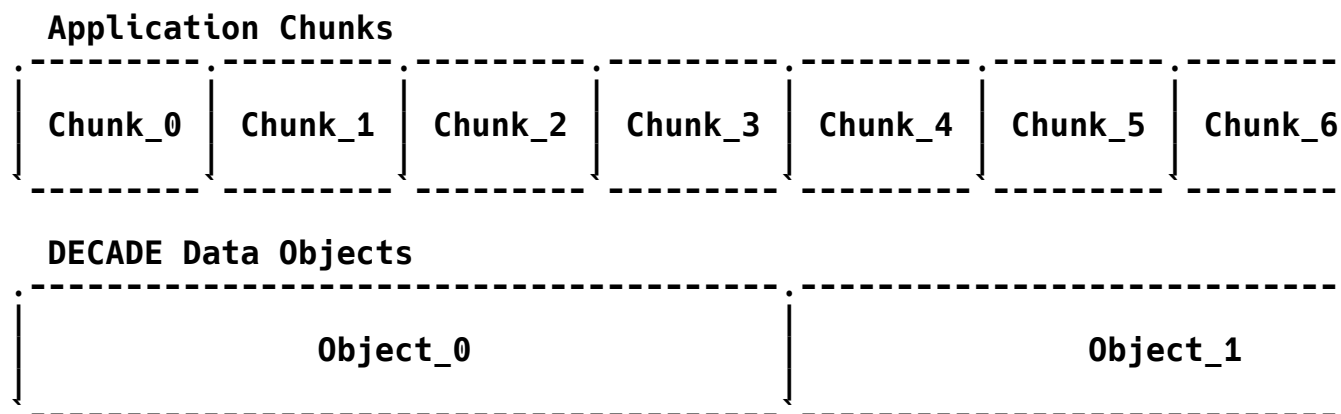objects of size 64 KB. To accomplish this, it can map a sequence of
4 chunks into a single data object, as shown in Figure 5.

```
   Application Chunks
   .--------.--------.--------.--------.--------.--------.--------
  |          |          |          |          |          |          |
  | Chunk_0  | Chunk_1  | Chunk_2  | Chunk_3  | Chunk_4  | Chunk_5  | Chunk_6
  |          |          |          |          |          |          |
   --------- --------- --------- --------- --------- --------- --------

   DECADE Data Objects
   .-------------------------------------.--------------------------
  |                                       |
  |               Object_0                |               Object_1
  |                                       |
   ------------------------------------- --------------------------
```

                Figure 5: Mapping Application Chunks to DECADE Data Objects

In this example, the application maintains 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 may be conveyed from
either the original uploading DECADE client, another Endpoint with
which the application is communicating, etc. As long as the data
contained within each sequence of chunks is globally unique, the
corresponding data objects have globally unique names.

5.4.2. Application with Continuous Streaming Data

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 data objects. Figure 6 depicts how a video streaming
application might produce variable-sized data objects such that each
data object contains 10 seconds of video data. In a manner similar
to the previous example, the application may maintain a mapping that
is able to determine the name of a data object given the time offset
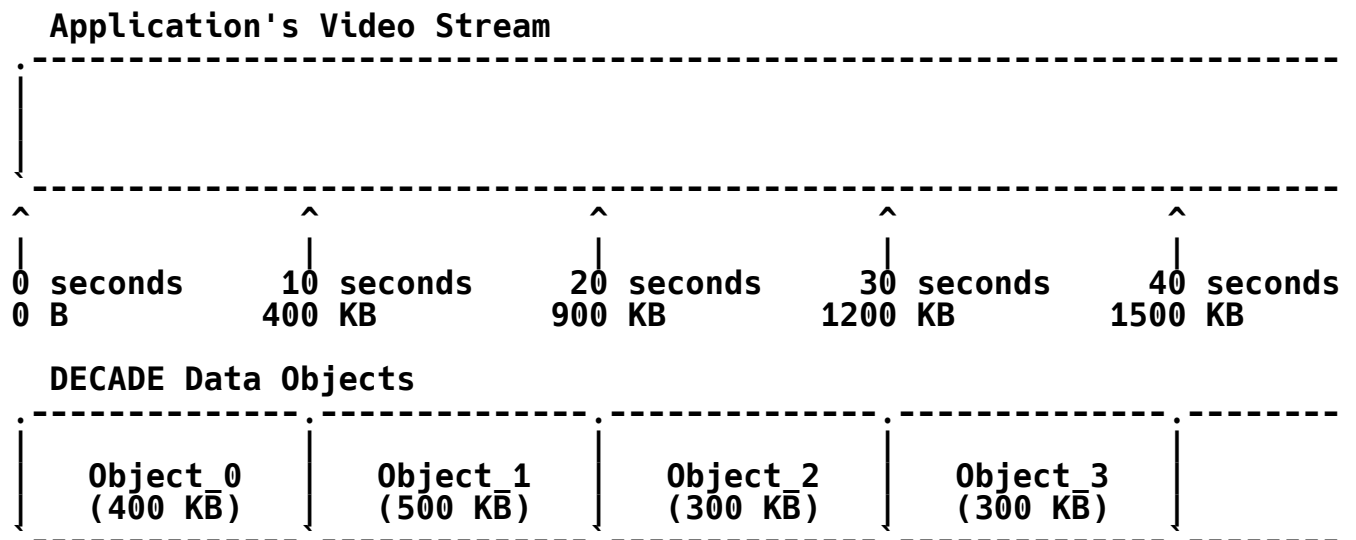of the video chunk.

```
   Application's Video Stream
    .-----------------------------------------------------------------
    |
    |
    |
    |
    .-----------------------------------------------------------------
   ^                 ^                 ^                 ^                 ^
   |                 |                 |                 |                 |
   0 seconds         10 seconds        20 seconds        30 seconds        40 seconds
   0 B               400 KB            900 KB            1200 KB           1500 KB

   DECADE Data Objects
    .-------------.-------------.-------------.-------------.--------
    |             |             |             |             |
    |   Object_0  |   Object_1  |   Object_2  |   Object_3  |
    |  (400 KB)   |  (500 KB)   |  (300 KB)   |  (300 KB)   |
    |             |             |             |             |
    .-------------.-------------.-------------.-------------.--------
```

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

5.5. Token-Based Authorization and Resource Control

   A key feature of a DECADE system is that an Application Endpoint can
   authorize other Application Endpoints to store or retrieve data
   objects from its in-network storage via tokens. The peer client then
   uses the token when sending requests to the DECADE server. Upon
   receiving a token, the server validates the signature and the
   operation being performed.

   This is a simple scheme, but has some important advantages over an
   alternative approach, for example, in which a client explicitly
   manipulates an Access Control List (ACL) associated with each data
   object. In particular, it has the following advantages when applied
   to DECADE systems. First, authorization policies are implemented
   within the application, thus the Application Endpoint explicitly
   controls when tokens are generated, to whom they are distributed, and
   for how long they will be valid. Second, fine-grained access and
   resource control can be applied to data objects. Third, there is no
   messaging between a client and server to manipulate data object
   permissions. This can simplify, in particular, applications that
   share data objects with many dynamic peers and need to frequently
   adjust access control policies attached to data objects. Finally,
   tokens can provide anonymous access, in which a server does not need
   to know the identity of each client that accesses it. This enables a
   client to send tokens to clients belonging to other storage
   providers, and to allow them to read or write data objects from the
   storage of its own storage provider. In addition to clients' ability
   to apply access control policies to data objects, the server may be

configured to apply additional policies based on user, object
properties, geographic location, etc. A client might thus be denied
access even though it possesses a valid token.

5.6. Discovery

A DECADE system should include a discovery mechanism through which
DECADE clients locate an appropriate DECADE server. A discovery
mechanism should allow a 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). (The
discovery mechanism might also result in an error if no such servers
can be located.) After discovering one or more servers, a DECADE
client can distribute load and requests across them (subject to
resource limitations and policies of the servers themselves)
according to the policies of the Application Endpoint in which it is
embedded. The discovery mechanism outlined here does not provide the
ability to locate arbitrary DECADE servers to which a client might
obtain tokens from others. To do so will require application-level
knowledge, and it is assumed that this functionality is implemented
in the content distribution application.

As noted above, the discovered DECADE server should be authorized to
allow the client to store data objects and then generate tokens to
allow other clients to retrieve these data objects. This
authorization may be:

- a result of off-line administrative procedures;

- access network dependent (e.g., all the subscribers to a
  particular ISP may be allowed by the ISP);

- due to a prior subscription;

- etc.

The particular protocol used for discovery is out of scope of this
document, but any specification should reuse well-known protocols
wherever possible.

6. DECADE Protocol Considerations

This section presents the DRP and the SDT protocol in terms of
abstract protocol interactions that are intended to be mapped to
specific protocols in an implementation. In general, the DRP/SDT
functionality for DECADE client-server interaction is very similar to
that for server-server interaction. Any differences are highlighted
below. DRP is used by a DECADE client to configure the resources and
authorization used to satisfy requests (reading, writing, and
management operations concerning data objects) at a server. SDT will
be used to transport data between a client and a server, as
illustrated in Figure 1.

6.1. Naming

A DECADE system SHOULD use [RFC6920] as the recommended and default
naming scheme. Other naming schemes that meet the guidelines in
Section 4.3 MAY alternatively be used. In order to provide a simple
and generic interface, the DECADE server will be responsible only for
storing and retrieving individual data objects.

The DECADE naming format SHOULD NOT attempt to replace any naming or
sequencing of data objects already performed by an application.
Instead, naming is intended to apply only to data objects referenced
by DECADE-specific purposes. An application using a DECADE client
may use a naming and sequencing scheme independent of DECADE names.
The DECADE client SHOULD maintain a mapping from its own data objects
and their names to the DECADE-specific 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.

6.2. Resource Protocol

DRP will provide configuration of access control and resource-sharing
policies on DECADE servers. A content distribution application
(e.g., a live P2P streaming session) can have permission to manage
data at several servers, for instance, servers belonging to different
storage providers. DRP allows one instance of such an application,
i.e., an Application Endpoint, to apply access control and resource-
sharing policies on each of them.

On a single DECADE server, the following resources SHOULD be managed:
a) communication resources in terms of bandwidth (upload/download)
and also in terms of number of active clients (simultaneous
connections); and b) storage resources.

6.2.1. Access and Resource Control Token

   The tokens SHOULD be generated by an entity trusted by both the
   DECADE client and the server at the request of a DECADE client. For
   example, this entity could be the client, a server trusted by the
   client, or another server managed by a storage provider and trusted
   by the client. It is important for a server to trust the entity
   generating the tokens since each token may incur a resource cost on
   the server when used. Likewise, it is important for a client to
   trust the entity generating the tokens since the tokens grant access
   to the data stored at the server.

   The token does not normally include information about the identity of
   the authorized client (i.e., it is typically an anonymous token).
   However, it is not prohibited to have a binding of the token to an
   identity if desired (e.g., binding of the token to the IP address of
   the authorized party).

   Upon generating a token, a DECADE client can distribute it to another
   client. Token confidentiality SHOULD be provided by whatever
   protocol it is carried in (i.e., Application Protocol, DRP, or SDT).
   The receiving client can then connect to the server specified in the
   token and perform any operation permitted by the token. The token
   SHOULD be sent along with the operation. The server SHOULD validate
   the token to identify the client that issued it and whether the
   requested operation is permitted by the contents of the token. If
   the token is successfully validated, the server SHOULD apply the
   resource control policies indicated in the token while performing the
   operation.

   Tokens SHOULD include a unique identifier to allow a server to detect
   when a token is used multiple times and reject the additional usage
   attempts. Since usage of a token incurs resource costs to a server
   (e.g., bandwidth and storage) and an uploading DECADE client may have
   a limited budget, the uploading DECADE client should be able to
   indicate if a token may be used multiple times.

   It SHOULD be possible to revoke tokens after they are generated.
   This could be accomplished by supplying the server the unique
   identifiers of the tokens that are to be revoked.

6.2.2. Status Information

   DRP SHOULD provide a status request service that clients can use to
   request status information of a server. Access to such status
   information SHOULD require client authorization; that is, clients
   need to be authorized to access the requested status information.
   This authorization is based on the user delegation concept as

described in Section 4.5. The following status information elements
SHOULD be obtained: a) list of associated data objects (with
properties); and b) resources used/available. In addition, the
following information elements MAY be available: c) list of servers
to which data objects have been distributed (in a certain time
frame); and d) list of clients to which data objects have been
distributed (in a certain time frame).

For the list of servers/clients to which data objects have been
distributed to, the server SHOULD be able to 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 may be used for accounting purposes, e.g., the list of
clients to which data objects have been distributed.

Access information MAY be provided for accounting purposes, for
example, when uploading DECADE clients are interested in access
statistics for resources and/or to perform accounting per user.
Again, access to such information requires client authorization and
SHOULD be based on the delegation concept as described in
Section 4.5. The following type of access information elements MAY
be requested: a) what data objects have been accessed by whom and how
many times; and b) access tokens that a server has seen for a given
data object.

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

6.2.3. Data Object Attributes

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

Depending on authorization, clients SHOULD be permitted to get or set
such attributes. This authorization is based on the delegation as
per Section 4.5. DECADE does not limit the set of permissible
attributes, but rather specifies a set of baseline attributes that
SHOULD be supported:

Expiration Time: time at which the data object can be deleted

Data Object size: in bytes

Media type: labeling of type as per [RFC6838]

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

The data object attributes defined here are distinct from application
metadata. Application metadata is custom information that an
application might 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, it can be
stored within data objects themselves.

6.3. Data Transfer

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

To write a data object, a client first generates the object's name
(see Section 6.1), and then uploads the object to a server and
supplies the generated name. The name can be used to access
(download) the object later; for example, the client can pass the
name as a reference to other clients that can then refer to the
object. Data 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. If supported, a server can verify the integrity and
other security properties of uploaded objects.

A client can request named data objects from a server. In a
corresponding request message, a client specifies the object name and
a suitable access and resource control token. The server checks the
validity of the received token and its associated properties related
to resource usage. If the named data object exists on the server and
the token can be validated, the server delivers the requested object
in a response message. If the data object cannot be delivered, the
server provides a corresponding status/reason information in a
response message. Specifics regarding error handling, including
additional error conditions (e.g., overload), precedence for returned
errors and its relation with server policy, are deferred to eventual
protocol specification.

6.4. Server-Server Protocols

An important feature of a DECADE system is the capability for one
server to directly download data objects from another 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 server.

DRP and SDT SHOULD support operations directly between servers.
Servers are not assumed to trust each other nor are they configured
to do so. All data operations are performed on behalf of clients via
explicit instruction. However, the objects being processed do not
necessarily have to originate or terminate at the client (i.e., the
data object might be limited to being exchanged between servers even
if the instruction is triggered by the client). Clients thus will be
able to indicate to a server which remote server(s) to access, what
operation is to be performed, or in which server the object is to be
stored, and the credentials indicating access and resource control to
perform the operation at the remote server.

Server-server support is focused on reading and writing data objects
between servers. The data object referred to at the remote server is
the same as the original data object requested by the client. Object
attributes might also be specified in the request to the remote
server. In this way, a server acts as a proxy for a client, and a
client can instantiate requests via that proxy. The operations will
be performed as if the original requester had its own client co-
located with the server. When a client sends a request to a server
with these additional parameters, it is giving the server permission
to act (proxy) on its behalf. Thus, it would be prudent 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 retrieval operation, the server is to retrieve the
data object from the remote server using the specified credentials,
and then optionally return the object to a client. In the case of a
storage operation, the server is to store the object to the remote
server using the specified credentials. The object might optionally
be uploaded from the client or might already exist at the server.

6.5. Potential DRP/SDT Candidates

Having covered the key DRP/SDT functionalities above, it is useful to
consider some potential DRP/SDT candidates as guidance for future
DECADE protocol implementations. To recap, the DRP is a protocol for
communication of access control and resource-scheduling policies from
a DECADE client to a DECADE server, or between DECADE servers. The

SDT is a protocol used to transfer data objects between a DECADE
client and DECADE server, or between DECADE servers. An evaluation
of existing protocols for their suitability for DRP and SDT is given
in Appendix A. Also, [INTEGRATION-EX] provides some experimental
examples of how to integrate DECADE-like in-network storage
infrastructure into P2P applications.

7. How In-Network Storage Components Map to DECADE

   This section evaluates how the basic components of an in-network
   storage system (see Section 3 of [RFC6392]) map into a DECADE system.

   With respect to the data access interface, DECADE clients can read
   and write objects of arbitrary size through the client's Data
   Controller, making use of standard data transfer (SDT). With respect
   to data management operations, clients can move or delete previously
   stored objects via the client's Data Controller, making use of SDT.
   Clients can enumerate or search contents of 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, Application Endpoints might consult their
   local Data Index in the client's Data Controller (Data Search
   Capability).

   With respect to 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 client's Resource
   Controller. The server is responsible for implementing the access
   control checks. Clients can manage the resources (e.g., bandwidth)
   on the DECADE server that can be used by other Application Endpoints
   (Resource Control Interface). Resource-sharing policies are
   generated by a content distribution application and provided to the
   client's Resource Controller. The server is responsible for
   implementing the resource-sharing policies.

   Although the particular protocol used for discovery is outside the
   scope of this document, different options and considerations have
   been discussed in Section 5.6. Finally, with respect to the storage
   mode, DECADE servers provide an object-based storage mode. Immutable
   data objects might be stored at a server. Applications might
   consider existing blocks as data objects, or they might adjust block
   sizes before storing in a server.

8. Security Considerations

   In general, the security considerations mentioned in [RFC6646] apply
   to this document as well. A DECADE system provides a distributed
   storage service for content distribution and similar applications.
   The system consists of servers and clients that use these servers to
   upload data objects, to request distribution of data objects, and to
   download data objects. Such a system is employed in an overall
   application context (for example, in a P2P application), and it is
   expected that DECADE clients take part in application-specific
   communication sessions. The security considerations here focus on
   threats related to the DECADE system and its communication services,
   i.e., the DRP/SDT protocols that have been described in an abstract
   fashion in this document.

8.1. Threat: System Denial-of-Service Attacks

   A DECADE network might be used to distribute data objects from one
   client to a set of servers using the server-server communication
   feature that a client can request when uploading an object. Multiple
   clients uploading many objects at different servers at the same time
   and requesting server-server distribution for them could thus mount
   massive distributed denial-of-service (DDOS) attacks, overloading a
   network of servers. This threat is addressed by the server's access
   control and resource control framework. Servers can require
   Application Endpoints to be authorized to store and to download
   objects, and Application Endpoints can delegate authorization to
   other Application Endpoints using the token mechanism. Of course the
   effective security of this approach depends on the strength of the
   token mechanism. See below for a discussion of this and related
   communication security threats.

   Denial-of-service attacks against a single server (directing many
   requests to that server) might still lead to considerable load for
   processing requests and invalidating tokens. SDT therefore MUST
   provide a redirection mechanism to allow requests to other servers.
   Analogous to how an HTTP reverse proxy can redirect and load balance
   across multiple HTTP origin servers [RFC2616].

8.2. Threat: Authorization Mechanisms Compromised

   A DECADE system does not require Application Endpoints to
   authenticate in order to access a server for downloading objects,
   since authorization is not based on Endpoint or user identities but
   on a delegation-based authorization mechanism. Hence, most protocol
   security threats are related to the authorization scheme. The
   security of the token mechanism depends on the strength of the token
   mechanism and on the secrecy of the tokens. A token can represent

authorization to store a certain amount of data, to download certain
objects, to download a certain amount of data per time, etc. If it
is possible for an attacker to guess, construct, or simply obtain
tokens, the integrity of the data maintained by the servers is
compromised.

This is a general security threat that applies to authorization
delegation schemes. Specifications of existing delegation schemes
such as [RFC6749] discuss these general threats in detail. We can
say that the DRP has to specify appropriate algorithms for token
generation. Moreover, authorization tokens should have a limited
validity period that should be specified by the application. Token
confidentiality should be provided by application protocols that
carry tokens, and the SDT and DRP should provide secure
(confidential) communication modes.

8.3. Threat: Spoofing of Data Objects

In a DECADE system, an Application Endpoint is referring other
Application Endpoints to servers to download a specified data object.
An attacker could "inject" a faked version of the object into this
process, so that the downloading Endpoint effectively receives a
different object (compared to what the uploading Endpoint provided).
As a result, the downloading Endpoint believes that is has received
an object that corresponds to the name it was provided earlier,
whereas in fact it is a faked object. Corresponding attacks could be
mounted against the application protocol (that is used for referring
other Endpoints to servers), servers themselves (and their storage
subsystems), and the SDT by which the object is uploaded,
distributed, and downloaded.

A DECADE systems fundamental mechanism against object spoofing is
name-object binding validation, i.e., the ability of a receiver to
check whether the name it was provided and that it used to request an
object actually corresponds to the bits it received. As described
above, this allows for different forms of name-object binding, for
example, using hashes of data objects, with different hash functions
(different algorithms, different digest lengths). For those
application scenarios where hashes of data objects are not applicable
(for example, live streaming), other forms of name-object binding can
be used. This flexibility also addresses cryptographic algorithm
evolution: hash functions might get deprecated, better alternatives
might be invented, etc., so that applications can choose appropriate
mechanisms that meet their security requirements.

DECADE servers MAY perform name-object binding validation on stored objects, but Application Endpoints MUST NOT rely on that. In other words, Application Endpoints SHOULD perform name-object binding validation on received objects.

9. Acknowledgments

We thank the following people for their contributions to and/or detailed reviews of this document or earlier drafts of this document: Carlos Bernardos, Carsten Bormann, David Bryan, Dave Crocker, Yingjie Gu, David Harrington, Hongqiang (Harry) Liu, David McDysan, Borje Ohlman, Martin Stiemerling, Richard Woundy, and Ning Zong.

10. References

10.1. Normative References

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

10.2. 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.

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

   [RFC5754] Turner, S., "Using SHA2 Algorithms with Cryptographic
              Message Syntax", RFC 5754, January 2010.

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

   [RFC6646] Song, H., Zong, N., Yang, Y., and R. Alimi, "DECoupled
              Application Data Enroute (DECADE) Problem Statement", RFC
              6646, July 2012.

   [RFC6749] Hardt, D., "The OAuth 2.0 Authorization Framework", RFC
              6749, October 2012.

   [RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type
              Specifications and Registration Procedures", BCP 13, RFC
              6838, January 2013.

   [RFC6920] Farrell, S., Kutscher, D., Dannewitz, C., Ohlman, B.,
              Keranen, A., and P. Hallam-Baker, "Naming Things with
              Hashes", RFC 6920, April 2013.

   [INTEGRATION-EX]
              Zong, N., Ed., Chen, X., Huang, Z., Chen, L., and H. Liu,
              "Integration Examples of DECADE System", Work in Progress,
              August 2013.

   [GoogleFileSystem]
              Ghemawat, S., Gobioff, H., and S. Leung, "The Google File
              System", SOSP '03, Proceedings of the 19th ACM Symposium
              on Operating Systems Principles, October 2003.

   [GoogleStorageDevGuide]
              Google, "Google Cloud Storage - Developer's Guide",
              <https://developers.google.com/storage/docs/
              concepts-techniques>.

   [OpenFlow]
              Open Networking Foundation, "Software-Defined Networking:
              The New Norm for Networks", April 2013,
              <https://www.opennetworking.org/images/stories/downloads/
              sdn-resources/white-papers/wp-sdn-newnorm.pdf>.

   [CDMI]   Storage Networking Industry Association (SNIA), "Cloud
              Data Management Interface (CDMI (TM)), Version 1.0.2",
              June 2012,
              <http://snia.org/sites/default/files/CDMI%20v1.0.2.pdf>.

Appendix A. Evaluation of Candidate Protocols for DECADE DRP/SDT

   In this section we evaluate how well the abstract protocol
   interactions specified in this document for DECADE DRP and SDT can be
   fulfilled by the existing protocols of HTTP, CDMI, and OAuth.

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 DRP 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. Resource Control Primitives for Communication

   Communication resources include bandwidth (upload/download) and the
   number of simultaneously 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 have direct support for controlling the communication
   resources for a given client. However, servers typically perform
   this function via implementation algorithms.

A.1.1.3. Resource Control Primitives for Storage

   Storage resources include the amount of memory and lifetime of
   storage. HTTP does not allow direct control of storage at the server
   endpoint. 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 SDT 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, not the
   resource itself.

A.1.3. Primitives for Removing Duplicate Traffic

   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 that 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
a URL. In the reply, the hash is sent in an Entity Tag (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 it 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 endpoint 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 [RFC6749] (for access
control) in combination with HTTP [GoogleStorageDevGuide]. An
example of using OAuth for DRP is given in Appendix A.3.

A.2. 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.2.1. CDMI Support for DRP Primitives

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

A.2.1.1. Access Control Primitives

Access control includes mechanisms for defining the access policies
for the server and then checking the authorization of a user before
allowing content storage or retrieval. CDMI defines an Access
Control List (ACL) per data object and thus supports access control
(read and/or write) at the granularity of data objects. 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.2.1.2. Resource Control Primitives for Communication

Communication resources include bandwidth (upload/download) and the
number of simultaneously connected clients (connections). CDMI
supports two key data attributes that 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.2.1.3. Resource Control Primitives for Storage

Storage resources include amount of quantity and lifetime of storage.
CDMI defines metadata for individual data objects and general storage
system configuration that 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

-cdmi_retention_autodelete: whether object should be automatically
   deleted after retention period

A.2.2. CDMI Support for SDT 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. CDMI supports
   standard HTTP methods for PUT and POST as described in
   Appendix A.1.2.1.

A.2.2.2. 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.2.3. 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 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.2.4. 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

A.3. OAuth

   As mentioned in Appendix A.1, OAuth [RFC6749] may be used as part of
   the access and resource control of a DECADE system. In this section,
   we provide an example of how to configure OAuth requests and
   responses for DRP.

   An OAuth request to access DECADE data objects should include the
   following fields:

      response_type: Value should be set to "token".

      client_id: The client_id indicates either the application that is
      using the DECADE service or the end user who is using the DECADE
      service from a DECADE storage service provider. DECADE storage
      service providers should provide the ID distribution and
      management function.

      scope: Data object names that are requested.

   An OAuth response should include the following information:

      token_type: "Bearer"

      expires_in: The lifetime in seconds of the access token.

      access_token: A token denotes the following information.

      service_uri: The server address or URI which is providing the
      service;

      permitted_operations (e.g., read, write) and objects (e.g., names
      of data objects that might be read or written);

      priority: Value should be set to be either "Urgent", "High",
      "Normal" or "Low".

      bandwidth: Given to requested operation, a weight value used in a
      weighted bandwidth sharing scheme, or an integer in number of bits
      per second;

      amount: Data size in number of bytes that might be read or
      written.

      token_signature: The signature of the access token.

Authors' Addresses

   Richard Alimi
   Google

   EMail: ralimi@google.com

   Akbar Rahman
   InterDigital Communications, LLC

   EMail: akbar.rahman@interdigital.com

   Dirk Kutscher
   NEC

   EMail: dirk.kutscher@neclab.eu

   Y. Richard Yang
   Yale University

   EMail: yry@cs.yale.edu

   Haibin Song
   Huawei Technologies

   EMail: haibin.song@huawei.com

   Kostas Pentikousis
   EICT

   EMail: k.pentikousis@eict.de