DECADE
Internet-Draft
Intended status: Informational
Expires: September 13, 2012

H. Liu
Yale University

Mar 12, 2012

# Data Streaming Subscription in DECADE
## draft-liu-decade-subscription-01

Abstract

This document describes a potential performance issue in DECADE to support real-time applications. It shows the current content hashing based naming scheme might harm the performance when the content data is generated in real time and low propagation latency of the data is required. This draft propose a new naming scheme and protocol, subscribe/unsubscribe, for real-time applications to address the problem caused by content hashing. DECADE clients use subscribe/unsubscribe to express their interests to receive data streaming. DECADE servers pro-actively push stream data to eligible clients, which saves the time consumed on exchanging the hashing of new generated data.

Table of Contents

# 1. Introduction

Existing DECADE protocol identifies all the objects by their content hashes. A client cannot fetch an object from DECADE servers until it learns the hash of this object. This property introduces additional delays in "real-time applications", e.g. live streaming, video conference, etc., in which data is generated on the fly and short latency is required to distribute data to clients.

This draft discusses about the performance loss of real-time applications with existing DECADE protocol. Moreover it presents a new mechanism "subscribe/unsubscribe" in DECADE framework to better support real-time applications.

## 1.1. Concepts

### 1.1.1. Real-time Applications

The applications that have a source generating live data and deliver the data to the receivers with latency as short as possible.

### 1.1.2. DECADE Sender

An end host which uploads data to a DECADE server and leverages DECADE framework to deliver the data to one or multiple DECADE receivers.

### 1.1.3. DECADE Receiver

An end host which downloads data from DECADE servers.

# 2. Problems of Content Hash Only Naming for Real-time Applications

In naming schemes only based on content hashing, an object is identified uniquely by its hash of its content. In other words, if a client wants to download an object, it should firstly learn the object's hash value and send a request to DECADE servers with this hash. This works well in "offline" applications, such as file sharing and video on demand, in which the objects and their hashes already exist, and the object delivery performance is not delay sensitive. However, it will introduces significant additional delays which harms "real-time" application performance.

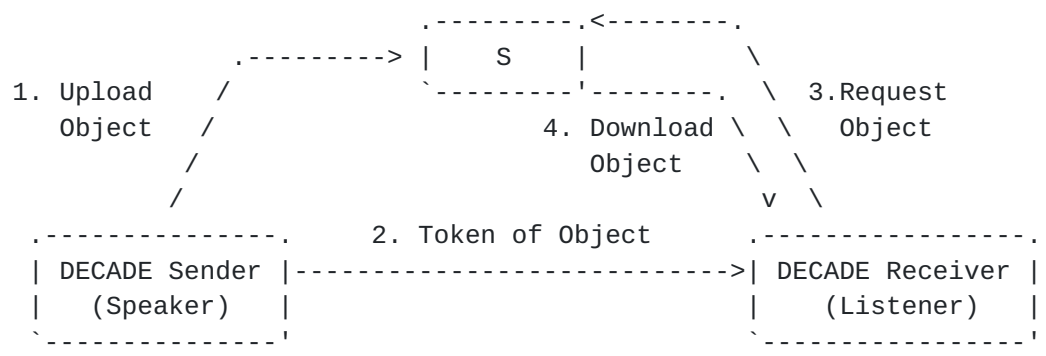


Figure 1: Real-time Object Distribution with Traditional DECADE

Figure 1 illustrates a simple case of voice over IP (VoIP) using content hash only naming. When the client on the left is a speaker. It wants to upload only one copy of its voice data to DECADE server S and asks S to upload the data to multiple listeners. The work flow is as the following: (1) The speaker buffers its voice data until its size meets the minimum object size requirement in DECADE (or in the application). Denote D_buffer as the delay caused by the buffering; (2) The speaker uploads the object to S (with delay D_up) and in the meanwhile sends the token to the receiver (with delay D_token); (3) With the token, the listener requests and downloads the object from S (with delay D_down which is the RTT between S and the listener). In summary, the total latency from the moment a piece of voice is generated to the moment this piece reaches the listener side is D_1 = D_buffer + max(D_up, D_token) + D_down.

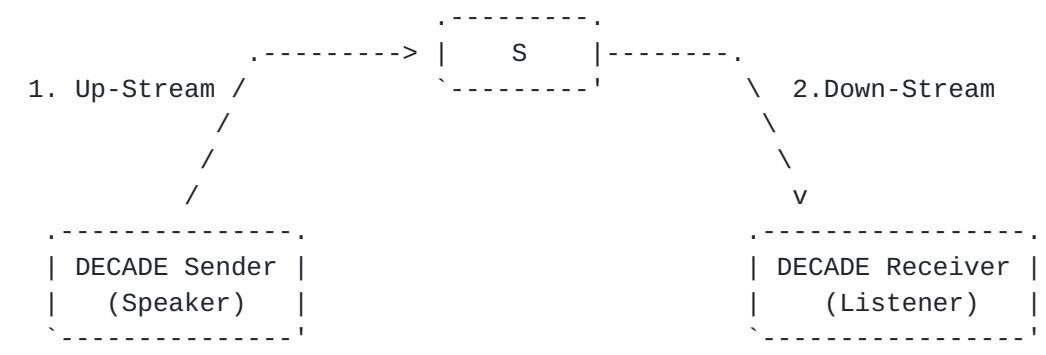


Figure 2: Real-time Streaming with A Simple Forwarding Server

Compare with an alternative design shown in Figure 2. If the speaker just pushes data to S and S pushes data to listenrs, the total latency is only D_2 = D_up + D_down/2. D1 - D2 >= D_buffer + D_down/2. In reality, D1 can be much larger than D2. In paricular, D1 can be larger than typical delay threshold (say 200 ms) for VoIP's continulity, while D2 is not.

Therefore, to better support real-time applications in DECADE, we propose a new protocol flow to achieve the ideal case in Figure 2 compatibly with existing DECADE framework and with scalability and manageability.

## 3. Subscribe/Unsubscribe

A stream is a special object whose length we do not know and whose hash does not exist. This is the fundamental reason that content-hash only naming cannot support live streaming well. In this section we propose a new type of protocol subscribe/unsubscribe to support streams in DECADE. There are two key ideas in this new protocol: (1) DECADE servers push stream data to the subscribers; (2) Tokens are used to authorize the eligibility of stream subscriptions.

### 3.1. Work Flow of Stream Subscription

Figure 3 illustrates an example of how subscribe/unsubscribe works. At the beginning, the speaker on the left sends a stream creating message to its DECADE server S1 and S1 will allocate a stream ID for the speaker. Then, the speaker uses this stream ID to construct tokens and upload stream data to S1. After the listener gets the stream token, it presents the token in its subscription of the stream. Its DECADE server S2, if S2 is not S1, will recursively subscribe the stream from S1 with the token. Finally if the subscription is authorized, S1 will forward the stream to S2 and S2 in turn push the stream to the listener. For scalability and robustness, a stream token might only valid for a duration. The listener should update the stream token periodically from the speaker. When the listener does not want to receive the stream any more, it unsubscribes the stream from S2. When S2 finds there is no subscribers for the stream, it will unsubscribe the stream from S1.

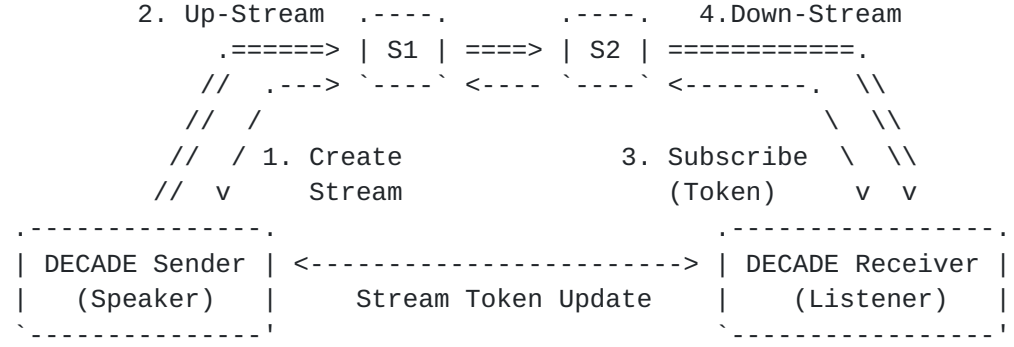


Figure 3: Work Flow of Stream Subscription

### 3.2. Stream ID

Each stream must has a unique ID. One possible solution is that each DECADE provider owns a unique prefix for the streams created on its DECADE servers. And the DECADE provider guarantees the uniqueness of streams inside its DECADE network itself.

The requirement for the length of stream ID is the same with the object ID defined in [I-D.ietf-decade-arch]

### 3.3. Stream Token

Stream Tokens have the same format of DECADE token defined in [I-D.ietf-decade-arch]. In streaming tokens, the "object ID" part is actually stream ID. DECADE servers use stream token to figure out when a subscriber's account is eligible to receive the stream.

### 3.4. Stream Forwarding Table

The DECADE server creates a streaming forwarding table to manage and forward streams. When a DECADE client creates a stream in its DECADE server, it actually sets up an item in the server's stream forwarding table. As shown in Figure 4, the key of each item in the table is stream ID. With a stream ID, the server can obtain the owner of the stream which is used to authorize tokens in subscriptions and account resource usage. The server can also learn the current subscribers of the stream and decide where to forward a stream or when to unsubscribe a stream (after the subscriber list becomes empty.)

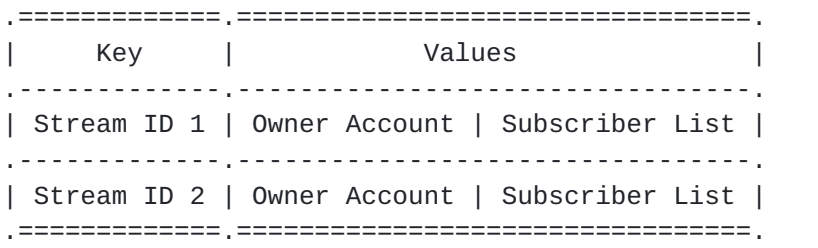


Figure 4: Stream Forwarding Table in DECADE Servers

Note that only the DECADE server or DECADE provider (e.g. S1 in Figure 3) on which a stream is created has the stream owner account information. For other DECADE servers or providers (e.g. S2 in Figure 3), they only maintain subscribers who subscribe the stream via themselves.

## 4. Protocol

This section shows an example of DECADE subscription protocol on HTTP.

As shown in Figure 5, in the subscription request, the client puts the stream ID into the URL. The "Connection" header should be "Keep-Alive" to establish a persistent with the DECADE server. There are two new HTTP headers for DECADE: (1) "Decade-Origin" which indicates the backup location where the stream can be fetched, and (2) "Decade-Token" which presents the token for access the stream to the DECADE server.

The server status uses code 200 (OK), 404 (Not Found) or 401 (Unauthorized) to tell the clients the result of the subscription. If the subscription is accepted, the server will push streaming data to the client. Otherwise, the server closes the connection.

```
GET /StreamID HTTP/1.1
Connection: Keep-Alive
Host: server1.decade.net
Decade-Origin: http://server2.decade.net
Decade-Token: TWFuIGlzIGRpc3Rpbmd
```

Figure 5: A Message of DECADE Streaming Subscription Request

```
HTTP/1.1 200 OK
Connection: Keep-Alive
Content-Type: application/octet-stream

streaming data
```

Figure 6: A Message of DECADE Streaming Subscription Response

## 5. Security Considerations

This document does not contain any security considerations.

## 6. IANA Considerations

This document does not have any IANA considerations.

# 7. References

## 7.1. Normative References

## 7.2. 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-05 (work in progress), February 2012.

[I-D.ietf-decade-survey]
Alimi, R., Rahman, A., and Y. Yang, "A Survey of In-network Storage Systems", draft-ietf-decade-survey-06 (work in progress), August 2011.

[I-D.ietf-decade-arch]
Alimi, R., Yang, Y., Rahman, A., Kutscher, D., and H. Liu, "DECADE Architecture", draft-ietf-decade-arch-04 (work in progress), October 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.

Author's Address

Hongqiang Liu
Yale University

Email: hongqiang.liu@yale.edu