Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 10, 2015

Y. Tsuzaki
Kyoto University
R. Atarashi
IIJ Innovation Institute Inc.
S. Suzuki
Keio University
K. Mitsuya
K. Okada
Lepidum Co. Ltd.

March 09, 2015

Network configuration Web API for Bandwidth Reservation
draft-tsuzaki-netconfig-webapi-00.txt

Abstract

This draft introduces a framework for a dynamic bandwidth reservation via Web API for Web applications. In this document, we propose Web APIs for Web clients to request bandwidth allocation to network controllers. The network controller could be both of SDN compliant or Non-SDN compliant one. In this document, a network specification definition language is also proposed.

Table of Contents

1. Introduction

This draft proposes a framework for a dynamic bandwidth reservation via Web API for Web applications. We assume that there are network controllers to control the network devices and gather information about their control domain. Those controllers equip Web APIs so that Web clients can request bandwidth allocated virtual private paths between contents Web servers and the clients. Network administrators describe the service specifications with "service description language", and the bandwidth are allocated to the clients according to the service specifications. This draft explains the overview of this architecture and how resource reservations are made.

2. Requirement

- From the Viewpoint of Network Administrators
  Based on the service specifications configured by the administrators, network management controllers automatically respond to the client requests via Web APIs.

- From the Viewpoint of Clients
  By accessing the Web API for the network resource reservations, clients can reserve QoS guaranteed communication bandwidth for Web contents downloads.

- Use Case
  The network administrators prepare Web APIs for configuring network paths and bandwidth reservations. When a client need to download large contents from a Web server, the client send the requiring resource information to the network management server via Web APIs. The network management server construct a QoS guaranteed communication path for the client based on the information received from the client.

3. Terminology

- Management Server: Servers which control the network devices in a domain. These servers also provide the application interfaces for Media Clients to signal resource requests. Administrators describe the network configurations and policies of networks by SDL/NDL and put them to Management Servers. Management servers are also referred as Network Management Servers.

- Media server: Kind of a web server, which delivers media contents to Media Clients.

- Media client: Client application run on a Web Browser, which receives and present media contents to an end user.

- Service specification: the description of network service components described by SDL/NDL

- Service Description Language (SDL): A language by which administrators describes network device information. Administrators describe SDL and put the descriptions to Management Servers.

- Network Description Language (NDL): A language by which administrators describes network service information. Administrators describe NDL and put the descriptions to Management Servers.

   o Resource request: action by which Media Clients obtain resource
      reserved communication path to Media Servers.

4. System architecture

```
                    +-----------------+
                    |   Management    |
         Resource   |     server      |
         Request  +-+---+-------+--+-+
         +------> | WEB API |   |  |
         |        +---------+   |  |  Path
         |                      |  |  Setup
         |                      |  |
         |              , - \+  +/- ,
         |          , '      \  /     ' ,
         |        ,           \/          ,
    +----+---+   +--------------------------------+   +--------+
    |  Media |            Reserved                    |  Media |
    | client |              Path                      | server |
    +--------+   +--------------------------------+   +--------+
                   ,         SDN/Non-SDN          ,
                    ,          BackBone          ,
                                                ,'
                      ,                      , '
                        ' ,  + , _ _ _ ,  '
```

Figure 1: System Architecture

Figure 1 depicts the system overview of application triggered resource reservation architecture. All the network components except for the end clients in the domain (routers, servers) are under the control of the management server. The network management server gathers network management information such as status of network devices or links in the network, and also command those devices to set up QoS guaranteed communication paths between Media Servers and Media Clients.

The scope of this architecture is to define Web interfaces to signal resource allocation from Web browser applications to management servers.

4.1. Management server

Management servers are servers that control network components on the networks. Network administrators describe network device groups and network service description language with language called "service definition language". Service definition language is detailed in Section 5.1.

Management servers serves Web APIs for clients to make resource reservations. To trigger network resource reservations, Media Clients access these Web APIs on the management servers. Upon receiving requests from Media Clients, a management server calculate appropriate communication paths between Media Servers and Media Clients. The intermediate nodes (routers or switches) can be both of SDN compliant and Non-SDN compliant devices, but each of those devices have to be configurable by the management server via some remote configuration methods such as Netconf[RFC6241] or SSH.

4.2. Media client

Media client is a client application program run on a Web browser, which receives and present media contents to an end user. Media client receives Media Program, which is a list of contents can be presented, from a Program Server. When the user selects a content from the presented list of contents, Media client start playing the content.

4.3. Program Server

Program Server store and provides a Media Program, which is a list of available contents. We use HTTP to provide a Media Program to a Media Client.

The content specified in the Media Program consists of the title of the content and URL of the content. We expect content URL point to a location of a MEPGDash[MPEGDASH] file. The Media Program can be generated either by statically or dynamically.

At this moment, we do not define how Media client finds a Program Server. We assume this information is already available in Media client.

4.4. Media Server

Media server is a server program which store and provide metadata of a program as a MPEGDash format, and the contents of each media referenced from the MPEGDash formatted metadata.

Contents can be split into multiple segments by duration, or prepared in multiple bit rates.

Since the links between Media Program, MPEGDash file and segmented contents are described as a URL, all types of contents can store on one Media Server or among multiple Media Servers.

4.5. System components

```
   +----------------+        +--------------+
   |Media Client    |        |Program Server|
   +----------------+        +--------------+
   | +------------+ |        | +---------+  |
   | |  Program   +----------> | Program |  |
   | |Information | |        | |   List  |  |
   | |  Manager   +--------+ | +---------+  |
   | +--+---------+ |      | +--------------+
   |    |           |      |
   |    v           |      | +--------------+
   | +------------+ |      | |Media Server  |
   | |  Resource  | |      | +--------------+
   | |  Manager   +----+   | | +----------+ |
   | +--+---------+ |  |   +-> |  Media   | |
   +----|------^----+  +-----> | Contents | |
        |      |             | +----------+ |
        |      +-------+     +--------------+
        |              |
   +----|--------------|--------------------+
   |    |              |  Management Server |
   +----v--------------|--------------------+
   |  +------+       +-+-------+            |
   |  | Web  |       | Client  |            |
   |  | APIs |       | Manager |            |
   |  +------+       +---------+            |
   |        +---------+     ^               |
   |                  v     |               |
   |  +----------+   +------+-----+         |
   |  | Topology +-->| Topology    |         |
   |  | Database |   | Calculator |         |
   |  +----------+   +------------+         |
   +----------------------------------------+
```

Figure 2: System component

Figure 2 shows a simple component diagram of this architecture. When a Media Client starts to obtain media contents from Media Servers, the program information manager of the client first get the program list from program servers. The program lists are described in the media presentation description(MPD) format of MPEGDash. The resource manager then access to the resource usage request Web API on the management server. When received a request from a client, the management server calculate the allocatable bandwidth between the Media Client and the Media Server via the topology calculator. Then the client manager of the management server respond a resource usage report to the Media Client. Based on the information in the resource usage report, the Media Client trigger resource allocation by accessing the resource request Web API on the management server. Then the management server allocate bandwidth to the client via the topology calculator and send success message back to the client.

5. API

5.1. Service definition language

Network administrators define the service specifications utilizing the service specification language, Network Description Language(NDL) and Service Description Language(SDL). NDL is to define the group of network components such as router groups. With SDL, administrators can define service specifications on the network. Service specifications are the descriptions which define the relationship between network devices or network groups that compose network services. Examples of service definitions are network configurations such as segment IP address blocks or VLAN id for the segment. An example of NDL/SDL is shown in Figure 3 and Figure 4

```
node {
  ovs1;
  ovs2;
  media1;
  media2;
  pc11;
  pc12;
  pc13;
  pc14;
  pc21;
  pc22;
  pc23;
  pc24;
}
location {
  loc1 {
    media1;
    ovs1;
    pc11;
    pc12;
    pc13;
    pc14;
  }
  loc2 {
    media2;
    ovs2;
    pc21;
    pc22;
```

```
        pc23;
        pc24;
      }
    }
    group {
      group101 {
        media1;
        media2;
        pc11;
        pc12;
        pc13;
        pc14;
        pc21;
        pc22;
        pc23;
        pc24;
        ovs1;
        ovs2;
      }
      group1623 {
        ovs1;
        ovs2;
      }
      group1624 {
        ovs1;
        ovs2;
      }
      group1625 {
        ovs1;
        ovs2;
      }
    }
    link {
      type = layer1;
      edge1 = pc11;
      edge2 = pc12;
    }
```

Figure 3: Example of NDL

```
networks {
  network group101 {
    address = "192.168.1.0/24";
    vlan = 101;

    device ovs1 {
      type = L2Switch;
      address = "192.168.1.1";
    }

    device ovs2 {
      type = L2Switch;
      address = "192.168.1.2";
    }

    device media1 {
      type = Server;
      address = "192.168.1.30";
    }

    device media2 {
      type = Server;
      address = "102.168.1.31";
    }
  }
}
```

Figure 4: Example of SDL

SDL also enables registrations of events on the network and event bound actions. For example, if the traffics from certain source IP address exceeds the defined per-flow bandwidth limitation on the certain physical link, the traffic can be automatically shaped according to the definitions of SDL. Administrators define resource usage limitation using this functionality of SDL. For example, administrators can limit the usage of bandwidth per the domain to which user equipments attached. The bandwidth allocation for each user is determined based on these service specifications.

5.2. Web API

5.2.1. Resource usage report

Media servers advertise resource usage of links to Media Servers. The resource usage reports have two types. One is periodic resource usage reports broadcasted from management servers. Periodic usage reports include the uplink bandwidth usage of each servers(Figure 5). Another resource usage type is solicited usage report which is delivered to clients through WebAPI on the management servers. In a solicited usage report request(Figure 6), a Media Client specify the server from which it want to download media contents. The Media Server which received the solicited usage reports calculates the physical link set which connect the client and the server, and report available bandwidth the management server afford to allocate to the client(Figure 7). If multiple paths between the client and the server exist, the max available bandwidth will be returned to the client. At a solicited resource usage report request, a Media Client opens a web socket to the management server.

```
{
  [
    {
      "server": <String>
      "resource": {
        "bandwidth": <Num> // Option
        "latency": <Num> // Option
      }
    },
    ...
  ]
}
```

Figure 5: Unsolicited resource usage report json format

o server: server IP address or FQDN in string

o resource: available resource of the server

```
{

  "from": <String>
  "to": <String>
}
```

Figure 6: Solicited resource usage report request json format

o from: from IP address or FQDN in string

o to: to IP address or FQDN in string

```
{

  "resource": {
    "bandwidth": <Num> // Option
    "latency": <Num> // Option
  }
}
```


Figure 7: Solicited resource usage report response json format

- o resource: available resource of the server

5.2.2. Resource request

Media clients acquire reserved communication paths by accessing resource requests API on the management server. The resource requests have three types, initial resource request, resource modification request from clients and management server trigger resource modification request. We explain these types of resource requests in this section. According to the session_id information in the request, management server associate the web socket object of the request source client and the session-id.

The clients post json format requests on the reservation. Figure 8 is the format of the resource request json.


```
{
  "session_id": <String>
  "class": <Num>
  "type": <Num>
  "server": <String>
  "resource": {
    "bandwidth": <Num> // Option
    "latency": <Num> // Option
  }
}
```


Figure 8: Resource request json format

- o session_id: random created UUID to identify the session
- o class: user priority class in digit number
- o type: 0: Initial 1: Modification
- o server: the server to which the client willing to connect
- o resource: resource object contains bandwidth and latency

5.2.2.1. Initial resource request

```
   +------+      +------------+      +------+
   |Media |      | Management |      |Media |
   |Client|      |   Server   |      |Server|
   +--+---+      +------------+      +------+
      |                                  |
      |                                  |

      |                 |                |
      |      RRRQ       |                |
      +---------------> |                |
      |                 +                |
      |                                  |
      |               RRRQR              |
      +--------------------------------> |
      |                                  |
      |                 +      RRRQ      |
      |                 | <--------------+
      |                 |                |
      |                 |      RRRS      |
      |                 +--------------> |
      |                 |                |
      |       RRRS      |                |
      | <---------------+                |
      |                 +                |
      |               RRRQS              |
      | <--------------------------------+
      |                                  |
      +                                  +
```

Figure 9: Initial resource request sequence

- RURR: Resource Usage Report Request
- RURA: Resource Usage Report Advertisement
- RRRQ: Resource Reservation ReQuest
- RRRS: Resource Reservation ReSponse
- RRRQR: Resource Reservation ReQuest Request
- RRRQS: Resource Reservation ReQuest Response

A Media Client initially obtains a contents list on the Media Server. This contents list is described in the media presentation description (MPD) format of MpegDash. The acquisition of contents list is done by ordinal HTTP GET method. Then the client request resource usage reports to the management server as mentioned in Section 5.2.1. Based on information in the resource usage report and contents list, the client determine the contents bitrate and send a resource request to the management server based on the determined contents bitrate. The resource request contains a session-id randomly generated on clients(ex) UUID). The client simultaneously send a resource reservation request request to Media Server to trigger Media Server to send a request to the management server. The RRRQR also contains same session-id as resource request. The management server verify the request from the Media Server and the Media Client, and send response to both side if the information from the client and the server correspond. The management server stores the session-id, web socket information and allocated resources. These information are used for resource modifications and keep-alive. After received RRRS indicating the resource reservation was done successfully, the Media Server send RRRQS to the Media Client. Then the client get to be able to download the media contents with guaranteed quality.

5.2.2.2. Client trigger resource modification request

A Media Client MAY offer resource modification requests when resource usage reports say the uplink capacity of the Media Server from which the client downloads the media contents.

```
   +------+      +------------+      +------+
   |Media |      | Management |      |Media |
   |Client|      |   Server   |      |Server|
   +--+---+      +------------+      +------+
      |                                  |
      |      RRRQ       +                |
      +---------------> |                |
      |                 +                |
      |                                  |
      |                RRRQR             |
      +--------------------------------> |
      |                                  |
      |                 +      RRRQ      |
      |                 | <--------------+
      |                 |                |
      |                 |      RRRS      |
      |                 +--------------> |
      |                 |                |
      |       RRRS      |                |
      | <---------------+                |
      |                 +                |
      |                RRRQS             |
      | <--------------------------------+
      |                                  |
      +                                  +
```

        Figure 10: Resource modification sequence (client trigger)

5.2.2.3. Management server trigger resource modification request

```
   +------+      +------------+      +------+
   |Media |      | Management |      |Media |
   |Client|      |   Server   |      |Server|
   +--+---+      +------------+      +------+
      |                                  |
      |      RRMRQ      +                |
      | <---------------+                |
      |                 |                |
      |      RRRQ       |                |
      +---------------> |                |
      |                 +                |
      |                                  |
      |               RRRQR              |
      +-------------------------------> |
      |                                  |
      |                 +      RRRQ      |
      |                 | <--------------+
      |                 |                |
      |                 |      RRRS      |
      |                 +--------------> |
      |                 |                |
      |       RRRS      |                |
      | <---------------+                |
      |                 +                |
      |               RRRQS              |
      | <--------------------------------+
      |                                  |
      +                                  +
```

Figure 11: Resource modification sequence (management server trigger)

RRMRQ: Resource reservation modification request.

Management servers MAY trigger resource downgrade/upgrade requests to Media Clients, when the used bandwidth of a certain link saturate or become to have room. This push messaging can be done by Web sockets or WebRTC. As resource reservation modification request contains available resource for the received client, client determine the contents bitrate based on the information contained in RRMRQ and pre-downloaded contents list information at the initial resource reservation. The following process is similar to the initial resource reservation described in Section 5.2.2.1.

5.2.2.4. Resource cancellation

When a client do not need the allocated resources, the client can explicitly stop using the resource by posting a json described in Figure 12. Upon receiving cancellation message, the management server disassociate session-id from the client websocket, and release the resource bound to the session-id.


```
{
  "session_id": <String>
}
```


Figure 12: Resource cancel json format

5.2.3. Keep-alive

Management servers MAY keep-alive the clients to keep monitoring the usage of the reserved resources. While the clients can send resource free messages explicitly at the end of media streaming, client computers tend to disconnect from the networks suddenly or the browser applications can be reloaded by user operations. To avoid such kind of wasted resources, management servers send keep-alive messages include the session-ids sent from the clients at the initial resource reservations. When received a keep-alive message, the client verify the session-id contained in the keep-alive message. If the keep-alive message is not the one the client stores, the client ignore the keep-alive messages. If the server do not receive the keep-alive responses from the client for certain configured times, the server free the resource bound to the session-id. By default, the keep-alive interval is 300 seconds and the default keep-alive timeout count is 3.

6. Security Considerations

TBD

7. IANA Considerations

TBD

8. Acknowledgement

The author would like to thank Yasuo Okabe, Osamu Nakamura and Kaoru Maeda for their good contributions to this document.

9. References

9.1. Normative References

[RFC6455] Fette, I. and A. Melnikov, "The WebSocket Protocol", RFC 6455, December 2011.

9.2. Informative References

[MPEGDASH]
"ISO/IEC 23009-1:2014: Dynamic adaptive streaming over HTTP (DASH) -- Part 1: Media presentation description and segment formats", May 2014, <http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html>.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

Authors' Addresses

Yoshiharu Tsuzaki
Kyoto University

Email: tsuzakiyo@net.ist.i.kyoto-u.ac.jp

Ray Atarashi
IIJ Innovation Institute Inc.

Email: ray@iijlab.net

Shigeya Suzuki
Keio University

Email: shigeya@wide.ad.jp

Koshiro Mitsuya
Lepidum Co. Ltd.

Email: mitsuya@lepidum.co.jp

Kouji Okada
Lepidum Co. Ltd.

Email: okd@lepidum.co.jp