Route Demo Setup Documentation

1 Introduction

This document provides the description of the prototype ROUTE demo setup. An example usage guide to recreate the setup is provided in Section 3.

2 Existing System Architecture

The existing system architecture is shown in Figure 1, the left hand side is the sending end. DASH content is generated offline and the server-side ROUTE sender pushes it to the client side ROUTE receiver. This ROUTE receiver is derived from the open-source Mad FLUTE implementation.

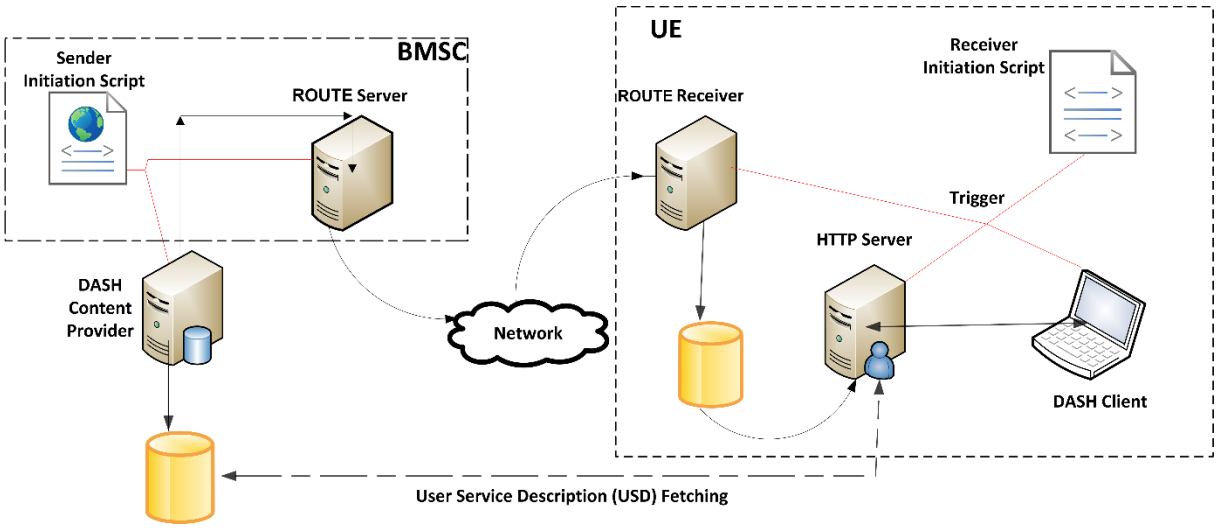

Figure 1 Simulation of end-to-end architecture

2.1 System Software and Hardware

Both the sending and receiving end are running on off-the-shelf hardware and open source OS as described in Table 1.

Table 1: Server/Client hardware specifications

Component	Description
CPU	Intel(R) Core(TM)2 DUO CPU E7400 @ 2.80 GHz
Memory	2 GB
Network Interface	Netlink BCM5784M Gigabit Ethernet PCIe with 1 Gbps capacity
Kernel Version	Linux 3.5.0-23-generic i386

Linux Distribution	Ubuntu 12.04.2 LTS

The client and server machine are time synchronized via NTP (or both sender and receiver can run on the same machine).

2.1.1 Sender Side Specification

Sender side consists of offline generated DASH content and the ROUTE sender as described in the following.

2.1.1.1 DASH Content Generation

Some important parameters of interest of DASH content that is being sent is provided here. The setup is multicasting two channels (each sent as an MPD describing a single video and audio representation). For the source content, we can use "Big Buck Bunny", “Elephant Dream” AVI video which has the properties presented in Table 1.3:

Table 1.3: Source video content

Type	**Parameter**	**Channel 1**	**Channel 2**
Content	**Name**	Elysium Clip	Tears of steel
	Source	Sony	CC
ROUTE	**Duration**	12 min 14 sec (1.5 minutes clip looped)	12 min 14 sec
DASH	**Segment Sizes**	500 msec, 1 seconds	
Video	**Codec**	AVC	
	Resolution	1920x1080p	
	Frame rate	24 fps	
	Bitrate	8 Mbps	
Audio	**Codec**	AAC LC	
	Channels	2	
	Bitrate	64kbps	

The content is then converted into a live-profile MPD with 1 second as well as 500 msec segment duration.

2.1.1.2 ROUTE Sending

Each audio and video stream is sent in a separate session as depicted in figure below. Each media segment is preceded by the initialization segment of the respective media. Audio initialization segments are preceded by MPD because of slightly shorter duration and hence higher frequency of audio segments than video.

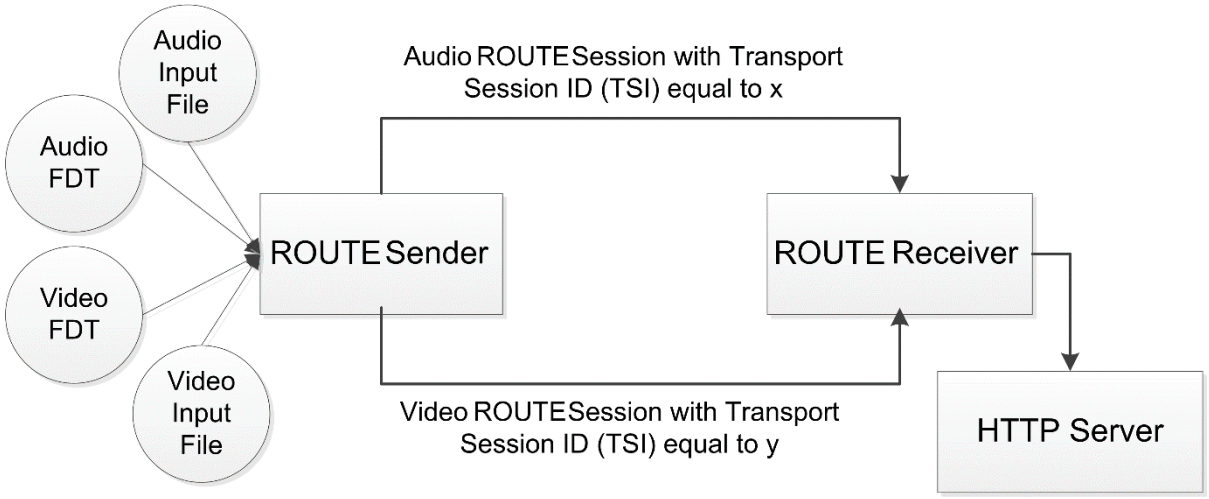

Figure 1.3: Adopted approach

SDP example file with description of parameters is provided below.

```
v=0
o=root 2890844526 2890842807 IN IP4 10.4.128.52
s=DASH
t=0 0
a=source-filter: incl IN IP4 * 10.4.128.52
a=flute-tsi:1
a=flute-ch:1
m=application 4000 FLUTE/UDP 0
c=IN IP4 224.1.1.1/4000
```

Figure 2. SDP example for video session

```
v=0
o=root 2890844526 2890842807 IN IP4 10.4.128.52
s=DASH
t=0 0
a=source-filter: incl IN IP4 * 10.4.128.52
a=flute-tsi:2
a=flute-ch:1
```

```
m=application 4000 FLUTE/UDP 0
c=IN IP4 224.1.1.1/4000
```

Figure 3. SDP example for audio session, the only difference from video session SDP is the TSI

The SDP files for the second channel are described similarly, using flute-tsi 3 and 4 respectively.

Table 1.6 explains the fields of the SDP.

Table 1.6: SDP file description

Field	SubField(s)		Description
v			Version. Only 0 is supported
6*o	<username>		User's Login
	<sess-id>		Session ID. NTP timestamp is recommended
	<sess-version>		Session version. NTP timestamp is recommended
	<nettype>		Network type. IN indicates internet
	<addrtype>		Address type
	<unicast-address>		Source address
s			Session name
2*t	<start-time>		Using zero for both indicates a permanent session
	<stop-time>		Other values would bound the session
7*a	5*<source-filter>	<filter-mode>	incl: Only accept packets from <src-list>
		<nettype>	IN for internet
		<address-types>	Use IP4 addresses
		<dest-address>	Destination address. * to accept all from c
		<src-list>	Source address
	<flute-tsi>		Transport Session ID (TSI)
	<flute-ch>		Number of channels in the session
3*c	<nettype>		IN for internet
	<addrtype>		Use IP4 addresses
	<connect-addr>		MulticastGroupAddress/TimeToLive

2.1.2 Client Side

As shown in Section 2, client side consists of the following main components:

1. A ROUTE receiver.
2. A local HTTP server to serve DASH content, e.g. Apache HTTP server [1].
3. A DASH player to playback the content, e.g. DASH JS player [2].
4. An application (UI/Script etc.) that can initialize items 1 through 3 above in a proper, timely fashion (provided in [4]).

Using the SDP file as specified above for the two sessions and a simple bash shell script, this setup

1. Receives the two ROUTE sessions for audio and video (with MPD and initialization data). This is done using sample SDPs provided in [4] and documented in Section 2.1.1.2,
2. Publish it to the client-side HTTP server,
3. Play it back it via the client, e.g. DASH JS player [2].

3 Usage Guide

This guide assumes the user is well familiar with Linux and shell scripts.

1.1 System Preparation

1. Prepare client and server machines to or exceeding the specifications in Table 1
2. Set up clock synchronization e.g. using ntpd.
3. Prepare build environment (typical tools as gcc, make etc.)
4. Install LAMP (sudo tasksel install lamp-server)
5. In the following, www_root represents the HTTP root directory (under Ubuntu /var/www/html or /var/www/). Apache user (typically "www-data") should be the owner of this directory.
6. In addition to the user running the software, add the user "www-data" also to sudoers (without password). This is necessary to run flute reception processes with elevated priority so that no packets are missed. This can be done by e.g. running "sudo visudo". Add the following line there: "www-data ALL=(ALL) NOPASSWD: ALL".
7. Install Google Chrome development channel.

1.2 Setting Up the software

1. Download the software archive (including the prepared content) [4]. Extract it under www_root. It will make a directory www_root/Work
2. www_root and its sub-folders should be owned by www-data (by changing ownership: sudo chown –R www-data www_root).
3. Give full read-write permissions to other users (sudo chmod –R 777 www_root).
4. To setup multicast on local host, start by disabling all network interfaces other than the local host. Then running the following:
    a. Configure loopback for multicast, increase all buffer sizes:
    sudo www_root/Work/Config/routeud.sh
    b. Generate SDP files for receiver. In absence of any enabled network interface except the loopback, set IP to 0.0.0.0 as shown below:
        - cd www_root/Work/Route_Receiver/bin/

- ../../Config/gensdp.sh 0.0.0.0

c. Generate SDP files at sender:
   - cd www_root/Work/Route_Sender/bin/
   - ../../Config/gensdp.sh 0.0.0.0

3.1.1 Updating

To update the receiver side, execute the following steps for patching:

1. Navigate to /var/www/html/Work/
2. Execute: sudo mv Route_Receiver/ Route_Receiver_bu
3. Place update archive [5] under cd /var/www/html/Work/
4. Execute: tar -xvf 2015_01_19_Route_Receiver_v2.tgz
5. Execute: sudo chown -R www-data Route_Receiver

1.3 Running the demo

There is a looping script for starting the sender under www_root/Work/Route_Sender/bin. To send 1 second segments, execute:

```
sudo ./LoopService 1000
```

To send 500 msec segments, execute:

```
sudo ./LoopService 500
```

Receiver is run in browser at http://localhost/Work/Route_Receiver/Receiver_bitmovin .

4 References

[1] http://httpd.apache.org/
[2] https://github.com/Dash-Industry-Forum/dash.js
[3] http://www.chromium.org/getting-involved/dev-channel
[4] https://www.dropbox.com/s/fjuiy4mxfa5vv8l/2015_01_14_Work.tgz?dl=1
[5] https://www.dropbox.com/s/1d5dyot84k0tin2/2015_01_19_Route_Receiver_v2.tgz?dl=1