Autonomous Control Middleware Research Section
ETRI Technical Report 13VC1310-TR-03

218 Gajeong-ro, Yuseong-gu
Daejeon, 305-700, South Korea
http://www.etri.re.kr/

Paper Draft Review: Cloud Gaming Service Platform Supporting the Multiple Devices based on the Real-time Streaming

Sung-Soo Kim
sungsoo@etri.re.kr

March 27, 2013

Abstract

The technical memo describes the comments on the paper draft for submitting to the ETRI Journal.

1 Original Draft

1.1 System Architecture: System Overview

Our game servers have various types how many clients connected to the server depending on the client device resolutions as shown in Table 1. So, we defined the game server types as a number of connected users. If G denotes a game server, G_t is a number of game server defined as $G_t = \{S_1, S_2, \ldots S_n\}$, G_u is a number of maximum connected users defined as $G_u = \{U_1, U_2, \ldots U_8\}$ and G_c is a number of current connected clients to the game server defined as $G_c = \{C_1, C_2, \ldots C_8\}$, then G is defined as:

$$G = \{G_t, G_u, G_c\} \tag{1}$$

In our game servers, several games are running on the game server at the same time, game sounds are mixing through the one sound card. So, we create the virtual sound device for capturing separately. According to game server types in step (1), we allocate the unique number using pre-defined index number into the virtual sound driver as ordering 1 to 8 left to right corresponding to the number of maximum connected user G_u. So, virtual sound driver S_d is defined as:

$$S_d = \{D_1, D_2, \ldots D_8\} \tag{2}$$

After creating the virtual sound device, we must set to the basic sound output device in windows system for most game sound played through this virtual sound device.

The rest of the paper, we describe focusing on the EQS with capture, audio/video encoding and streaming.

1.2 System Architecture: EQS

After the game server types in (1) and virtual sound driver in (2) are defined, EQS is ready to accept the client request for capture and encoding and streaming to the clients. As shown in Figure 2, new client requests join to the game server, DSP assigns the suitable game server according to the resolution, memory state, numbers of connected users and network state.

In other to sound capture, we need to assign a shared memory Queue corresponding to the G_d. So, shared memory queue G_q is defined as:

$$S_q = \{Q_1, Q_2, Q_8\} \tag{3}$$

Algorithm 1 Audio/Video Capture/Encoded and Streaming

```
1:  procedure EQS(G_i)
2:      AudioCapture SC_i;
3:      VideoCapture VC_i;
4:      PacketizationFrame p;
5:      EncodedAudio A_i;
6:      EncodedVideo V_i;
7:      YUVImage Y_i;
8:      while V_i ≠ NULL do
9:          SC_i ← convertRGB2YUV(G_i, Q_i);
10:         VC_i ← captureScreenRect(G_i, V_i);
11:         Y_i ← convertRGB2YUV(G_i, V_i);
12:         V_i ← encodeVideo(VC_i);
13:         A_i ← encodeAudio (SC_i);
14:         p ← PacketizationAudioVideo(A_i, V_i);
15:         G_c ← transmitAVstream(p);
16:     end while
17: end procedure
```

2 Comments

The below paragraph should be modified in detail because it is too much abstract. Moreover, to achieve *coherence*, then, you should show how all of the *ideas* contained in a paragraph are relevant to the *main topic*.

2.1 Original Paragraph

Figure 3 shows a RTP Packetizer and Sender. Flows are as follows:

- Create the UDP socket for the RTP sending using the client IP and Port no.
- Create the RTP Parser
- RTP pack for x264 video frame using the parameter such as rtp handler, timestamp, number of NAL and length
- RTP pack for AAC audio using the parameter such as rtp handler, timestamp, data length
- Deliver to the client using libcup udp sender write() callback function

2.2 System Architecture: System Overview

2.2.1 Review Questions and Comments

Question #1: *Why should you define a game server?*

If G denotes a game server, G_t is a number of game server defined as $G_t = \{S_1, S_2, \ldots S_n\}$, G_u is a number of maximum connected users defined as $G_u = \{U_1, U_2, \ldots U_8\}$ and G_c is a number of current connected clients to the game server defined as $G_c = \{C_1, C_2, \ldots C_8\}$, then G is defined as:

$$G = \{G_t, G_u, G_c\} \tag{4}$$

2.2.2 Modified Version

Our system supports five service modes for real-time streaming according to the resolutions of the client devices with certain maximum users as shown in Table 1. Here, we define the service mode, $\mathcal{SM}$, which is determined by the resolutions of client devices, r, and the maximum connected users, n. Thus, $\mathcal{SM}$ can be defined as the function, $\mathbf{f}$, which have r and n as the parameters for real-time game streaming services.

$$\mathcal{SM} = \mathbf{f}(r, n)$$

2.2.3 Modified Version

The sounds of the games is important role in providing the immersive gaming services along with visual elements. In order to support multi-users for each different games on single server at the same time, our system should support game streaming for each game respectively. However, conventional audio processing based on the dedicated hardware is difficult to directly exploit in multiple game streaming. The main problem of dedicated audio processing hardware in our configuration is blending of multiple sounds for the different games. Thus, we exploit the *virtualization* technique based on the virtual audio device for isolating the multiple sounds respectively.

First, we allocate the unique pre-defined index into the virtual audio device from 1 to 8 corresponding to the number of maximum connected users, n. After that, our system configures the basic sound output device in windows system for multiple game sounds played through the virtual audio devices. Finally, the system captures the sounds for several games respectively in order to encode the game streams as H.264 format.

The rest of the paper, we describe the major functions in the EQS such as capturing audio/video streams of the multiple games, encoding and streaming for the captured game streams.

2.3 System Architecture: EQS

2.3.1 Review Questions and Comments

Question #2: *What do you want to present? The below paragraph will review after the discussion is completed.*

After the game server types in (1) and virtual sound driver in (2) are defined, EQS is ready to accept the client request for capture and encoding and streaming to the clients. As shown in Figure 2, new client requests join to the game server, DSP assigns the suitable game server according to the resolution, memory state, numbers of connected users and network state.

In other to sound capture, we need to assign a shared memory Queue corresponding to the G_d. So, shared memory queue G_q is defined as:

$$S_q = \{Q_1, Q_2, Q_8\} \tag{5}$$

Algorithm 2 Audio/Video Capture/Encoded and Streaming

```
1: procedure EQS(G_i)
2:     AudioCapture SC_i;
3:     VideoCapture VC_i;
4:     PacketizationFrame p;
5:     EncodedAudio A_i;
6:     EncodedVideo V_i;
7:     YUVImage Y_i;
8:     while V_i ≠ NULL do
9:         SC_i ← convertRGB2YUV(G_i, Q_i);
10:        VC_i ← captureScreenRect(G_i, V_i);
11:        Y_i ← convertRGB2YUV(G_i, V_i);
12:        V_i ← encodeVideo(VC_i);
13:        A_i ← encodeAudio (SC_i);
14:        p ← PacketizationAudioVideo(A_i, V_i);
15:        G_c ← transmitAVstream(p);
16:    end while
17: end procedure
```