Real-time streaming for the animation of talking faces in multiuser environments

Joern Ostermann[1], Juergen Rurainsky[2], Reha Civanlar[1]

[1]AT&T Labs-Research
Middletown, NJ - USA
{osterman,civanlar}@research.att.com

[2]University of Dortmund
Dortmund – Germany
jru@kt.e-technik.uni-dortmund.de

ABSTRACT

In order to enable face animation on the Internet using high quality synthetic speech, the Text-to-Speech (TTS) servers need to be implemented on network-based servers and shared by many users. The output of a TTS server is used to animate talking heads as defined in MPEG-4. The TTS server creates two sets of data: audio data and Phonemes with optional Facial Animation Parameters (FAP) like smile. In order to animate talking heads on a client it is necessary to stream the output of the TTS server to the client. Real-time streaming protocols for audio data already exist. We developed a real-time transport protocol with error recovery capability to stream Phonemes and Facial animation Parameters (PFAP), which are used to animate the talking head. The stream was designed for interactive services and allows for low latency communications. The typical bit rate for enabling a talking face is less than 800 bit/s.

1. INTRODUCTION

2D and 3D talking head-and-shoulder models are developed to be used as human computer interfaces [1]. These faces are controlled by a Text-to-Speech (TTS) synthesizer [2]. They produce mouth and tongue movements synchronized with the speech in real-time on a PC or other computer platform. MPEG-4 includes this technology to enable animation of different face models within scenes in a MPEG-4 player [3].

The TTS decoder architecture as defined in MPEG-4 assumes a TTS server located at the client. This limits the choices of voices available to the MPEG-4 player to those locally available. Advanced TTS servers are using voice databases larger than 100 Mbytes of storage [5]. Therefore new voices cannot easily be installed on the MPEG-4 player.

However, there is no need for each client to have a TTS server running, if a TTS server in the network offers the appropriate selection of voices. This network based TTS server has to stream audio (synthetic speech) as well as TTS markup information for the animation of the talking face.

These streams can be used to animate an unlimited number of talking faces on different PCs that are joining a multicast session, or individual face models using a unicast session. In this paper, we propose a protocol that defines the transport of the TTS markup information from a network-based TTS server to an MPEG-4 client or talking head.

2. FACE ANIMATION IN MPEG-4

MPEG-4 specifies a face model in its neutral state, a number of feature points on this neutral face as reference points, and a set of FAPs (Facial Animation Parameters), each corresponding to a particular facial action deforming a face model in its neutral state [6]. Deforming a neutral face model according to some specified FAP values at each time instant generates a facial animation sequence. The FAP value for a particular FAP indicates the magnitude of the corresponding action, e.g. a big versus a small smile or deformation of a mouth corner.

Three groups of facial animation parameters are defined. First, for low-level facial animation (jaw, eyebrow, tongue...), a set of 66 FAPs is defined. Second, for high-level animation, a set of primary facial expressions such as joy, and sadness are defined. Third, for speech animation, fourteen visemes define mouth shapes that correspond to phonemes.

MPEG-4 allows for a synchronized presentation of several streams containing voice, music, text, video, and animated graphics. This requires the timing of all media to be known during content creation and at the time when a media server send the streams to a client.

However, for talking faces we do not know the timing of the synthesized speech before it is actually synthesized. Therefore, MPEG-4 developed a mechanism to synchronize synthesized speech with face animation by controlling the animation of the face using the speech synthesizer. A syntactic decoder in the M-TTS decoder (see Figure 1) decodes the text input from the server and sends the text and control information to the proprietary speech synthesizer. The TtsFAPInterface() is defined in [3], and has to be used to transmit phonemes and bookmarks from the speech synthesizer to the Pho-

neme/Bookmark to FAP converter. A phoneme/book-mark-to-FAP converter translates phonemes into FAPs that are used to animate the mouth of the talking face.

In order to enable the synchronization of facial expressions with the speech, bookmarks in the text are used as shown in Figure 2. A bookmark defines the FAP that is animated, its target amplitude, the time the face model has to reach the amplitude, and the curve that the amplitude has to follow in order to change from its current value to the value defined in the bookmark. Execution of a bookmarks starts at the beginning of the following word.

The Compositor (Figure 1) synchronously presents the synthesized speech as well as the animated face. It is also able to control the output of the TTS server with commands like start/stop/pause.
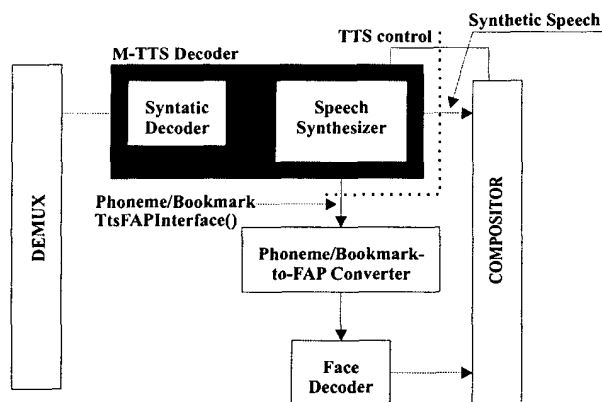

Figure 1: MPEG-4 Audio TTS decoder architecture.

<FAP 2 1 40 1 40 2000 3>text<FAP 2 1 63 1 63 600 3> ....text.........<FAP 2 1 0 1 0 1000 3>...text.....
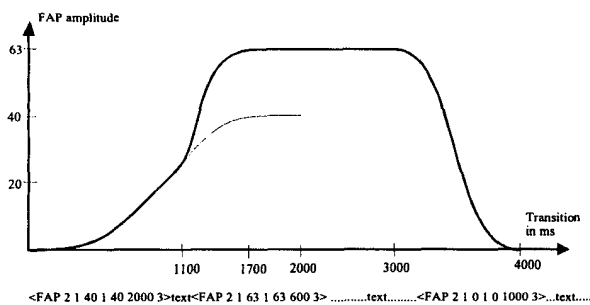
Figure 2: Example for text with bookmarks for one facial expression (joy) and the related amplitude of the animated FAP. The syntax of a bookmark is: <FAP 2 (expression) 1 (joy) amplitude 1 (joy) amplitude transition 3 (Hermite time curve)>.

3. DECODER ARCHITECTURE FOR A NETWORK BASED TTS SERVER

A separation of the TTS server and the talking face, and reallocating both on different positions in a network like

the Internet, involves transmissions of the outputs of the TTS server to the talking face(s). The possibility to control the outputs of the TTS server needs to be transmitted as well (see dotted line in Figure 1). The TTS decoder architecture for a network based TTS server is shown in Figure 3. At the TTS server and at the client interfaces have to be used to transmit and to receive the outputs of the TTS server. These interfaces can be used to adjust the data from the TTS server to the needs of the channel. In case of the Internet as channel between the TTS server- and client- interface a set of protocols is defined.
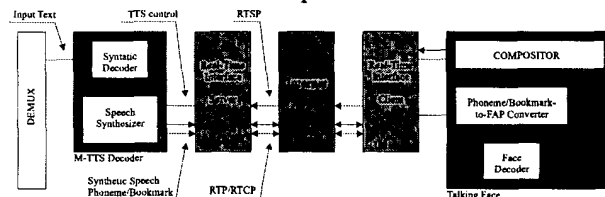

Figure 3: Network based Audio TTS decoder architecture; (RTP/RTCP) Real-time Transport protocol/Real-time Transport Control Protocol, (RTSP) Real-time Transport Streaming Protocol.

4. FACE ANIMATION USING PFAP STREAMS

The interfaces shown in Figure 3 are designed as real-time interfaces. In order to transport real-time data the Real-time Transport Protocol (RTP) can be used. This protocol is based on a RTP header and a payload, which changes the format with the type of data in this payload. Different payload format definitions for audio and video data are specified, but not one for phonemes and FAPs. The following sections describing the newly developed Phoneme/Facial Animation Parameters stream (PFAP) as RTP payload format.

The PFAP payload is designed to hold phonemes, and FAPs. In addition, recover information for FAPs are specified with this payload format. These three types of data are transported using descriptors to specify the structure of each element. Figure 4 shows the PFAP stream as a RTP payload format definition. Only the *Packet Descriptor* has a fixed position. All other descriptors are in the order of usage.

The *Packet Descriptor* has two functions: describe the type of recover information (dynamical or complete) and define the type of descriptor (phoneme or FAP) following the optional recover information.

The *Phoneme Descriptor* is designed to hold the phoneme symbol, phoneme duration, stress, and f0Average of this phoneme. The phoneme symbol is defined through a hexadecimal number, which leads together with a mapping table to the right phoneme out of a phoneme alphabet like DARPA or ARPAbet. The phoneme duration is given in units of milliseconds. Stress marks a stressed phoneme, and f0Average the frequency of

the synthesized audio signal for this phoneme in units of 2 Hz. The next descriptor (phoneme or FAP), end of packet or end of text is defined using two information bits.
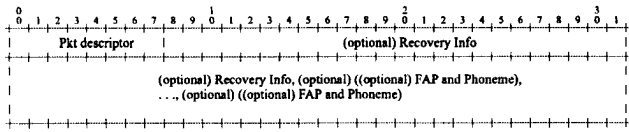
```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Pkt descriptor    |            (optional) Recovery Info            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          (optional) Recovery Info, (optional) ((optional) FAP and Phoneme),    |
|          ..., (optional) ((optional) FAP and Phoneme)                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: PFAP Payload; (Pkt descriptor) Packet Descriptor)

The *FAP Descriptor* defines a FAP, desired amplitude, transition, and a time curve. The valid range of FAPs transmitted via *FAP Descriptor* is defined from 3 to 74. Facial expressions start with FAP 69 (joy). A bookmark for a combined facial expression is separated into two *FAP Descriptors* (Example: bookmark sequence for a combined expression <FAP 2 1 40 2 30 2000 3>; transformed bookmark sequence <FAP 69 40 2000 3><FAP 70 30 2000 3>, see Figure 2). The amplitude of a particular FAP is measured in different units according to the part of the face and the desired movement. The maximum amplitude is 2529600 in *angle unit*, which is used for spins. The transition and time curve are defining the amplitude changes in time. MPEG-4 does not define a maximum transition, but this payload has a maximum transition of 16383 ms, in order to assign a fixed number of bits for this value.

The PFAP RTP payload offers to recover FAP(s) in the case packets get lost. Since the format is designed for real-time interactive services, we only recover state-like information, i.e. the facial expressions but not the transient phonemes. Dynamical recovery information may be transmitted with each regular. Complete recovery is transmitted as a separated packet. Dynamical recovery holds FAP(s) from previous *n* packets, and complete recovery recovers the state of the facial expressions and holds all FAP(s) with a non-zero amplitude.

5. SYNCHRONIZATION

The new PFAP stream and the synthetic speech are transmitted using the Real-time interfaces shown in Figure 3. These streams are payloads of two independent (Real-time Transport Protocol) RTP streams. Such RTP streams need to be synchronized at the client. Therefore the field *timestamp* of the RTP header, and the fields *NTP timestamp* and *RTP timestamp* of the additionally transmitted (Real-time Transport Control Protocol) RTCP packets have to be used. The field *timestamp* of a RTP packet, with the PFAP stream as payload, defines the playtime of the first phoneme described in this stream plus an initial offset (random number). The field *timestamp* of a RTP packet, with audio as payload, defines the playtime of the first byte of the payload plus an initial offset (random). A defined clock frequency is used to calculate the value of

these fields. The field *NTP timestamp* of the RTCP packets of the PFAP and audio stream are synchronized, so that they can be matched at the client. The field *RTP timestamp* of the RTCP packets of the PFAP and Audio stream holds the associated random offset plus the *NTP timestamp* with the same units like the *timestamp* of the RTP packet. The *NTP timestamp* will be not more mentioned in the future discussion, because of the assumption, that the *NTP timestamp* of the audio stream and the *NTP timestamp* of the PFAP stream are synchronized. Table 1 gives an overview of the values in the described fields of the RTP header and RTCP packets.

Table 1: Timestamps of the RTCP and RTP packets, (initial) random number, (d1) duration of the first sentence, (d2) duration of the second sentence.

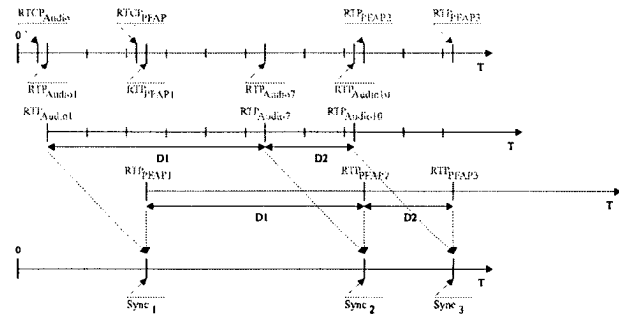| Packet	RTCP$_{Audio}$	RTP$_{Audio1}$	RTP$_{Audio7}$	RTP$_{Audio10}$
TimeStamp	initial$_{Audio}$	initial$_{Audio}$	initial$_{Audio}$+d1	initial$_{Audio}$+d1+d2
Packet	RTCP$_{PFAP}$	RTP$_{PFAP1}$	RTP$_{PFAP2}$	RTP$_{PFAP3}$
TimeStamp	Initial$_{PFAP}$	Initial$_{PFAP}$	Initial$_{PFAP}$+d1	Initial$_{PFAP}$+d1+d2

Figure 5: (first timeline) chronical order of received packets at the client, (second timeline) chronical order of the audio packets, (third timeline) chronical order of the PFAP stream packets, (fourth timeline) synchronization at the client, (d1) duration of the first sentence, (d2) duration of the second sentence

The synchronization of the RTP audio and RTP PFAP stream packets is shown in Figure 5, and achieved if *Sync* of Eq. 1 is ZERO with (TS) Timestamp, (X,Y) Packet numbers.

$$Audio_{TS} = RTP_{Audio_x} - RTCP_{Audio}$$

$$PFAP_{TS} = RTP_{PFAP_y} - RTCP_{PFAP}$$ (1)

$$Sync = Audio_{TS} - PFAP_{TS}$$

Holding one stream and shifting the other one can be used to determine the appropriate audio byte or phoneme, so that the equation for *Sync* becomes truth, in case of lost packets.

6. EXPERIMENTEL RESULTS

We use the *User Datagram Protocol* (UDP) in combination with RTP as transport protocols over the

Internet. UDP packets may get lost during transmission from the sender to the client(s). The range of network loss can be assumed between 0 % - 30 %. The kind of lost can vary from a single packet to a bursts of several consecutive packets.

A two channel model with two independent 2-state Gilbert models [4] was used to simulate network loss (see Figure 6). One channel was used to simulate network loss for regular packets with dynamical recovery information, and the other channel for network loss for complete recovery information. This model allows comparing both types of recovery information without changing the properties of a channel, which is usually used to transmit both types of packets.
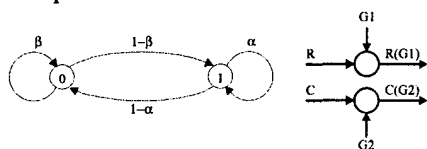

Figure 6: (left) 2-state Gilbert model, (α, β) probabilities to being in loss and no loss state; (right) 2 channel network loss model, (R) regular packets, (G1,G2) loss (Gilbert model) , (R(G1), C(G1)) regular packets and complete recover packets as functions of loss.

Test patterns were generated using the model shown in Figure 6. States 0 and 1 represent no loss and loss state, respectively. The probabilities of being in either state are expressed as α and β. The probability P_L of being in the loss state depends on the maximum number of packets B in a burst and is:

$$P_L = (1 - P_L)(1 - \beta) + \alpha P(1 \mid 10) + \alpha P(1 \mid 110) + \dots$$
$$+ \alpha P(1 \mid \underbrace{1\dots10}_{B-1}) \qquad (2)$$

with: $\alpha = 1 - 1/b, b \in [2,3,4]$, the average burst length b, and $P(1 \mid 10) = P(0)(1 - \beta)\alpha$.

We said a burst loss occurred when the model remains in state 1 for multiple consecutive packets. We limit bursts to 5 packets (parameter B of Eq. 2), which usually corresponds to 5 sentences. We ran the model simulation to generate loss pattern with loss rates $L_K \in [0\%,1\%,10\%,20\%30\%]$, for $b \in [2,3,4]$.

We created test sequences from several paragraphs of text with a high number of FAPs. We mark a frame as erroneous, if at least one FAP has a amplitude different to the no loss example. We measure the distortion d as the number of erroneous frames divided by the total number of frames. Figure 7 shows the distortion for a network loss of 10%. The line prevPackets shows the bit rate for the PFAP stream if we send dynamic recover information for the previous 1/2/4/7 packets. The curve compPackets shows the bit rate for complete recovery where a packet with complete recovery is sent after sending 1/2/3 regular packets. It can be seen that for text with many FAPs, the

use of complete recovery is more efficient than dynamic recovery. However, for text with a small amount of FAPs, the reverse is obviously true. Curves with network loss rates of 1%, 20%, and 30 % show similar behavior.
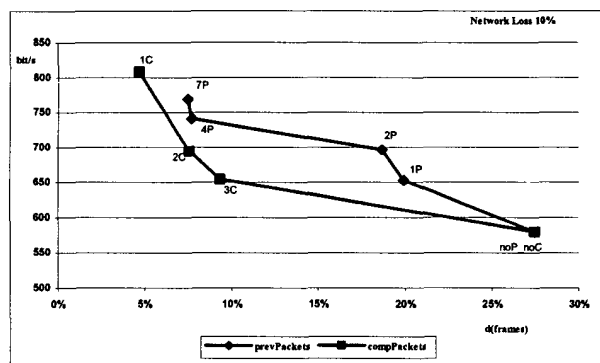

Figure 7: (1C,2C,3C) every {1st,2nd,3rd} regular packet complete recovery information was sent; (1P,2P,4P,7P) {1,2,4,7} previous packets were recovered with dynamical recovery information; (noP_noC).

7. CONCLUSIONS

The PFAP payload format enables to use face animation with network-based TTS. The error resilient transport of phonemes and facial expressions requires a data rate of less than 800 bit/s for text with many facial expressions. The latency of face animation using a network-based TTS server depends on the speed of the TTS server as well as the payload format. While one PFAP packet may hold the phonemes and facial expressions of an entire sentence, the user may choose to pack a sentence into several shorter packets thus reducing latency.

8. REFERENCES

[1] Rist, T., Andre, E., & Muller, J. Adding Animated Presentation Agents to the Interface. *Intelligent User Interfaces '97*, (Orlando, Florida), pp.79-86.

[2] Ostermann, J., Beutnagel, M., Fischer, A., Wang, Y. Integration of talking heads and text-to-speech synthesizers for visual TTS. ICSLP 99, Australia, December 99.

[3] ISO/IEC International Standard 14496-3; "Generic coding of audio-visual objects - Subpart 6: Text-to-Speech Interface", 1998.

[4] Sanneck, H., "End-to-End Packet Delay and Loss Behavior in the Internet", in SIGCOMM '93, pp.289-298, Sept. 1993.

[5] Beutnagel, M., Conkie, A., Schroeter, J., Stylianou, Y., & Syrdal, A. (16 March, 1999) The AT&T Next-Generation Text-to-Speech System, Joint Meeting of ASA/EAA/DAGA in Berlin, Germany.

[6] Tekalp, A. M., Ostermann, J., "Face and 2-D mesh animation in MPEG-4," Signal Processing: Image Communication, vol. 15 (4-5), 2000, pp. 387-421.