REIHE INFORMATIK
14/2001

RTP/I Payload Type Definition for Feedback Tools

Jürgen Vogel
Universität Mannheim
Praktische Informatik IV
L15, 16
D-68131 Mannheim

RTP/I Payload Type Definition for Feedback Tools

Jürgen Vogel
Department for Praktische Informatik IV
University of Mannheim
vogel@informatik.uni-mannheim.de

Abstract

This document specifies an application-level protocol (i.e., payload type) for feedback tools using the Real Time Protocol for Distributed Interactive Media (RTP/I). RTP/I defines a standardized framing for the transmission of application data and provides protocol mechanisms that are universally needed for the class of distributed interactive media. A feedback tool is used in synchronous collaborative environments for permanent feedback about certain criteria (e.g., audio quality). This document specifies how to employ a feedback tool with RTP/I and defines application data units (ADUs) for feedback tool operations. This protocol definition allows standardized communication between different feedback tool implementations.

1 Introduction

Distributed interactive media are media which allow a set of spatially separated users to synchronously interact with the medium itself. Typical examples of distributed interactive media are shared whiteboards, which are used to present and edit slides in a teleconferencing environment [Gey99], as well as distributed virtual environments (DVEs) [Hag96], shared text editors [HC97], and computer games with network support [GD98]. Feedback tools also belong to the class of distributed interactive media. They are used to poll opinions about certain criteria (e.g., technical parameters such as audio and video quality) of the participants in a collaborative session. Polling results are then displayed to all participants in an appropriate way (e.g., as histogram). In the following, we briefly introduce the class of distributed interactive media and the Real Time Protocol for Distributed Interactive Media (RTP/I). For a more detailed discussion please refer to [Mau00, MHKE01, MHK$^+$00].

In order to provide high responsiveness and to avoid the drawbacks of centralized approaches, such as the presence of a single point-of-failure and lack of scalability, applications for distributed interactive media often employ a replicated distribution architecture. In this architecture each user runs an instance of the application which manages a local copy of the medium's shared state. For example, the state of a feedback tool comprises all active criteria, including the individual votes of all participants.

The state of a distributed medium changes either by the passage of time or by means of user interactions (*events*). State changes due to the passage of time can be calculated locally and need not be distributed among application instances. In contrast, events have to be exchanged among instances via the network to all remote instances of the application so that each can modify its local copy of the state accordingly. For better handling, the application's state can be partitioned into several *sub-components*. Each feedback criterion is represented by a RTP/I sub-component.

RTP/I is an application-level protocol that employs the media model described above, and is applicable to arbitrary distributed interactive media. It consists of two main parts; both reside on the application level and are independent of the underlying network and transport layers:

- the framing protocol (RTP/I). RTP/I is used to frame the data transmitted by distributed interactive media. The RTP/I framing contains the information that is common to media of a specific class. This information makes it possible to understand to a large extent the semantics of the transmitted data without any medium-specific knowledge. Therefore, meaningful functionality and services that are independent of the media-specific data encapsulated by the framing information can be developed.

- the RTP/I control protocol (RTCP/I). RTCP/I is used to convey meta information about the medium and information about the participants in a session.

RTP/I is not a complete protocol. It needs to be adapted to the requirements of a specific medium or a group of media by defining either a payload or a profile. A profile adapts RTP/I to the needs of a group of distributed interactive media. A payload type definition is a specification document that defines how a particular medium is transported using the framework provided by RTP/I. Essentially, it describes how the medium-specific data are encoded and specifies a payload for feedback tools. The aim of such a standardized protocol is to allow communication between different feedback tool implementations. The specific encoding ”feedback tool” as defined by this document is assigned the payload type ”5”. Each RTP/I ADU carries this payload type as the identifier of the originating application.

The remainder of this document is structured as follows. First, we explain how RTP/I can be used for feedback. Then we define all the necessary application data units (ADUs) of a feedback tool protocol. These ADUs are transported either as an RTP/I state or event.

2 Usage of RTP/I

The state of a feedback tool consists of all criteria that can be voted upon at a certain point in time. Besides the description of a criterion (e.g., ”audio quality”) and a list of possible votes (e.g., ”no audio”, ”poor”, ”acceptable”, and ”good”), the medium’s state also includes a list of the votes given for each criterion.

Each feedback criterion is represented by a RTP/I sub-component. Each sub-component is assigned a unique ID that is allocated by a unique number service [MHK$^+$00]. The creation of a new criterion is signaled by a state transmission, whereas the payload of the RTP/I state encodes the criterion’s description and its list of possible votes. State changes (i.e., voting for a certain criterion or withdrawing a vote) are encoded as RTP/I events.

Each participant is assigned a unique identifier (participant identifier, pid). The pid can be used to map participants to their votes. Management of identifiers is outside the scope of this document, please refer to [MHK$^+$00].

In some cases it might be necessary to transmit the current state of a sub-component (e.g, resynchronization in case of an inconsistency). Such a state carries additionally a list of all given votes.

All the sub-components needed to display the medium’s state at a certain point in time are marked as active [MHK$^+$00]. It is therefore possible to limit the number of active criteria for feedback tools to a subset of all criteria defined. Changing the activeness of a criterion triggers the transmission of an event.

This document does not specify how the results of a voting process should be visualized by a feedback tool. Results could be displayed, for example, aggregated for all participants as a histogram. Alternatively, each vote could be stated explicitly in a text list.

This document does not specify a certain transport protocol. Rather, it is assumed that the application makes use of a reliable transport mechanism that guarantees the reliable distribution of operations. This mechanism can be integrated either into the application, or into RTP/I, or can be implemented at the transport level.

3 Feedback Tool ADUs

3.1 Question State ADU

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=0|    res  |v| nbr of answers|              ttl              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     qlength   |                 question                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                        question (continued)                   :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               question (continued)            |    padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   alength (0) |            answer expl (0)                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                     answer expl (continued)                   :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             answer expl (continued)           |    padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                                                               :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| alength (n-1) |               answer expl (n-1)               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                     answer expl (continued)                   :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           answer expl (continued)             |    padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    nbr of received answers    |               res             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   answer (1)  |      res      |              tl (1)           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           pid (1)                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                                                               :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   answer (n)  |      res      |              tl (n)           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           pid (n)                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

version (V) : 2 bits

This field specifies the version number of this protocol. The version defined by this specification is version 0.

reserved (res) : 6 bits

These bits are reserved for future use.

visible (v) : 1 bit

This field defines the visibility of the question, whereby 1 means that the question is currently visible and 0 that the question is currently invisible.

number of answers (nbr of answers) : 8 bits

This field gives the number of values that are possible as an answer to the question defined by this ADU. Answers are integers between 1 and number. The meaning of these values is defined by the following list of possible answers. The index of a list item (starting with 1) corresponds to the numerical value of the answer.

```
time to live (ttl) : 16 bits
     The ttl gives the time in seconds until a given answer expires and
     is deleted from the list of valid answers. A value of 0 denotes that
     an answer never does expire.

question length (qlength) : 8 bits
     This field identifies the length of the question text in bytes
     excluding any padding bytes.

question : qlength bytes + padding
     This field contains the question as ascii text. If the question
     length is not a multiple of 4 bytes, a minimum number of padding octets
     necessary to pad the list element to a full 32 bit boundary follow the
     question text.

answer explanation length (alength) : 8 bits
     This field identifies the length of the answer’s explanation in bytes
     excluding any padding bytes.

answer explanation (answer expl) : alength bytes + padding
     This field contains the explanation of the answer as ascii text. If the answer
     length is not a multiple of 4 bytes, a minimum number of padding octets
     necessary to pad the list element to a full 32 bit boundary follow the
     answer explanation.

number of received answers (nbr of received answers) : 16 bits
     This field gives the number of answers that have been received so far
     and are included as a list in this state ADU.

answer value (answer) : 8 bits
     The state ADU ends with a list of valid answers, in which each answer
     consists of a triple (answer value, time left, participant id).
     The answer value contains the numerical value chosen by the corresponding
     participant.

time left (tl) : 16 bits
     This field defines how many seconds are left before the answer
     becomes invalid. If the question type does not support expiring
     answers, it is set to 0.

participant identifier (pid) : 32 bits
     This field identifies the participant that has given the answer.
```

3.2 Event ADUs

3.2.1 Change Question Visibility ADU

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=0|   type    |    visible    |             res               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

type : 6 bits
     This field identifies the type of event. It is set to 0 (change
     question visibility.

visible : 8 bits
     Indicates whether the question is currently visible (1) or invisible (0).

reserved (res) : 16 bits
     These bits are reserved for future use.
```

3.2.2 Delete Question ADU

```
     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
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |V=0|   type    |                      res                      |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

    type : 6 bits
         This field identifies the type of event. It is set to 1 (delete
         question).

    reserved (res) : 24 bits
         These bits are reserved for future use.
```

3.2.3 Create Answer ADU

```
     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
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |V=0|   type    |    answer     |             res               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

    type : 6 bits
         This field identifies the type of event. It is set to 2 (create
         answer).

    answer value (answer) : 8 bits
         This field contains the answer value chosen by the participant.

    reserved (res) : 16 bits
         These bits are reserved for future use.
```

3.2.4 Delete Answer ADU

```
     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
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |V=0|   type    |                    res                        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

    type : 6 bits
         This field identifies the type of event. It is set to 3 (delete
         answer).

    reserved (res) : 24 bits
         These bits are reserved for future use.
```

References

[GD98] L. Gautier and C. Diot. Design and Evaluation of MiMaze, a Multi-player Game on the Internet. In *IEEE International Conference on Multimedia Computing and Systems*, pages 233–236, 1998.

[Gey99] W. Geyer. *Das digital lecture board – Konzeption, Design und Entwicklung eines Whiteboards für synchrones Teleteaching*. PhD thesis, University of Mannheim, Germany, 1999.

[Hag96] O. Hagesand. Interactive multiuser VEs in the DIVE system. *IEEE Multimedia*, 3(1):30–39, 1996.

[HC97] M. Handley and J. Crowcroft. Network Text Editor (NTE) – A scalable shared text editor for the MBone. In *ACM SIGCOMM*, pages 197–208, 1997.

[Mau00] M. Mauve. *Distributed Interactive Media*. PhD thesis, University of Mannheim, Germany, 2000.

[MHK$^+$00] M. Mauve, V. Hilt, C. Kuhmünch, J. Vogel, W. Geyer, and W. Effelsberg. RTP/I: An Application-Level Real-Time Protocol for Distributed Interactive Media. Internet Draft: draft-mauve-rtpi-00.txt, 2000.

[MHKE01] M. Mauve, V. Hilt, C. Kuhmñch, and W. Effelsberg. RTP/I - Toward a Common Application-Level Protocol for Distributed Interactive Media. *IEEE Transactions on Multimedia*, 3(1), 2001.