REIHE INFORMATIK
11/2001

RTP/I Payload Type Definition for Chat Tools

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

RTP/I Payload Type Definition for Chat 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 chat 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 chat tool provides an instant messaging service among an arbitrary number of users. This document specifies how to employ a chat tool with RTP/I and defines application data units (ADUs) for chat operations. This protocol definition allows standardized collaboration between different chat 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]. Chat tools also belong to the class of distributed interactive media. They allow a set of users to exchange text messages. 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 chat tool is a history of messages exchanged. The size of this history is dependend on different factors such as available ressources and user preferences.

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*. Since the state of a chat tool is small, it need not be partitioned.

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 chat tools. The aim of such a standardized protocol is to allow communication between different chat implementations. The specific encoding "chat tool" as defined by this document is assigned the payload type "3". 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 chat tools. Then we define all the necessary application data units (ADUs) of a chat protocol. These ADUs are transported either as an RTP/I state or event.

2 Usage of RTP/I

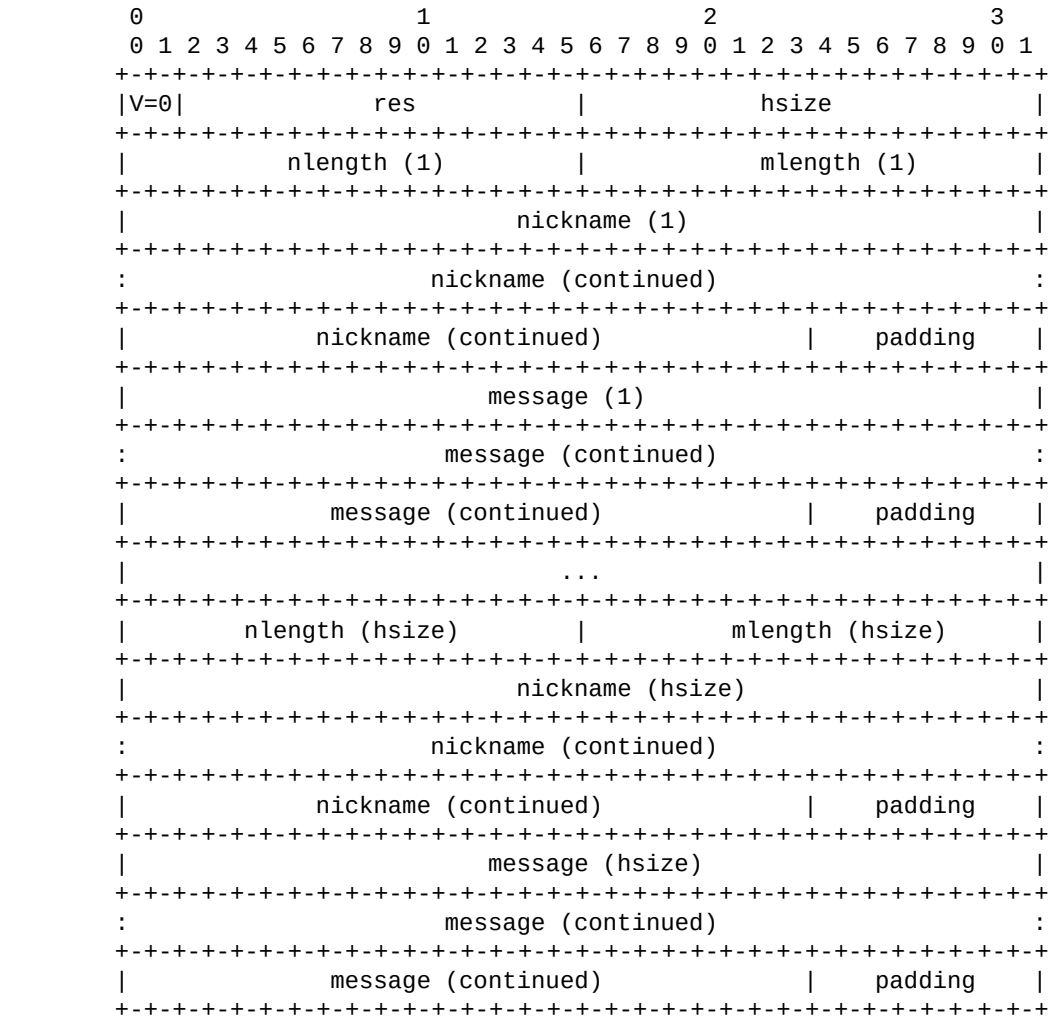
Each text message exchanged via the chat tool is assigned the nickname of the user that sent the message. This is reasonable since users can change their nickname at any time. The state of a chat tool consists of a history of messages having been exchanged by the users, including the corresponding nicknames. Since the state size is limited, the history should contain only a reasonable amount of messages. A state change is caused by adding (i.e. typing and sending) a new text message.

The state of a chat tool is encoded in a single RTP/I sub-component. Since each sub-component carries a unique identifier [MHK+00], the sub-component of a chat tool is assigned the identifier ”0”. Additionally, this sub-component must be marked at all times as ”active”. [MHK+00].

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 Chat Tool ADUs

3.1 State ADU



version (V) : 2 bits
This field specifies the version number of this protocol. The version defined by this specification is version 0.

reserved (res) : 14 bits
     This field is reserved for future use.

history size (hsize) : 16 bits
     This field identifies the number of messages encoded (as history of the chat) in this state ADU.

nickname length (nlength) : 16 bits
     This field identifies the length of the nickname of the message sender in bytes excluding any padding bytes.

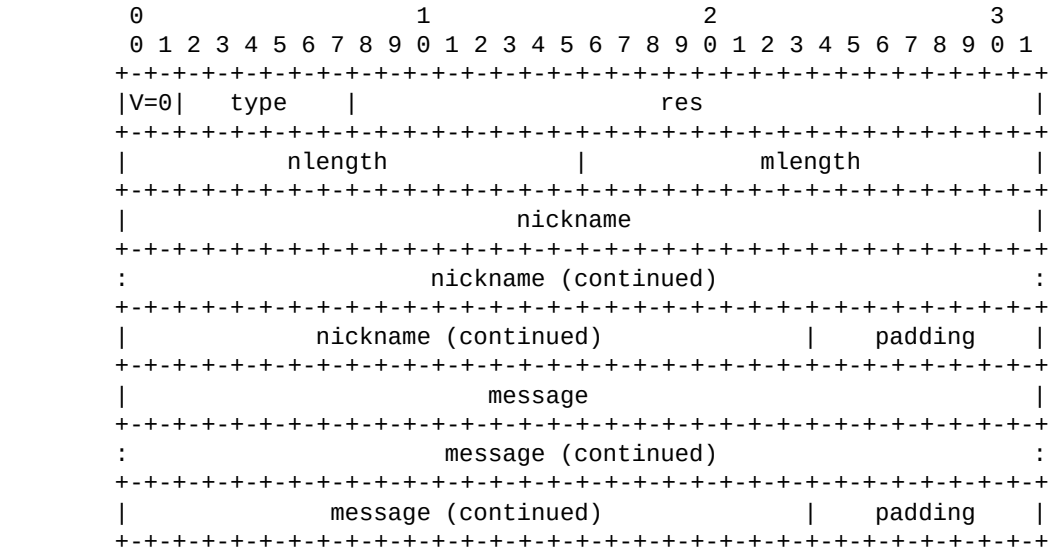
message length (mlength) : 16 bits
     This field identifies the length of the message in bytes excluding any padding bytes.

nickname : nlength bytes + padding
     This field contains the nickname of the message sender. If the name length is not a multiple of 4 bytes, a minimum number of padding octets necessary to pad the ADU to a full 32 bit boundary follow the nickname.

message : mlength bytes + padding
     This field holds the message. If the message length is not a multiple of 4 bytes, a minimum number of padding octets necessary to pad the ADU to a full 32 bit boundary follow the message.

3.2 Event ADUs

3.2.1 Add Message ADU



version (V) : 2 bits
     This field specifies the version number of this protocol. The version defined by this specification is version 0.

type : 6 bits
     This field identifies the type of event. It is set to 0 (add message).

nickname length (nlength) : 16 bits
     This field identifies the length of the nickname of the message sender in bytes excluding any padding bytes.

message length (mlength) : 16 bits
     This field identifies the length of the message in bytes excluding any padding bytes.

`nickname : nlength bytes + padding`
`This field holds the nickname of the message sender. If the name length is not a multiple of 4 bytes, a minimum number of padding octets necessary to pad the ADU to a full 32 bit boundary follow the nickname.`

`message : mlength bytes + padding`
`This field holds the message. If the message length is not a multiple of 4 bytes, a minimum number of padding octets necessary to pad the ADU to a full 32 bit boundary follow the message.`

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. Kuhmn̈ch, and W. Effelsberg. RTP/I - Toward a Common Application-Level Protocol for Distributed Interactive Media. *IEEE Transactions on Multimedia*, 3(1), 2001.