Network Working Group   O. Boyaci
Internet-Draft   H. Schulzrinne
Intended status: Standards Track   Columbia U.
Expires: April 17, 2009   October 14, 2008

              RTP payload format for Application and Desktop Sharing
                       draft-boyaci-avt-app-sharing-00

Status of this Memo

Abstract

   This document defines a protocol to support accessing general
   graphical user interface (GUI) desktops and applications remotely,
   either by a single remote user or embedded into a multiparty
   conference. The protocol is designed to allow sharing of, and access
   to general windowing system applications that are not expressly
   written to be accessed remotely.

Table of Contents

1. Definitions

    Application Host (AH): An application host (AH) is the computer
       which runs the shared application, distributes the screen updates
       to the participants, and regenerates human interface events
       received from participants.

    Participant: Participant is the computer which receives screen
       updates from AH and sends human interface events back to the AH.
       Participants do not need to store or run the shared application.
       More than one participant may connect to a single AH.

    Remoting Protocol: Remoting protocol messages allows AH to
       distribute windowing information and screen updates to
       participants.

    Human Interface Protocol (HIP): Human Interface Protocol (HIP)
       allows participants to send human interface device (HID) events to
       AH. HIDs generates mouse or keyboard events such as a key press,
       key release, mouse move, and mouse click.

2. Introduction

   Application and desktop sharing allows sharing of any software
   application with one or more participants over the Internet. The AH
   is the computer which runs the shared application. The participants
   receive the screen-view of the shared application from the
   application host (AH). They do not need to run or store the shared
   application. Their mouse and keyboard events are delivered and
   regenerated at the AH. An Application and desktop sharing system
   follows a client-server architecture [Figure 1].

```
                              +-------------+
                              |             |
                              | participant |
                              |             |
                              +-------------+
                                  ^    |
                                  |    |
                Screen updates    |    | HIP
                                  |    |
                                  |    |
                                  |    v
     +-------------+  Screen updates  +---------+
     |             |<----------------|          |
     | participant |                 |   AH     |
     |             |---------------->|          |
     +-------------+       HIP       +---------+
```
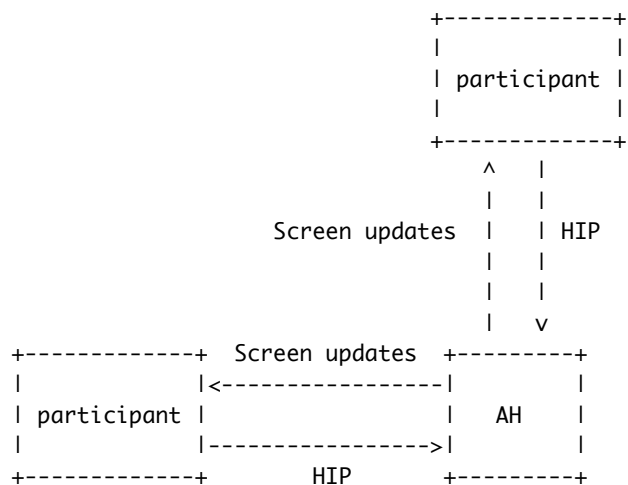
            Figure 1: Application sharing system architecture

   Application and desktop sharing enables collaborative work, software
   tutoring, and e-learning over the Internet. While two-party and
   multi-party conferencing using standards-based protocols is now
   common and well-developed, protocols for sharing applications are
   largely proprietary or based on the aging T.120 [T120] suite of
   protocols. In this document, we define an RTP payload format for
   application and desktop sharing.

   Application sharing differs from desktop sharing. In desktop
   sharing, a computer distributes all screen updates. In application
   sharing, the AH distributes screen updates if and only if they belong
   to the shared application's windows. Applications often consist of a
   changing set of related windows which serve the same task and are
   usually associated with the same process on the AH. Considering only
   the boundary of the shared windows is not enough. Other non-shared
   windows may cover the shared window or shared application may open
   new child windows such as those for selecting options or fonts. A

true application sharing system must blank all the nonshared windows
and must transfer all the child windows of the shared application.

We note that remote access to an application ("remote desktop") and
multiple users sharing an application within a collaboration setting
such as a multimedia call or multiparty conference are very similar.
The protocols defined in this document therefore support both.

Remote access differs from video transmission of the sort for which
most video encodings have been designed. In particular, screen
encoding may need to be lossless and typically operates on artificial
rather than natural (photographic) video input. The video input is
characterized by large areas of the screen that remain unchanged for
long periods of time, while others change rapidly. (However,
rendering the output of a modern computer-generated animation
application such as video games blurs the distinction between
traditional motion video output and screen sharing.)

Unlike earlier systems, such as T.120, we believe that application
sharing should be integrated into the existing IETF session model,
encompassing session descriptions using the Session Description
Protocol (SDP) [RFC2327] or successors and the Session Initiation
Protocol (SIP) [RFC3261]. Application sharing needs many of the same
control functions as other multimedia sessions, such as address
binding, session feature and media negotiation. We believe that use
of the session model is also beneficial for the remote desktop case,
as it allows to re-use many of the well-developed session components
and easily supports hybrid multimedia models, such as the delivery of
desktop audio to the remote user.

Remote access to graphical applications and desktops, as defined in
this document, has two important characteristics. First, the access
protocol is unaware of any semantic characteristics of the
applications being shared; it only transmits the visual
characteristics of the windows. This is different, therefore, from
shared-drawing or shared-editing tools that allow distributed
modification of documents. Secondly, the protocol is designed to
work with applications which were not written to be used remotely, by
intercepting or simulating their connections to their native window
systems. That distinguishes it from systems such as the X Window
System [X] which allow natively-written applications to be displayed
on remote viewers.

We distinguish between the AH user and participants. AH user employs
normal operating system mechanisms to interact with the running
application. Participants interact via the delivery protocols
described here.

The application sharing problem can be divided into four components: (1) setting up a session to the AH, (2) transporting human interface events from the participants to the AH, (3) delivering screen output from the AH to the participants, (4) moderating access to shared human interface devices such as pointing devices (e.g., mouse, joystick, trackball) and text input (keyboard). We refer to component (2) as the "Human interface protocol (HIP)" and component (3) as the "Remoting protocol". Remote user input access is moderated by the Binary Floor Control Protocol (BFCP) [RFC4582].

Session negotiation and description can be provided by existing session setup protocols. Thus, they are beyond the scope of this document, although it is important for an acceptable overall user experience.

The rest of this document is laid out as follows. Section 3 defines the common terminology for normative references. Section 4 gives an overview of the protocol's architecture and components. The remoting protocol and HIP are described in Section 5 and 6, respectively. Section 8 gives implementation notes, and Section 9 discusses security considerations.

3. Requirements Notation

   The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
   "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
   document are to be interpreted as described in [RFC2119].

4. Overview

4.1. Coordinate System

   The origin (0,0) of the coordinate system is the upper left corner.
   All coordinates carried in the protocol messages are absolute.

```
0,0                      <- x axis ->
  +------------------------------------------------+
  |                                                |
  |    220,150                                     |
  |       +----------------+                       |
  |       |                |      850,320          |
  |       |                |         +---------+   |
  |       |    450,400     |         |         |   |
  |       |       +----------------+ |         |   |
  |       |       |                | +---------+   |
  |       |       |                |               |
  |       +--------|               |               |
  |               |                |               |
  |               +----------------+               |
  |                       800,700                  |
  |                                                |
  |                                                |
  +------------------------------------------------+
                    AH                  1280,1024

  0,0
  +------------------------------------------------+
  |                                                |
  |    220,150                                     |
  |       +----------------+                       |
  |       |                |      850,320          |
  |       |                |         +---------+|
  |       |    450,400     |         |         ||
  |       |       +----------------+ |         ||
  |       |       |                | +---------+|
  |       |       |                |            |
  |       +--------|               |            |
  |               |                |            |
  |               +----------------+            |
  |                       800,700               |
  +------------------------------------------------+
                 Participant-1           1024,768
```

```
0,0
   +----------------+--------------------------------+
   |                |         630,170                |
   |                |            +---------+         |
   |      230,250   |            |         |         |
   |          +----------------+ |         |         |
   |          |                | +---------+         |
   |          |                |                     |
   +---------||                |                     |
   |          |                |                     |
   |          +----------------+                     |
   |                    580,550                      |
   |                                                 |
   |                                                 |
   |                                                 |
   |                                                 |
   +-------------------------------------------------+
                Participant-2            1280,1024

0,0
   +----------------+----------+
   |                |          |
   |  60,180        +---------+ |
   |     +----------------+ |   |
   |     |                | |   |
   |     |                | |-+ |
   |     |                | |   |
   +----||                |    |
   +----+----------------+-----+
          Participant-3    640,480
```
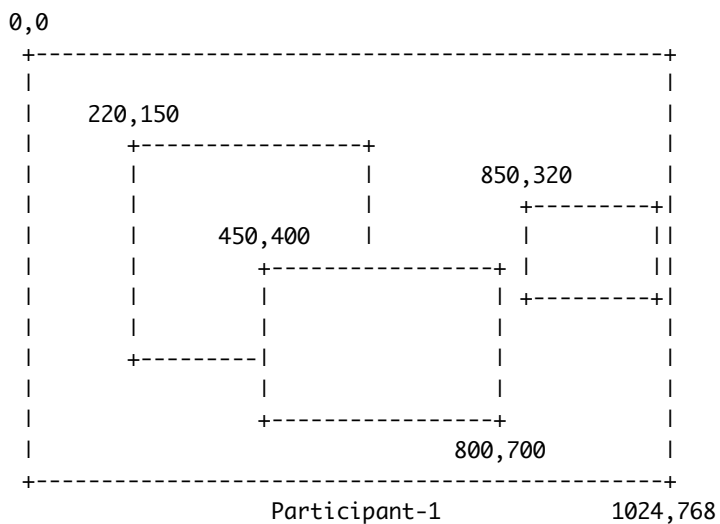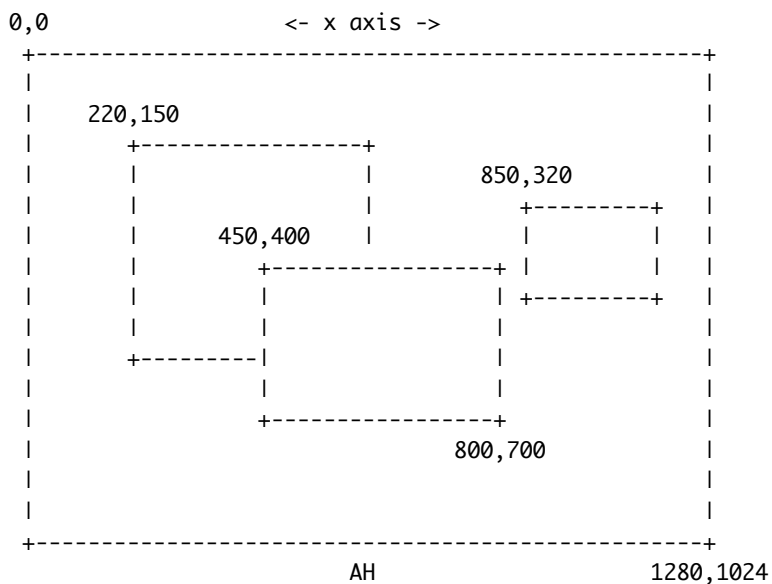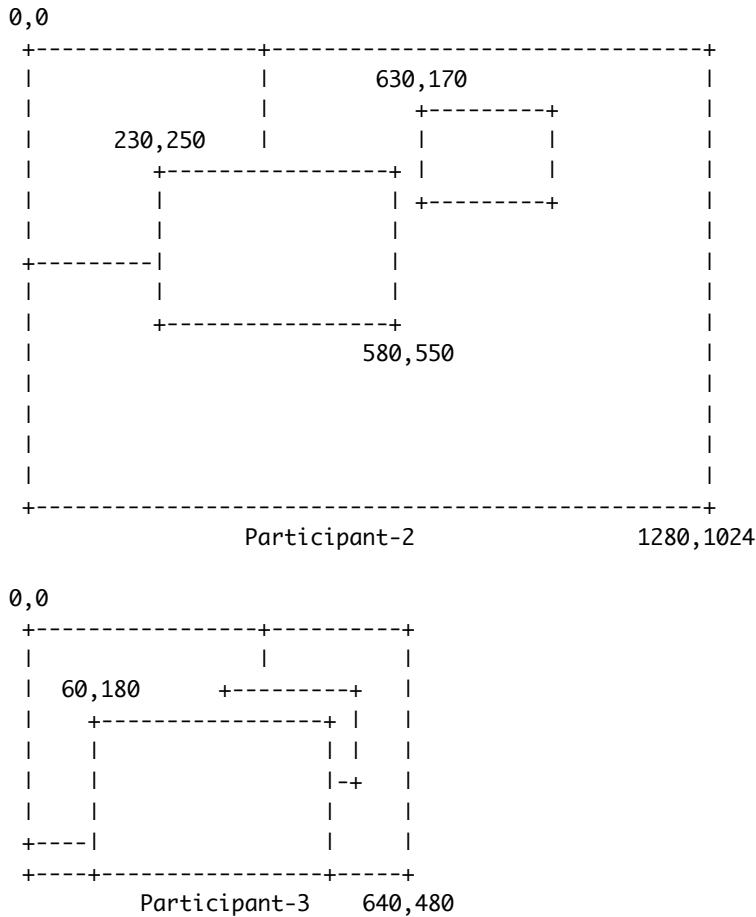
         Figure 2: An example to demonstrate the coordinate system

   Figure 2 shows an example scenario where three windows are shared.
   All coordinates are absolute in Figure 2. A participant can display
   the windows in their absolute coordinates or it can display them in
   different coordinates. In Figure 2, participant-1 displays the
   windows in their original coordinates. Participant-2 shifts all the
   windows 220 pixels left and 150 pixels top. Participant-2 preserves
   the relations between windows, while participant-3 combines all the
   windows in order to fit them to it's small screen. In this example
   scenerio, all participants preserve the z-order of windows. A
   participant MAY allow changing the z-order of windows locally,
   without changing the z-order in the AH.

   The AH MUST only accept legitimate HIP events by checking whether the
   requested coordinates are inside the shared windows.

4.2. Architecture

   Detecting a change in the GUI of the shared application, the AH
   prepares an RTP [RFC3550] packet containing an encoded image of the
   updated region. RTP allows the participants to re-order the packets,
   recognize missing packets and synchronize application sharing with
   other media types like audio and video. The screen updates can be
   encoded with PNG [I-D.boyaci-avt-png], JPEG [RFC2435], Theora
   [theora] or other media types like H.264 [RFC3984], according to
   their characteristics. PNG is an open image format which uses a
   lossless compression algorithm and more suitable for computer
   generated images. JPEG is lossy, but more suitable for photographic
   images. Theora is an open source video codec comparable to H.264 and
   suitable for movie encoding.

   Although multiple users could receive the screen updates
   simultaneously, clearly only one of them can manipulate the
   application via keyboard and mouse events. The Binary Floor Control
   Protocol (BFCP) [RFC4582] MAY be used to restrict the control of the
   application to a single user. BFCP receives floor request and floor
   release messages from participants; and then it grants the floor to
   the appropriate participant for a period of time while keeping the
   requests from other participants in a FIFO queue. The details of
   utilizing BFCP in the context of application and desktop sharing is
   given in Appendix A.

   The protocol supports two different mouse pointer models. Mouse
   pointer images can be transmitted as RegionUpdate messages or they
   may be transmitted seperately as MousePointerInfo messages. The AH
   decides which mouse model to use. The participants MUST support both
   mouse models.

   HIP supports both UTF-8 encoded unicode characters and other keyboard
   keys which are not defined in unicode such as function and control
   keys. For keyboard events publicly available Java virtual key-codes
   [keycodes] are used.

   The application and desktop sharing models defined in this document
   can be integrated into the IETF conferencing model. The Session
   Initiation Protocol (SIP) [9] can be used to intiate and control
   remote access. This allows the use of existing SIP mechanisms for
   confidentiality, authentication and authorization, user location,
   conferencing.

   Additional, optional mechanisms can enhance application and desktop
   sharing. Audio streams can be associated with a desktop or
   application; participant-side scaling can be used to optimize
   transmission of data to participants with a small screen; and it is

often useful to allow copy-and-paste between applications running on
a participant and those running on an AH. This document does not
define any such extensions.

The AH can support both multicast and unicast transmissions. For
unicast connections, either UDP or TCP can be used. The AH can share
an application to TCP participants, UDP participants, and several
multicast addresses in the same sharing session.

4.3. Participants using UDP

The AH controls the transmission rate for participants using UDP,
because UDP itself does not provide flow and congestion control.
Several simultaneous multicast sessions with different transmission
rates can be created at the AH.

Participants can join a sharing session anytime, and they need the
shared windows' information and full screen buffer before receiving
partial updates. Therefore, participants using UDP send an RCTP-
based feedback message, Picture Loss Indication (PLI) [RFC4585],
after joining the session. The AH prepares and transmits the
windows' state information and image of the whole shared region after
receiving a PLI message.

4.4. Participants using TCP

Since TCP provides reliable communication and flow control, it is
more suitable for unicast sessions. TCP participants may have
different bandwidths, so an algorithm which sends the updates at the
link speed of each participant is needed.

Neither TCP nor RTP declares the length of an RTP packet. Therefore,
RTP framing [RFC4571] is used to split RTP packets within the TCP
byte stream.

The AH prepares and transmits the windows' state information and
image of the whole shared region to the new participant, right after
the TCP connection establishment.

4.5. Protocol Overview

Application and desktop sharing protocol consists of two
subprotocols: remoting and human interface protocol (HIP). Remoting
messages transmit screen updates from AH to participants. HIP
messages transmit mouse and keyboard events from the participant to
the AH.

Remoting and HIP messages are RTP messages and they consist of an RTP

header, message type, message type parameter, window identifier, and
message specific payload [Figure 3]. The payload structure and
payload length are determined by the message type. The HIP messages
have different payload type than the remoting messages.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                                                               .
.                         RTP header                           .
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Msg Type    |    Parameter  |          Window ID            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                                                               .
.                                                               .
.                   Message Specific Payload                   .
.                                              +-+-+-+-+-+-+-+-+
.                                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
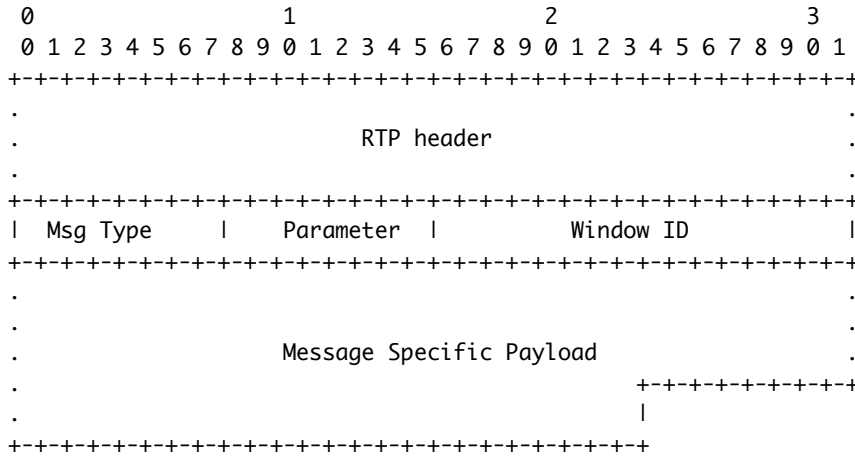
            Figure 3: Application sharing protocol message structure

4.5.1. Remoting Protocol Overview

   The remoting protocol consists of four messages from the AH to the
   participant and two control messages from participant to AH. The AH-
   to-participant messages are WindowStateInfo, RegionUpdate,
   MoveRectangle, and MousePointerInfo. The RTCP messages from UDP-
   based participant to AH are "Picture loss indication (PLI)" and "NACK
   request". Participants MUST implement all protocol messages
   described in this document.

   The WindowManagerInfo message informs the participants about ID of
   the windows, their positions and sizes, z-order, and their groupings.
   All remoting messages carry the window ID to identify the target of
   message. For TCP participants, the AH transmits WindowManagerInfo
   message right after establishing a connection. UDP participants send
   a "Picture loss indication (PLI)" to the AH as soon as they join the
   session. Receiving this PLI message, the AH transmits
   WindowManagerInfo message. The AH transmits RegionUpdate messages
   for updated regions. Whenever the shared window resizes or relocates
   the AH sends a WindowManagerInfo message. Similarly if the z-order
   of windows changes, the AH send a WindowManagerInfo message.
   MoveRectangle instructs the participant to move a region from one
   place to another, which is efficient for some drawing operations like
   scrolls. The MousePointerInfo message transmits the position and

icon of the mouse pointer. Some AHs may transmit pointer images
inside the RegionUpdate messages, so they may not need
MousePointerInfo message.

"Picture loss indication (PLI)" and "NACK request" are control
messages and they are transmitted as RTCP messages. The "NACK
request" is used only by UDP participants to request retransmission
of missing packets from the AH. AHs MAY support retransmissions.
PLI can be used by both UDP and TCP participants to request a full
screen refresh.

4.5.2. Human Interface Protocol (HIP) Overview

HIP consist of seven messages: namely, Mouse Pressed, Mouse Released,
Mouse Moved, Mouse Wheel Moved, Key Pressed, Key Released and Key
Typed. These messages are all from AH to participant and carried as
RTP messages. However, these HIP messages have different payload
type than the remoting messages.

5. Remoting Protocol

5.1. Payload Format

5.1.1. RTP Header Usage

   Marker Bit: The marker bit is used to indicate the last packet of a
      RegionUpdate. RegionUpdate messages may not fit into a single RTP
      packet. This enables the participant to finish decoding the
      picture, where it otherwise may need to wait for the next packet
      to explicitly know that. Only RegionUpdate messages can set this
      bit to one, all the other messages MUST set this bit to zero.

   Timestamp: The RTP timestamp indicate the time instance the remoting
      message is created at the AH. The RTP timestamp is based on a 90-
      kHz clock. If a RegionUpdate message occupies more than one
      packet, the timestamp SHALL be the same on all of those packets.
      Furthermore, the initial value of the timestamp MUST be random
      (unpredictable) to make known-plaintext attacks on encryption more
      difficult; see RTP [RFC3550].

   The remaining RTP header fields are used as specified in RFC 3550.

5.1.2. Payload Header

   All remoting protocol messages carry a common payload header
   [Figure 4] which follows the RTP header. Message type and parameter
   fields are 8 bit identifiers, whereas the window ID is a 16-bit
   identifier. The window ID field is unsigned and has a range of
   0-65535.

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   .                                                               .
   .                          RTP header                          .
   .                                                               .
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Msg Type     |   Parameter   |          Window ID            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

            Figure 4: Application sharing protocol message header

   The following four message types are defined in this document. The
   participants MUST implement all of them. Additional message types
   can be defined through IANA, as defined in Section 9. The
   participants MUST ignore unknown message types.

```
+-------+------------------+
| Value | Message Type     |
+-------+------------------+
|   1   | WindowManagerInfo |
|       |                  |
|   2   | RegionUpdate     |
|       |                  |
|   3   | MoveRectangle    |
|       |                  |
|   4   | MousePointerInfo |
+-------+------------------+
```

Table 1: Remoting Protocol Message Types

5.1.3. Payload Data

5.1.3.1. AH-to-participant messages

5.1.3.1.1. WindowManagerInfo

   The WindowManagerInfo message informs the participants about windows,
   their positions and sizes, z-order, and their groupings. This
   message transfers the complete window maneger state to the
   participants. Each shared window resize and relocation in any
   coordinate triggers a WindowmangerInfo message. Parameter and Window
   ID fields of payload header MUST be ignored. This message carries a
   message specific payload. One or more window records [Figure 5]
   follow the payload header.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Window ID   |   Group ID    |            Reserved           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Left                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              Top                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Width                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Heigth                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
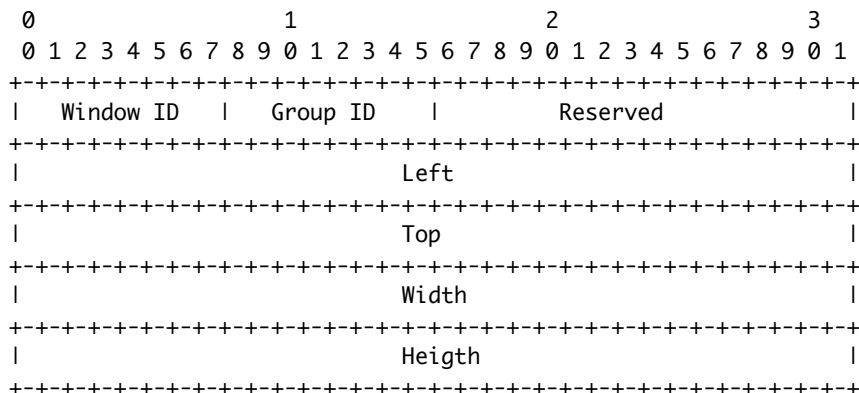
Figure 5: A Window Record

   Each window record is 20-bytes. The z-order information is given
   implicitly to the participants. The first record is near the

background and the last record is near the foreground. The "left"
and "Top" fields carries the upper-left coordinate of the window.
The "Width" and "Height" fields carries the width and the height of
the window, respectively. The units of left, top, width and height
are in pixels and they are unsigned. Each window is assigned a
"Window ID". The participant MUST create a window for each new
"Window ID" and MUST close this window after receiving a
WindowManagerInfo message which does not contain this "Window ID".
"Group ID" fields informs the participant about grouping of windows.
The AH MAY assign same "Group ID" to the windows which belongs to the
same process. Grouping information MAY be used by the participant
while relocating the windows or enforcing the z-order.

Figure 6 is an example WindowManagerInfo message for the three shared
windows in Figure 2. The participant MUST keep the existing window
image after a resize and relocation.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      1        |      0        |              0                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      1        |       1       |              0                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          220                                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          150                                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          350                                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          450                                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      2        |       2       |              0                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          850                                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          320                                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          160                                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          150                                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      3        |       1       |              0                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          450                                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          400                                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          350                                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          300                                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
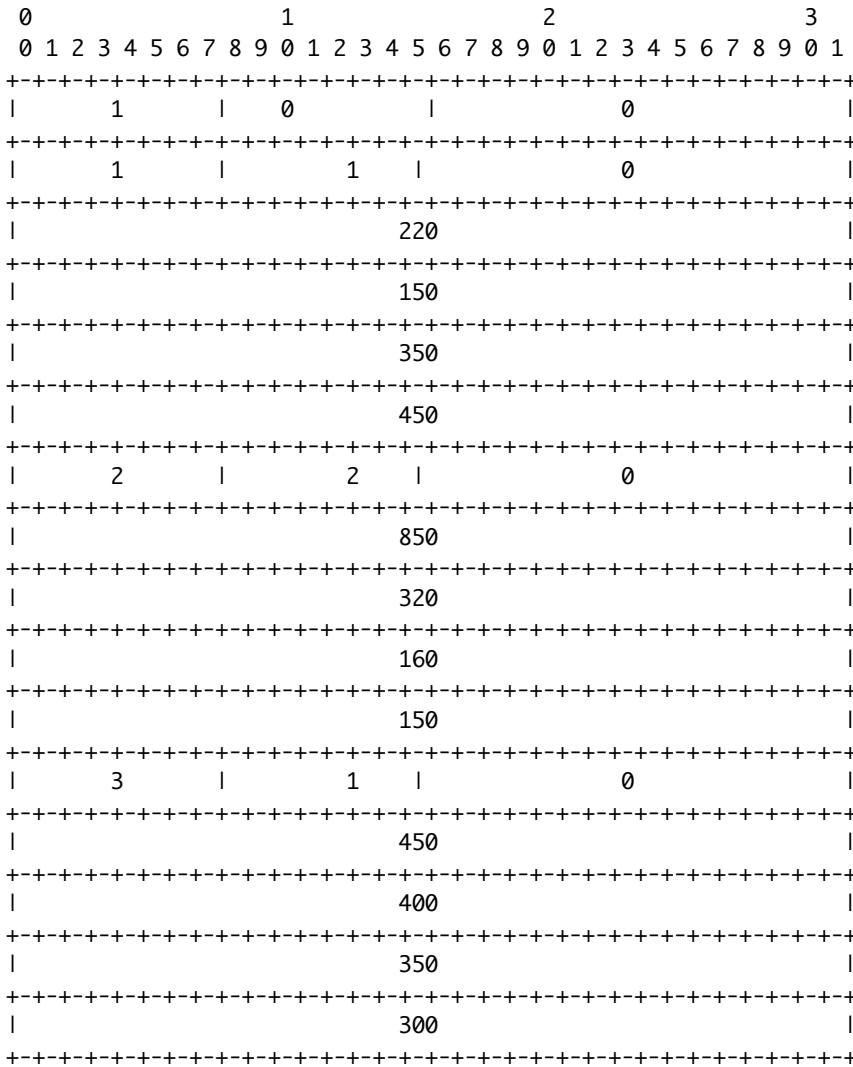
Figure 6: An example WindowManagerInfo message with three Window
Records

5.1.3.1.2. RegionUpdate

   The RegionUpdate message instructs the participant to update the
   specified region of a window with the new content. This message
   carries a message specific payload. This protocol supports all media
   types which have an RTP specification. It is possible that AH or
   participant may support only some media types. Therefore, they
   should negotiate supported media types during the session

establishment. The 8-bit "parameter" field of the payload header
will carry both the FirstPacket bit and the actual payload type of
the content. All AH and participant software implementations MUST
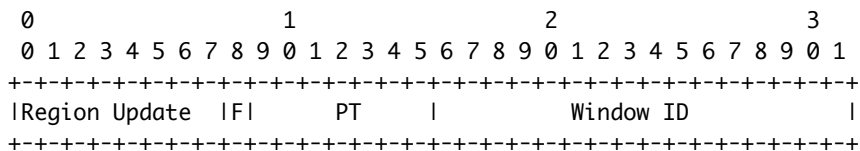support PNG images [I-D.boyaci-avt-png].

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Region Update  |F|    PT     |            Window ID            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: Region update header

If the content of the update does not fit into a single RTP message,
it will be carried in several RTP payloads. All the payloads will
carry the 32-bit payload header, while left and top fields are
carried only in the first RTP payload. The marker bit and
FirstPacket bit informs the particapants about the fragmentation.

```
+------------+----------------+----------------------+
| Marker bit | FirstPacket bit | Fragment Type       |
+------------+----------------+----------------------+
|     1      |       1         | Not Fragmented      |
|            |                 |                     |
|     0      |       1         | Start Fragment      |
|            |                 |                     |
|     0      |       0         | Continuation Fragment |
|            |                 |                     |
|     1      |       0         | End Fragment        |
+------------+----------------+----------------------+
```

Table 2: Fragmentation Info

Figure 8 displays an example RegionUpdate message. The RegionUpdate
is non fragmented, therefore both the marker bit in the RTP header
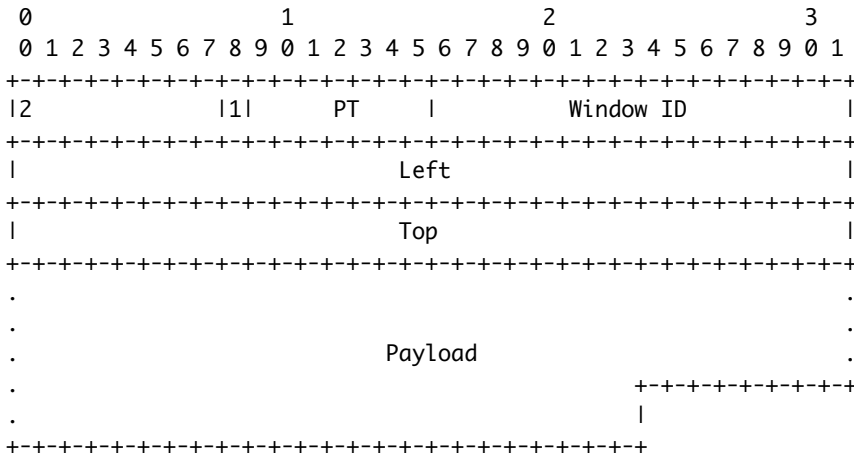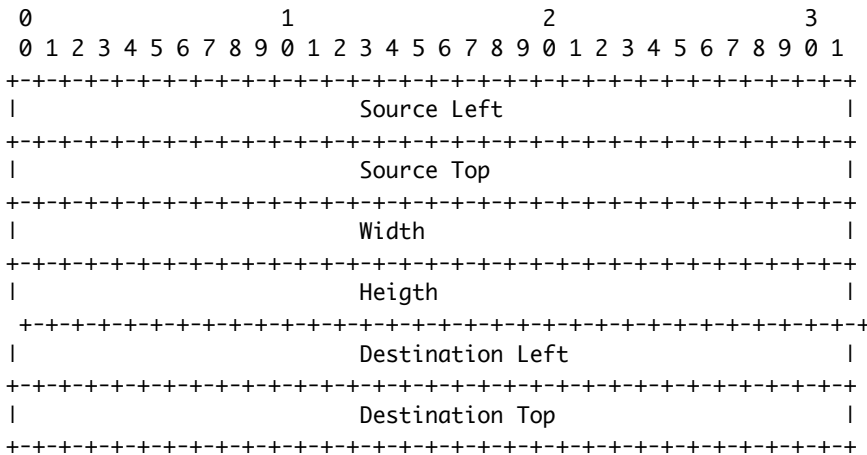and FirstPacket bit in the payload header is set to 1.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|2          |1|    PT     |           Window ID               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Left                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Top                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                                                             .
.                                                             .
.                          Payload                           .
.                                        +-+-+-+-+-+-+-+-+-+
.                                        |
.
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8: An example non fragmented RegionUpdate message

5.1.3.1.3. MoveRectangle

MoveRectangle message instructs the participant to move the specified region of a window to a new position. This message carries a message specific payload.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Source Left                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Source Top                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Width                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Heigth                             |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Destination Left                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Destination Top                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 9: Message specific payload for move rectangle

5.1.3.1.4. MousePointerInfo

Some AHs MAY include the mouse pointer image inside the RegionUpdate messages. However, some AHs MAY choose to inform the participant about the mouse position and icon explicitly. If the RegionUpdate

messages contain the mouse pointer icon, then the MousePointerInfo
message is unnecessary. When receiving this message, the participant
should draw the mouse pointer to the given position. This message
carries a message specific payload. The format of this message is
same as RegionUpdate message Section 5.1.3.1.2 except they have
different message types. The payload of MousePointerInfo message can
be only the left and top coordinates. In this case, the participant
MUST move the existing pointer image to the given coordinates.
Payload MAY carry both the left and top coordinates and the new image
of the mouse pointer. The participant MUST store and use this image
until a new image arrives from the AH. If the AH uses
MousePointerInfo messages, it SHOULD inform the late joiners about
the current position and image of mouse pointer.

5.1.3.2. Participant-to-AH messages

   Participants using UDP can send two RTCP messages to the AH. Late-
   joiners MAY inform the AH using the "Picture loss indication (PLI)"
   message in order to receive a full screen update. For the missing
   packets, UDP participants MAY send a "NACK Request".

5.1.3.2.1. Picture Loss Indication (PLI)

   "Picture Loss Indication (PLI)" message instructs the AH to generate
   a full screen update of the shared region. Before the full screen
   update, the AH will send a WindowManagerInfo message to inform the
   new participant about windows. Both TCP and UDP participants MAY
   transmit this message. The message format conforms to the "Picture
   Loss Indication (PLI)" section 6.3.1 of [RFC4585].

5.1.3.2.2. NACK Request

   "NACK Request" message informs the AH about missing RTP packets. The
   message format conforms to the "Generic NACK" section 6.2.1 of
   [RFC4585]. Multicast participants and AHs MAY take necessary
   precautions to prevent NACK storms such as waiting random amount of
   time before sending a "NACK Request" message.

6. Human Interface Protocol (HIP)

   Participants MAY send human interface events to AH in order to
   interact with the shared application.

6.1. Payload Format

6.1.1. RTP Header Usage

   Marker Bit: Must be set to zero by the transmitter and ignored by
      the receiver.

   Timestamp: The RTP timestamp indicate the time instance the keyboard
      or mouse event is occured at the participant. The RTP timestamp
      of HIP messages is based on a 90-kHz clock. Furthermore, the
      initial value of the timestamp MUST be random (unpredictable) to
      make known-plaintext attacks on encryption more difficult; see RTP
      [RFC3550].

   The remaining RTP header fields are used as specified in RFC 3550.

6.1.2. Payload Header

   All keyboard and mouse event messages carry the same common header
   Figure 4 discussed in Section 5.1.2. Window ID MUST be set to the ID
   of the window where keyboard or mouse event take place.

   The following HIP message types are defined:

```
+-------+----------------+
| Value | Message Type   |
+-------+----------------+
|   1   | MousePressed   |
|       |                |
|   2   | MouseReleased  |
|       |                |
|   3   | MouseMoved     |
|       |                |
|   4   | MouseWheelMoved |
|       |                |
|   5   | KeyPressed     |
|       |                |
|   6   | KeyReleased    |
|       |                |
|   7   | KeyTyped       |
+-------+----------------+
```

                    Table 3: HIP Message Types

6.1.3. Payload Data

6.1.3.1. MousePressed

   The MousePressed message instructs the AH to generate a mouse pressed
   event at the given coordinates of the screen. This message carries a
   message specific payload. The coordinates are relative to the
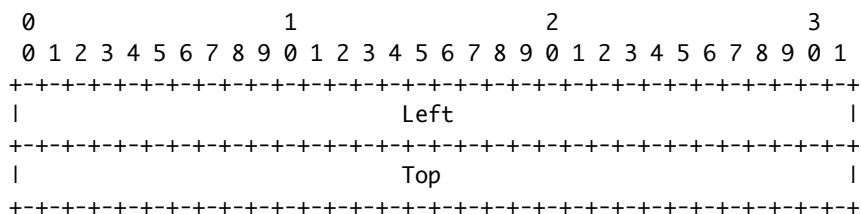   participant side window. The "parameter" field of this message
   carries the mouse button information. The values of 1, 2 and 3 are
   defined for left, right, and middle button, respectively. The AH and
   participant MAY negotiate additional values for other mouse buttons.
   The AH MAY ignore unrecognized values.

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                              Left                             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                              Top                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
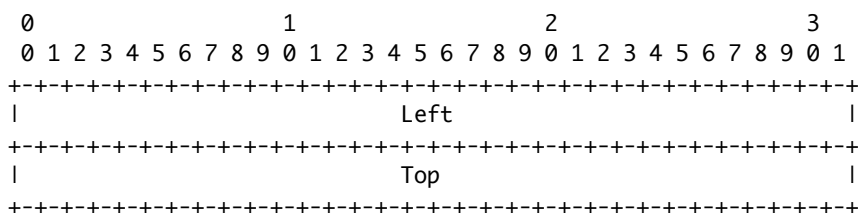
            Figure 10: Message specific payload for mouse pressed

6.1.3.2. MouseReleased

   The MouseReleased message instructs the AH to generate a mouse
   released event at the given coordinates of the screen. This message
   carries a message specific payload. The coordinates are relative to
   the participant side window. The "parameter" field of this message
   carries the mouse button information. The values of 1, 2 and 3 are
   defined for left, right, and middle button, respectively. Other
   values MAY be defined for other mouse buttons. The AH MAY ignore
   unrecognized values.

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                              Left                             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                              Top                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
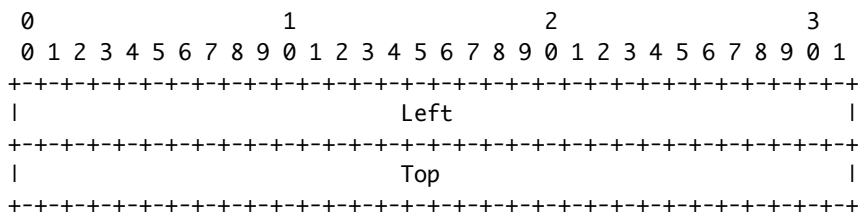
            Figure 11: Message specific payload for mouse released

6.1.3.3. MouseMoved

   The MouseMoved message instructs the AH to move the mouse pointer to
   the the given coordinates of the screen. This message carries a
   message specific payload. The coordinates are relative to the
   participant side window.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Left                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Top                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

           Figure 12: Message specific payload for mouse moved

6.1.3.4. MouseWheelMoved

   The MouseWheelMoved message instructs the AH to generate a mouse
   wheel moved event at given coordinates of the screen. This message
   carries a message specific payload. The coordinates are relative to
   the participant side window. The "info" field carries the wheel
   rotation amount as "120 * (number of notches)". A regular mouse
   wheel has discrete, evenly spaced notches. When user rotates the
   wheel, a wheel message is sent to OS as each notch is encountered.
   The "info" field does not carry number of notches in order to support
   a smooth-scrolling wheel. Instead, "info" field carries each notch
   as 120. A regular mouse MAY only send multiples of 120. A smooth-
   scrolling wheel MAY send any values. A positive value indicates that
   the wheel was rotated forward, away from the user; a negative value
   indicates that the wheel was rotated backward, toward the user.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Left                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Top                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Info                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
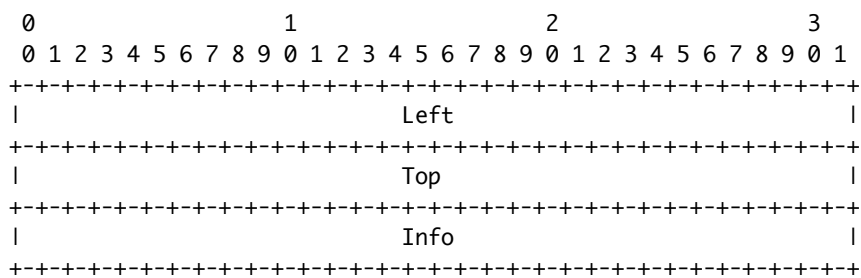
        Figure 13: Message specific payload for mouse wheel moved

6.1.3.5. KeyPressed

   The KeyPressed message instructs the AH to generate a key pressed
   event. This message carries a message specific payload. Java
   virtual keycodes are used and they are publicly available on the
   openJDK website [keycodes]. The actual values are inside the
   KeyEvent.java file. For example, F1 key is defined as "int VK_F1 =
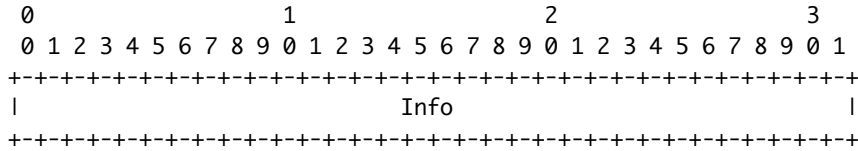   0x70;" in KeyEvent.java.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Info                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

           Figure 14: Message specific payload for key pressed

6.1.3.6. KeyReleased

   The KeyReleased message instructs the AH to generate a key released
   event. This message carries a message specific payload. Java
   keycodes are used and they are publicly available at openJDK website
   [keycodes]. The actual values are inside the KeyEvent.java file.
   For example, F1 key is defined as "int VK_F1 = 0x70;" in
   KeyEvent.java. A KeyReleased event for a key without a prior
   KeyPressed event for this key is acceptable.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Info                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
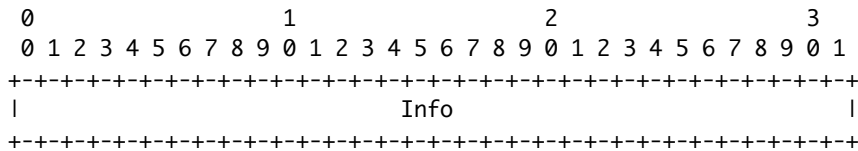
           Figure 15: Message specific payload for key released

6.1.3.7. KeyTyped

   KeyTyped message instructs the AH to inject some number of UTF-8
   encoded characters into operating systems input queue. This message
   carries a message specific payload. There is no padding for the
   UTF-8 string. The participant MUST send more than one KeyTyped
   message if the string does not fit into a single KeyTyped packet.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                                                               .
.                                                               .
.                         UTF-8 String                         .
.                                              +-+-+-+-+-+-+-+-+
.                                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

             Figure 16: Message specific payload for key released

7. Implementation Notes

   Application hosts shouldn't blindly send every screen update they
   observed to the participants. Instead, they should monitor the state
   of their TCP transmission buffers (through mechanisms such as the
   select() command) and only send the most recent screen data when
   there is not a backlog. This will prevent screen latency for
   rapidly-changing images, when a viewer usually only needs to see the
   final state of the image.

8. Security Considerations

   RTP packets using the payload format defined in this specification
   are subject to the security considerations discussed in the RTP
   specification [RFC3550].

   Application sharing inherently exposes the shared applications to
   risks by malicious participants. They may, for example, access
   resources beyond the application itself, e.g., by installing or
   running scripts. It may be difficult to constrain access to specific
   user data, e.g., a specific set of slides, unless the user
   application can be sandboxed or run in some kind of "jail", with the
   sandbox control outside the view of the remoting protocol.

9. IANA Considerations

   The IANA has created a new registry for Application and Desktop
   Sharing Parameters called "Application and Desktop Sharing
   parameters". This new registry has a number of subregistries, which
   are described in the following sections.

9.1. Remoting Message Types Subregistry

   This section establishes the Remoting Message Types subregistry under
   the Application and Desktop Sharing Parameters registry. As per the
   terminology in RFC 2434 [RFC2434], the registration policy for
   Remoting Message Types shall be "Specification Required". For the
   purposes of this subregistry, the Remoting Message Types for which
   IANA registration is requested MUST be defined by a standards-track
   RFC. Such an RFC MUST specify the Remoting Message Type's value,
   name, format, and semantics.

   For each Remoting Message Type, the IANA registers its value, its
   name, and the reference to the RFC where the Remoting Message Type is
   defined. The following table contains the initial values of this
   subregistry.

```
            +-------+------------------+------------+
            | Value | Message Type     | Reference  |
            +-------+------------------+------------+
            |   1   | WindowManegerInfo | [RFC nnnn] |
            |       |                  |            |
            |   2   | RegionUpdate     | [RFC nnnn] |
            |       |                  |            |
            |   3   | MoveRectangle    | [RFC nnnn] |
            |       |                  |            |
            |   4   | MousePointerInfo | [RFC nnnn] |
            +-------+------------------+------------+
```

      Table 4: Initial values of the Remoting Message Type subregistry

9.2. HIP Message Types Subregistry

   This section establishes the HIP Message Types subregistry under the
   Application and Desktop Sharing Parameters registry. As per the
   terminology in RFC 2434 [RFC2434], the registration policy for HIP
   Message Types shall be "Specification Required". For the purposes of
   this subregistry, the HIP Message Types for which IANA registration
   is requested MUST be defined by a standards-track RFC. Such an RFC
   MUST specify the HIP Message Type's value, name, format, and
   semantics.

For each HIP Message Type, the IANA registers its value, its name,
and the reference to the RFC where the HIP Message Type is defined.
The following table contains the initial values of this subregistry.

```
+-------+----------------+------------+
| Value | Message Type   | Reference  |
+-------+----------------+------------+
|  121  | MousePressed   | [RFC nnnn] |
|       |                |            |
|  122  | MouseReleased  | [RFC nnnn] |
|       |                |            |
|  123  | MouseMoved     | [RFC nnnn] |
|       |                |            |
|  124  | MouseWheelMoved| [RFC nnnn] |
|       |                |            |
|  125  | KeyPressed     | [RFC nnnn] |
|       |                |            |
|  126  | KeyReleased    | [RFC nnnn] |
|       |                |            |
|  127  | KeyTyped       | [RFC nnnn] |
+-------+----------------+------------+
```

Table 5: Initial values of the HIP Message Type subregistry

9.3. Media Type Registrations

Following the guidelines in RFC 4855 [RFC4855] and RFC 4288
[RFC4288], this section registers new 'application' media subtypes
for remoting and hip.

9.3.1. Registrations of Media Type application/remoting

Type name: application

Subtype name: remoting

Required parameters:

   rate: RTP timestamp clock rate, which is equal to the sampling
      rate. The typical rate is 90000; other rates may be specified.

   retransmissions: Informs the participants whether the AH supports
      UDP retransmissions. The possible values are yes and no.

   Optional parameters: none

   Encoding considerations: This media type is framed binary data (see
      RFC 4288, Section 4.8).

   Security considerations: See Section Section 8 of RFC nnnn

   Interoperability considerations: none

   Published specification: RFC nnnn

   Additional information: none

   Person & email address to contact for further information: Omer
      Boyaci <boyaci@cs.columbia.edu> and Henning Schulzrinne
      <hgs@cs.columbia.edu>

   Intended usage: COMMON

   Restrictions on usage: This media type depends on RTP framing, and
      hence is only defined for transfer via RTP (RFC 3550). Transfer
      within other framing protocols is not defined at this time.

   Applications that use this media type: Application and Desktop
      sharing tools. Remote tutoring tools.

   Author: Omer Boyaci and Henning Schulzrinne

   Change controller: IETF Audio/Video Transport working group
      delegated from the IESG.

9.3.2. Registrations of Media Type application/hip

   Type name: application

   Subtype name: hip

   Required parameters:

      rate: RTP timestamp clock rate, which is equal to the sampling
         rate. The typical rate is 90000; other rates may be specified.

   Optional parameters: none

   Encoding considerations: This media type is framed binary data (see
      RFC 4288, Section 4.8).

Security considerations: See Section Section 8 of RFC nnnn

Interoperability considerations: none

Published specification: RFC nnnn

Additional information: none

Person & email address to contact for further information: Omer
   Boyaci <boyaci@cs.columbia.edu> and Henning Schulzrinne
   <hgs@cs.columbia.edu>

Intended usage: COMMON

Restrictions on usage: This media type depends on RTP framing, and
   hence is only defined for transfer via RTP (RFC 3550). Transfer
   within other framing protocols is not defined at this time.

Applications that use this media type: Application and Desktop
   sharing tools.

Author: Omer Boyaci and Henning Schulzrinne

Change controller: IETF Audio/Video Transport working group
   delegated from the IESG.

10. Mapping to SDP

   The information carried in the Media Type specification has a
   specific mapping to fields in the Session Description Protocol (SDP)
   [RFC4566], which is commonly used to describe RTP sessions. When SDP
   is used to specify sessions, the mappings are as follows:

10.1. Mapping remoting Media Type Parameters into SDP

      The media type ("application") goes in SDP "m=" as the media name.

      The media subtype ("remoting") goes in SDP "a=rtpmap" as the
      encoding name.

      The parameter "rate" also goes in "a=rtpmap" as the clock rate.

      The mandated parameter "retransmissions" MUST be included in the
      SDP "a=fmtp" attribute.

10.2. Mapping hip Media Type Parameters into SDP

      The media type ("application") goes in SDP "m=" as the media name.

      The media subtype ("hip") goes in SDP "a=rtpmap" as the encoding
      name.

      The parameter "rate" also goes in "a=rtpmap" as the clock rate.

10.3. SDP Example

   The following example shows an example SDP usage. This SDP message
   is from AH to participant. HIP stream and BFCP server is connected
   to each other via "label" and "m-stream" attributes according to SDP
   Format for BFCP Streams [RFC4583]. This SDP message informs the
   participant that AH supports both TCP and UDP for remoting. The AH
   supports UDP retransmissions, so participants MAY send NACK requests
   for missing packets. The port numbers MUST be same if AH is remoting
   the same content over both TCP and UDP. In this example, AH is
   sending the same content from port 6000. It is possible that AH may
   have more than one remoting session, in this case each session MUST
   use different port numbers.

```
m=application 50000 TCP/BFCP *
a=floorid:0 m-stream:10
m=application 6000 RTP/AVP 99
a=rtpmap:99 remoting/90000
a=fmtp: retransmissions=yes
m=application 6000 TCP/RTP/AVP 99
a=rtpmap:99 remoting/90000
m=application 6006 TCP/RTP/AVP 100
a=rtpmap:99 hip/90000
a=label:10
```

Appendix A. Using BFCP for Application and Desktop Sharing

   Application and Desktop Sharing tools MAY utilize Binary Floor
   Control Protocol (BFCP) [RFC4582] for managing the ownership of AH's
   human interface devices (HID).

   BFCP defines several messages, but only five of them is a MUST for
   Application and Desktop Sharing, namely "Floor Request", "Floor
   Release", "Floor Granted", "Floor Released" and "Floor Request
   Queued".

   In Application and Desktop Sharing context, the floor is the AH's
   HIDs. In this context, it is possible that the AH MAY temporarily
   block HID events without revoking the floor control. For example,
   the AH MAY temporarily block HID events if the shared application
   loses the focus or is covered by a non-shared application. The AH
   informs the current floor holder about the status of HIDs via STATUS-
   INFO attribute of "Floor Granted" messages. The participant MAY
   receive several "Floor Granted" messages with different "HID Status"
   values. Participant applications MAY inform the user about current
   "HID Status". HID Status values are 16-bit unisgned values and are
   defined as:

```
                    +-------+-----------------------+
                    | Value | Status                |
                    +-------+-----------------------+
                    |   0   | STATE_NOT_ALLOWED      |
                    |   1   | STATE_KEYBOARD_ALLOWED |
                    |   2   | STATE_MOUSE_ALLOWED    |
                    |   3   | STATE_ALL_ALLOWED      |
                    +-------+-----------------------+
```

                    Figure 17: HID Status Values

11. References

11.1. Normative References

   [I-D.boyaci-avt-png]
             Boyaci, O. and H. Schulzrinne, "RTP Payload Format for
             Portable Network Graphics (PNG) image",
             draft-boyaci-avt-png-00 (work in progress),
             September 2008.

   [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
             Requirement Levels", BCP 14, RFC 2119, March 1997.

   [RFC2327] Handley, M. and V. Jacobson, "SDP: Session Description
             Protocol", RFC 2327, April 1998.

   [RFC2435] Berc, L., Fenner, W., Frederick, R., McCanne, S., and P.
             Stewart, "RTP Payload Format for JPEG-compressed Video",
             RFC 2435, October 1998.

   [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston,
             A., Peterson, J., Sparks, R., Handley, M., and E.
             Schooler, "SIP: Session Initiation Protocol", RFC 3261,
             June 2002.

   [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V.
             Jacobson, "RTP: A Transport Protocol for Real-Time
             Applications", STD 64, RFC 3550, July 2003.

   [RFC3984] Wenger, S., Hannuksela, M., Stockhammer, T., Westerlund,
             M., and D. Singer, "RTP Payload Format for H.264 Video",
             RFC 3984, February 2005.

   [RFC4571] Lazzaro, J., "Framing Real-time Transport Protocol (RTP)
             and RTP Control Protocol (RTCP) Packets over Connection-
             Oriented Transport", RFC 4571, July 2006.

   [RFC4582] Camarillo, G., Ott, J., and K. Drage, "The Binary Floor
             Control Protocol (BFCP)", RFC 4582, November 2006.

   [RFC4583] Camarillo, G., "Session Description Protocol (SDP) Format
             for Binary Floor Control Protocol (BFCP) Streams",
             RFC 4583, November 2006.

   [RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey,
             "Extended RTP Profile for Real-time Transport Control
             Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585,
             July 2006.

11.2. Informative References

   [RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an
              IANA Considerations Section in RFCs", BCP 26, RFC 2434,
              October 1998.

   [RFC4288] Freed, N. and J. Klensin, "Media Type Specifications and
              Registration Procedures", BCP 13, RFC 4288, December 2005.

   [RFC4855] Casner, S., "Media Type Registration of RTP Payload
              Formats", RFC 4855, February 2007.

   [T120]   International Telecommunication Union, "Data Protocols for
              Multimedia Conferencing", X ITU-T Recommendation T.120.

   [X]   Scheifler, R., "X Window System Protocol", X Consortium
              Standard X Version 11, November 2004.

   [keycodes]
              OpenJDK Community, "Java key-codes", <http://
              hg.openjdk.java.net/jdk7/awt-gate/jdk/archive/tip.zip>.

   [theora]   Xiph.org Foundation, "Theora".

Authors' Addresses

   Omer Boyaci
   Columbia University
   Dept. of Computer Science
   1214 Amsterdam Avenue
   New York, NY 10027
   US

   Email: boyaci@cs.columbia.edu

   Henning Schulzrinne
   Columbia University
   Dept. of Computer Science
   1214 Amsterdam Avenue
   New York, NY 10027
   US

   Email: schulzrinne@cs.columbia.edu

Full Copyright Statement

Intellectual Property

The IETF takes no position regarding the validity or scope of any
Intellectual Property Rights or other rights that might be claimed to
pertain to the implementation or use of the technology described in
this document or the extent to which any license under such rights
might or might not be available; nor does it represent that it has
made any independent effort to identify any such rights. Information
on the procedures with respect to rights in RFC documents can be
found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any
assurances of licenses to be made available, or the result of an
attempt made to obtain a general license or permission for the use of
such proprietary rights by implementers or users of this
specification can be obtained from the IETF on-line IPR repository at
http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any
copyrights, patents or patent applications, or other proprietary
rights that may cover technology that may be required to implement
this standard. Please address the information to the IETF at
ietf-ipr@ietf.org.