Home Networking (homenet)
Internet-Draft
Intended status: Informational
Expires: January 15, 2014

L. Ruminski
M. Kutzner
A. Dymek
S. Kwiatkowski
M. Langa
Nicolaus Copernicus University
Faculty of Mathematics and Computer Science

July 15, 2013

Grazed and Lightweight Open Protocol (GaLOP), v. 1.0
draft-ruminski-homenet-galop-proto-07

Status of this Memo

Abstract

This informational memo specifies a Grazed and Lightweight Open Protocol (GaLOP), designed to exchange information within the Hackney project [HACKNEY]. The document describes messages' structures, defined message types used in the communication and standard connection scenarios.

Copyright Notice

Introduction

The main goal for the protocol is to provide fast and reliable communication between mobile devices (client software) and Personal Computer (server software) via Bluetooth with minimal data size needed to realize full functionality of the project. The protocol was designed to be easy to extend - adding new features and developing clients for other mobile platforms should be as easy as possible.

Conventions used in this document

In examples, "CL:" and "SR:" indicate lines sent by the client and server respectively.

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 RFC-2119.

The following acronyms are used in this document:

SID	- Session Identifier
GID	- Gamepad functionality Identifier
DID	- Device Identifier
UUID	- Universally Unique Identifier
RFCOMM	- Radio frequency communication protocol
J2ME	- Java Platform, Micro Edition
MT	- Message Data Type
PL	- Payload

GaLOP in Bluetooth architecture model

By default, protocol uses the communication provided by RFCOMM. It is assigned to the same middleware layer, as RFCOMM and to application layer.

Message data definition

The basic unit of the communication in Hackney project is the message specified by MessageData class. The data is an array of bytes.
Each message consists of 2-byte header and at least 2-byte payload.

Byte 0 specifies the MT, byte 1 describes the SID in case of already established connection or DID instead. Header length is constant.

30-byte payload is intended for use only by messages of type STRING_EXCHANGE. Other messages use 2-byte payload.

```
        0
        0   1   2   3   4
        +-+-+-+-+-+-+-+-+
        |MT |SID|  PL   |
        +-+-+-+-+-+-+-+-+
```

Default message structure

```
0
0   1   2   3   4   5   6   7   8
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|MT |SID|      PL               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              PL               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              PL               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              PL               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type STRING_EXCHANGE message structure

Message data types with description

General messages

CONNECTION_HELLO - 1
Description: Message used for connection establishment.
Byte 1: DID, in case the message sent by client or SID as a response from a server.
Payload: Ignored.

CONNECTION_GOODBYE - 2
Description: Connection termination.
Byte 1: Ignored.

Payload: Ignored.

Mouse handler messages

MOUSE_NUMPAD_START_MOVE - 10

Description: Pressing a mouse button or starting to move the mouse using the keypad or joystick. Action is performed until message of type MOUSE_NUMPAD_STOP is received.

Byte 1: SID.

Payload:

Byte 2: Key code as in Appendix A.

MOUSE_NUMPAD_STOP - 11

Description: Stopping the action related with the message of type MOUSE_NUMPAD_START_MOVE.

Byte 1: SID.

Payload:

Byte 2: Key code as in Appendix A.

MOUSE_SCREEN_PRESSED - 12

Description: Touching the screen before the proper mouse action.

Byte 1: SID.

Payload:

Byte 2: 1,

Byte 3: Ignored.

MOUSE_SCREEN_MOVE - 13

Description: Moving the mouse, using touch screen.

Byte 1: SID.

Payload:

Byte 2: Abscissa of direction vector in which the motion is performed,

Byte 3: Ordinate of direction vector in which the motion is performed.

MOUSE_SCREEN_RELEASED - 14

Description: Releasing the screen.

Byte 1: SID.

Payload:

Byte 2: 1,

Byte 3: Ignored.

Keyboard handler messages

KEYBOARD_PRESSED - 20

Description: Pressing the keyboard.

Byte 1: SID.

Payload:

```
                                  Byte 2: ASCII of key value (in case of
                                  letter - ASCII code of capital letter),
```

Byte 3: Modifiers.

Modifier is a special key, which change the standard function of keys.
Available modifiers, with the values:
SHIFT: 2^0,
CTRL: 2^1,
ALT: 2^2,
ALTGR: 2^3,
SUPER/META/WIN: 2^4.
The combination of modifiers is possible as the sum of their values - for example, SHIFT and ALT: 2^0 + 2^2 = 1 + 4 = 5.
It's not expected to use more than three modifiers at the same time.

Hotkeys (keyboard shortcuts) and application management

HOTKEY_APPLIST_REQUEST - 30
Description: Requesting a list of applications that have keyboard shortcuts defined on the server side.
Byte 1: SID.
Payload:
Byte 2: 1,
Byte 3: Ignored.

HOTKEY_APPLIST_COUNT - 31
Description: Count of the items on application list which is sent by the server to client.
Byte 1: SID.
Payload:
Byte 2: Count of items on the list of apps,
Byte 3: Ignored.

HOTKEY_APP_CHOSED - 32
Description: Application selection from a list received from the server.
Byte 1: SID.
Payload:
Byte 2: Number of item from the list,
Byte 3: Ignored.

HOTKEY_APPHOTKEY_REQUEST - 33
Description: Requesting a list of keyboard shortcuts for the chosed application.
Byte 1: SID.

```
                                   Payload:
                                           Byte 2: 1,
```

Byte 3: Ignored.

HOTKEY_APPHOTKEY_COUNT - 34

Description: Count of the items on keyboard shortcuts list which is sent by the server to client.
Byte 1: SID.
Payload:
Byte 2: Count of items on the list of hotkeys,
Byte 3: Ignored.

HOTKEY_USED - 35

Description: Number of used shortcut.
Byte 1: SID.
Payload:
Byte 2: Number of item from the list of hotkeys,
Byte 3: Ignored.

STRING_EXCHANGE - 39

Description: The message used to transfer applications and hotkeys descriptions. Is used with HOTKEY_APPLIST_COUNT and HOTKEY_APPHOTKEY_COUNT types messages.
As the only type uses extended payload, with a length of 30 bytes.
Byte 1: SID.
Payload:
Bytes from 2 to 30: String encoded in UTF-8 (15 characters, each encoded by 2 bytes).

Gamepad handler messages

GAMEPAD_ACCEL_MOVE - 40

Description: Begin action/motion by changing the orientation
of the device against its default settings.
Measurement is
made by the accelerometer.
Byte 1: SID.
Payload:
Byte 2: the code corresponding to a change in
orientation against the default settings (landscape position):
Tilt the device to the left - 2,
Tilt the device to the right - 4,

Tilt the device to the left against
to its height - 8,

Tilt the device to the right against to its height - 16,
Tilt the device to the front - 32,
Tilt the device to the back - 64,
Byte 3: GID set by GAMEPAD_PLAYER_SET type message.

GAMEPAD_ACCEL_STOP - 41
Description: Stopping the action performed by GAMEPAD_ACCEL_MOVE type message.
Byte 1: SID.
Payload:
Byte 2: the code of the action which should be terminated.
Values are identical to values defined for GAMEPAD_ACCEL_MOVE type message.
Byte 3: GID set by GAMEPAD_PLAYER_SET type message.

GAMEPAD_KEY_PRESSED - 42
Description: The message used when special function key is pressed. These keys work with the accelerometer, together performing the functionality of a gamepad controller.
Byte 1: SID.
Payload:
Byte 2: code of a special function key:
'A' button - 2,
'B' button - 4,
'C' button - 8,
'D' button - 16,
'E' button - 32,
'F' button - 64,
Byte 3: GID set by GAMEPAD_PLAYER_SET type message.

GAMEPAD_KEY_RELEASED - 43
Description: Releasing a special function key.
Byte 1: SID.
Payload:
Byte 2: code of a special function key.
Values are identical to values defined for GAMEPAD_KEY_PRESSED type message,
Byte 3: GID set by GAMEPAD_PLAYER_SET type message.

GAMEPAD_PLAYER_SET - 44
Description: Message used to assign a set of

keys to
player actions or release a set by the player.

```
                        Byte 1: SID.
                        Payload:
                                Byte 2: actual GID, or 0 if message
                                is a response,
                                Byte 3: new/proposed GID.
```

Defined Client Device Identifiers

Java Platform, Micro Edition - 1,
Android - 2.
Values from 3 to 9 are reserved for future clients for other mobile platforms.

Usage scenarios

Connection establishment ("handshake")

```
        CL:        CONNECTION_HELLO with DID at byte 1 as above
        SR:        CONNECTION_HELLO with SID at byte 1 or
                   CONNECTION_GOODBYE

        If Client was identified as Android Device:
        SR:        GAMEPAD_PLAYER_SET
```

Interaction with the server without response being sent by the Server

```
        CL:        MOUSE_NUMPAD_START_MOVE or
                   MOUSE_NUMPAD_STOP or
                   MOUSE_SCREEN_PRESSED or
                   MOUSE_SCREEN_MOVE or
                   MOUSE_SCREEN_RELEASED or
                   KEYBOARD_PRESSED or
                   GAMEPAD_ACCEL_MOVE or
                   GAMEPAD_ACCEL_STOP or
                   GAMEPAD_KEY_PRESSED or
                   GAMEPAD_KEY_RELEASED
```

Hotkey usage

```
        CL:        HOTKEY_APPLIST_REQUEST
        SR:        HOTKEY_APPLIST_COUNT w/count at byte 2 as LC
        SR:        STRING_EXCHANGE (LC times)
        CL:        HOTKEY_APP_CHOSED
        CL:        HOTKEY_APPHOTKEY_REQUEST
        SR:        HOTKEY_APPHOTKEY_COUNT w/cnt. as above as HC
        SR:        STRING_EXCHANGE (HC times)
        CL:        HOTKEY_USED
```

Changing assigned player set

```
                 CL:        GAMEPAD_PLAYER_SET with actual GID at byte 2
```

and proposed GID at byte 3

SR: GAMEPAD_PLAYER_SET with 0 at byte 2 and new GID at byte 3, if available; 0 otherwise

Connection termination

CL: CONNECTION_GOODBYE
SR: CONNECTION_GOODBYE

Security Considerations

The Protocol was designed to be simple and lightweight.

The safety of its use depends on the security mechanisms used by Bluetooth Technology and implementation of Hackney Server.

The connection between client applications and server is not possible if the UUIDs on both sides are not identical.

The computer must be discoverable. This forces a conscious interaction from the user while using a computer.

Android-based devices require pairing with a computer before connection will be made. Pairing J2ME-enabled devices depends on Bluetooth implementation for Java virtual machine (but still UUIDs must be the same). Default Hackney server implementation uses so-called allow-list, allowing user to control client access to the server (for example, prohibit access to devices other than actually connected).

IANA Considerations

This specification makes no request of the IANA.

References

Informative References

[HACKNEY] <http://hackney.tk>

Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot.

Appendix A. Keycodes with assigned Actions for 10 and 11 Message Types

49 or -5 - Left Mouse Button,
51 - Right Mouse Button,
53 or -1 - Move up,
56 or -2 - Move down,
52 or -3 - Move left,
54 or -4 - Move right.

Author's Addresses

Lukasz Ruminski
Nicolaus Copernicus University
Faculty of Mathematics and Computer Science
Chopin Street 12/18
87-100 Torun
PL

Email: protazy@mat.umk.pl

Michal Kutzner
Nicolaus Copernicus University
Faculty of Mathematics and Computer Science
Chopin Street 12/18
87-100 Torun
PL

Email: kucyk@mat.umk.pl

Adrian Dymek
Nicolaus Copernicus University
Faculty of Mathematics and Computer Science
Chopin Street 12/18
87-100 Torun
PL

Email: include@mat.umk.pl

Szymon Kwiatkowski
Nicolaus Copernicus University
Faculty of Mathematics and Computer Science
Chopin Street 12/18
87-100 Torun
PL

Email: szymonk@mat.umk.pl

Marcin Langa
Nicolaus Copernicus University
Faculty of Mathematics and Computer Science
Chopin Street 12/18
87-100 Torun
PL

Email: iglis@mat.umk.pl