Group 12: The Remote SmartHouse Control (RSHC) Protocol

CS544 Spring 2013, Drexel University

Ryan Corcoran
ryan.m.corcoran@gmail.com

Amber Heilman
alh93@drexel.edu

Michael Mersic
mpm76@drexel.edu

Ariel Stolerman
ams573@cs.drexel.edu

April 27, 2013

Abstract

This is the abstract.

Contents

1 Service Description

This is the service description section. This is a sample citation [1], and a sample figure is shown in Fig. 1.

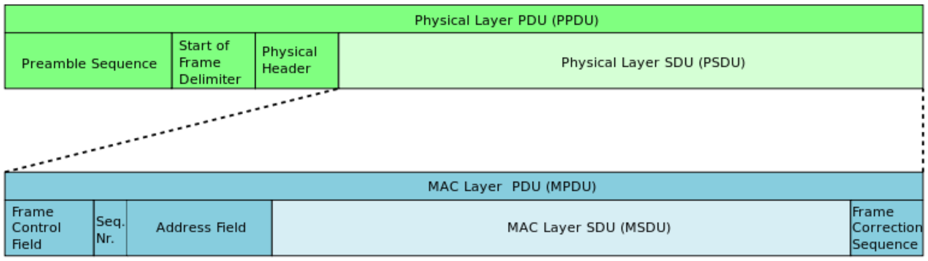

Figure 1: PDU sample figure, taken from [1]

2 Message Definition – PDU

This is the message definition section.

2.1 Addressing

2.2 Flow Control

2.3 PDU Definitions

RSHC communication includes 3 stages:

1. *Handshake.* Agree on protocol version and conduct authentication.
2. *Initialization.* Server sends an `init` message to the client with control information.
3. *Normal Protocol Interaction.* Client sends messages at will, and may receive responses from the server. Normally messages begin with a type (1 byte) followed by message-specific data: client queries or actions and server replies or confirmations.

All messages are constructed of a stream of bytes, either of a fixed size based on the message type or a custom size indicated in the message. In the rest of the section, PDU chunks are formatted as: `[`$\langle \mathit{title}|\mathit{value}\rangle$`:`$\langle \#\mathit{bytes}\rangle$`]`, for instance: `[0x02:1]`. PDUs are preceded by either `S` or `C`, indicating these messages are sent by the server or the client, respectively (or both). Next, each phase is discussed with a detailed description of the RSHC message PDUs.

2.3.1 Handshake

This is a synchronous phase for determining version and authentication. To initialize the communication, the client pokes the server with the message `[0x00:1]`. The server then sends the highest version it supports, and the client responds with the decided version, which should not exceed the version supported by the server. Each of these messages consists of 10 bytes of ASCII characters in the format “`RSHC xxxx\n`” where `xxxx` is the zero-padded version. For instance:

```
S|C > [‘RSHC 0001\n’: 10]
```

If the connection failed since the server does not support the requested version, it sends an error messages which includes the reason and closes the connection:

```
S > [0x01: 1][#err-msg-chars: 1][err-msg: #err-msg-chars]
```

Otherwise, the server sends the client an accept message, followed by a 16-byte challenge for authentication:

```
S > [0x02: 1][random-challenge: 16]
```

The client encrypts the challenge using DES with a preset 8-character user-defined password, and sends it in a 16-bytes message back to the server:

```
C > [response: 16]
```

If the response is incorrect, the server notifies with an error message and closes the connection:

```
S > [0x01: 1][#err-msg-chars: 1][err-msg : #err-msg-chars]
```

Otherwise, the server responds with an `init` message, which encodes the available devices and controls in the house to be driven by the client.

2.3.2 Initialization

The initialization phase consists of a single server message, in a continuation of the handshake process. With a single server message, the client is notified about all the device types, numbers and states, which altogether comprise the “state of the house”. After the client receives the server `init` message, it should have all the information about what devices can be controlled.

One of the challenges for RSHC is how to efficiently encode device information. On one hand, most houses can be assumed to include basic devices that should be available for remote control, like lights, air-conditioning or security alarm; these devices can be encoded efficiently, as common information can be encoded into the protocol (i.e. assumed to be known in advance for both sides). On the other hand, customizable controls for uncommon devices are also desirable, such as the ability to control pool water temperature (under the assumption that smarthouses do not often have swimming pools).

In this document we lay out a solution in which several devices are predefined, along with their possible states and operations. These device types are encoded with increasing integers starting at 0. As discussed in Sec. 4, we leave possible future support in custom messages that can be defined by the house (server) for uncommon devices by simply follow the encoding of known device types and continue the numbering (e.g. for a version that supports 5 known device types, they are encoded as 0–4, and the first custom type will be assigned 5).

The first version of RSHC supports 5 known device types. Tab. 1 details these types, along with their numeric code, state and actions. Actions are followed by the device states in which they are legal (in parenthesis).

The server `init` message is then constructed starting with the `init` message type `0x03`, followed by the list of known device types in order (i.e. first lights, then shades etc.) Each device type starts with a byte indicating the number of such devices, followed by their 16-byte names and current states. The complete `init` message is then structured as follows:

Device Code	Type	States	Actions
`0x00`	Light	`[0x00:1]` – off	`[0x00:1]` – turn on (0)
		`[0x01:1]` – on	`[0x01:1]` – turn off (1)
			`[0x02:1][level:1]` – dim (1)
`0x01`	Shade	`[0x00:1]` – up	`[0x00:1]` – put down (0)
		`[0x01:1]` – down	`[0x01:1]` – pull up (1)
			`[0x02:1][level:1]` – dim (1)
`0x02`	AirCon	`[0x00:1]` – off	`[0x00:1]` – turn on (0)
		`[0x01:1]` – on	`[0x01:1]` – turn off (1)
			`[0x02:1][temp:1]` – set-temp (1)
`0x03`	TV	`[0x00:1]` – off	`[0x00:1]` – turn on (0)
		`[0x01:1]` – on	`[0x01:1]` – turn off (1)
			`[0x02:1][channel:1]` – set-channel (1)
			`[0x03:1][volume:1]` – set-volume (1)
`0x04`	Alarm	`[0x00:1]` – off	`[0x00:1]` – turn on (0,2)
		`[0x01:1]` – on	`[0x01:1]` – turn off (1,2)
		`[0x02:1]` – armed	`[0x02:1]` – arm (0,1)

Table 1: List of supported device types.

```
[0x03 : 1]
[n0=#type 0 devices: 1][name0: 16][state0: 1]...[name n0: 16][state n0: 1]
...
[n4=#type 1 devices: 1][name0: 16][state0: 1]...[name n4: 16][state n4: 1]
```

For instance, the following message indicates there are 2 lights – bedroom light turned off and kitchen light turned on, no shades, no AC, one TV named 'main TV' turned on, and no security alarm:

```
[0x03][2]['bedroom'][0]['kitchen'][1][0][0][1]['main tv'][1][0]
```

The server `init` message concludes the synchronous part of the RSHC communication. From this point on, the client sends requests to the server – queries or actions – and the server responds accordingly. Next we detail the client and server messages in the normal communication phase of the protocol.

2.3.3 Client to Server Messages

2.3.4 Server to Client Messages

3 DFA

This is the DFA section.

4 Extensibility

This is the extensibility section.

5 Security Implications

This is the security implications section.

References

[1] Wikipedia, “Protocol data unit,” 2013. [Online]. Available: http://en.wikipedia.org/wiki/Protocol_data_unit

A Sample Appendix