**WARP Protocol**

**Version: 1.1**

(Updated 13th September 2014 by Hoai Phuoc Truong)

The protocol below aims to support internal two way communication between WARP board and PC Engine over the Ethernet. Modification to the protocol should ensure modularity and extensibility to support future developments and additional features. Also, this design avoids fields with variable lengths for simple implementation in WARP.

**I) General architecture of WARP Protocol:**

A typical packet from PC Engine to WARP (and vice versa) will have the following:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ethernet | WARP | | | Encapsulated dot11 data  (if applicable) |
| WARP Header | | WARP Data |
| Type | Subtype | (See below) |

**II) WARP Header:**

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Subtype | Expected WARP Element | Expected data at the end |
| 0 (Ignore) (Default value) | Any | None | No |
| 1 (Transmit) | 0 | Transmit Element | Yes |
| 1 |
| 2 (Control) | Any | Control Element | No |

|  |  |  |
| --- | --- | --- |
| Type | Subtype | Meaning |
| 0 (Ignore) (Default value) | Any | Ignore (for testing purpose) |
| 1 (Transmit) | 0 | Management frame transmit |
| 1 | Data frame transmit |
| 2 (Control) | 0 | Ignore |
| 1 | Transmission control (see WARP Element Transmit below) |
| 2 | MAC address control (see WARP Element MAC Address Control below) |

**III) WARP Elements:**

**1) Transmit Elements:**

a) WARP Transmit Element:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Field |  | Dot11 TX parameters | | | | | | | |
| Dot11 PHY TX parameters | | | Dot11 MAC TX parameters | | Fragment info | | | |
| bssid | | Flag | Max # of reTX | Data length | ID | Fragment # | Total # of fragment | Byte offset |
| Size (byte) | 6 | | 1 | 1 | 2 | 1 | 1 | 1 | 2 |
| Default | 00:00:00:00:00:00 | | 0 | 0 | 0 | 255 | 0 | 0 (will be ignored if 0) | 0 |
| Note | WARP is not responsible to validate this | | WARP flag |  |  | \* | \*\* | | \*\*\* |

\* ID of 255 is reserved and is not used. We expect these fragments to arrive and get reassembled quickly so that the WARP can save its queue space for other packets (queue size is 3000 elements at the moment). If more than 255 fragments (more than 1 byte for ID) have to be saved on the WARP, the link between PC Engine and the WARP had a significant delay issue and therefore we can safely drop old packets.

\*\* Maximum size of a frame before fragmentation using this protocol would be 28 x 1500 (MTU of Ethernet protocol) = 348000 bytes. This is sufficiently large for any protocol underneath.

\*\*\* First byte represents most significant 8 bits of the offset. Second byte represents least significant 8 bits of the offset. Although this offset is not enough to cover 348000 bytes of the maximum frame in \*\*, it supports most protocols underneath 802.11 data frame.

**2) Control Elements:**

a) WARP MAC Address Control Element:

|  |  |  |
| --- | --- | --- |
| Field | WARP MAC Address Control Element | |
| Operation code | MAC Address |
| Size (byte) | 1 | 6 |
| Default | 0 | 00:00:00:00:00:00 |

|  |  |
| --- | --- |
| Operation code (PC Engine to WARP) | Meaning |
| NOTHING (0) | Do nothing |
| MAC\_ADD (1) | Add |
| MAC\_REMOVE (32) | Remove |
| MAC\_CHECK\_IF\_EXIST (64) | Check if exist (from PC Engine to WARP) |

|  |  |
| --- | --- |
| Operation code (WARP to PC Engine) | Meaning |
| MAC\_EXISTED (65) | Existing (reply from WARP to PC Engine) |
| MAC\_NOT\_EXISTED (66) | Not existed (reply from WARP to PC Engine) |

b) WARP Transmission Control Element:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field |  | WARP Transmission Control Element | | | | |
| bssid Address | Disabled | TxPower | Channel | Rate | HW mode |
| Size (byte) | 6 | 1 | 1 | 1 | 1 | 1 |
| Default | 00:00:00:00:00:00 | 0 | 0 | 2 | 1 | 0 |
| Note | \* |  |  |  |  | Not defined yet |

\* This bssid should have been added previously by the PC Engine to MAC low. WARP will verify that the bssid is in the database. If not, WARP simply ignores this packet.

PC Engine would send the control element to WARP. WARP will attempt to modify/ add the configuration to fit the requirement and then reply to the PC Engine using the same format as the control message received initially.