# Radio over Ethernet (RoE) base protocol

[///Editor’s Note: this clause will describe the native RoE encapsulation transport format. The following subclauses will also describe the overall RoE architecture, showing encapsulation and decapsulation function locations, and the mapper function locations. This clause also lists the underlying assumptions a RoE enabled architecture has.]

## Overview

## RoE Ethernet Type

## Bit and octet ordering, and numerical presentation

## RoE common frame format

### ver (version) field

### pkt\_type (packet type) field

### flow\_id (flow identifier) field

### Timestamp

The **timestamp** field is 32 bits in size and expresses the absolute time for presentation, relative to a defined reference plane, of the information within the packet at the receiving endpoint of the RoE packet. Both the transmitting and receiving endpoints must share the same understanding of the Time of Day (ToD) in order for the information to be presented at the desired time.

The format of the timestamp field is shown in Figure 2.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | … | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| timestamp (integer ns) | timestamp (fractional ns) |

Figure 2: Format of the timestamp field

The 30 most significant bits of the timestamp field count in units of nanoseconds and the value ranges from 0ns to 999,999,999ns (0x0 to 0x3B9AC9FF, respectively). The two least significant bits of the timestamp field count in units of 0.25ns and the value ranges from 0ns to 0.75ns (0x0 to 0x3, respectively). If sub-nanosecond timestamping is not used, these two bits shall be set to 0 at the transmitter and shall be ignored at the receiver.

The timestamp value is capable of expressing a presentation time of up to one second in the future. Both endpoints must account for the rollover condition of the timestamp field after 999,999,999.75ns. See Annex B for example algorithms that truncate the presentation time into the timestamp field at the transmitting endpoint and that recover the presentation time from the timestamp field at the receiving endpoint. Refer to subclauses 4.10.4, 4.11.4, 4.12.4, and 4.13.4 for more details on the usage of the timestamp and the presentation time.

1. Example algorithm for timestamp conversion

The following C-like pseudocode algorithm illustrates how to:

* convert an IEEE 1588v2-based presentation time into the 32 bit RoE timestamp value (presentation\_to\_timestamp)
* convert the 32 bit RoE timestamp value into an IEEE 1588v2-based (PTP-based) presentation time (timestamp\_to\_presentation)

The correct usage of the recovered presentation time from the 32 bit RoE timestamp value requires common agreement on the local time. The clocks of the RoE packet transmitter and receiver must be synchronized to a common reference.

In this example algorithm, the 2 bit fractional nanoseconds field is not used and is set to 00b by the RoE transmitter and is ignored by the RoE receiver and the presentation time is limited to a time that is less than 500,000,000ns after the timestamp was generated at the RoE packet transmitter.

typedef uint64\_t uint48\_t;

#define TSTAMPSZE 1000000000

#define WINDOWSZE  500000000

struct Timestamp {

    uint48\_t secondsField;

    uint32\_t nanosecondsField;

};

// Convert a PTP timestamp (the presentation time) to a 32 bit timestamp.

// The 2 fractional nanoseconds bits are forced to 00b.

uint32\_t presentation\_to\_timestamp( const struct Timestamp\* presPTP ) {

    return presPTP->nanosecondsField << 2;

}

// Convert a 32 bit timestamp to a PTP timestamp (the presentation time).

// The 2 fractional nanoseconds bits are ignored.

void timestamp\_to\_presentation( const struct Timestamp\* localPTP,

                                struct Timestamp\* presPTP,

                                uint32\_t ts ) {

    int32\_t diff;

    uint32\_t wrap;

    ts >>= 2;       // remove fractional nanoseconds

    diff = ts - localPTP->nanosecondsField;

    wrap = abs(diff) > WINDOWSZE ? 1 : 0;

    presPTP->secondsField = localPTP->secondsField;

    presPTP->nanosecondsField = ts;

    if (diff < 0) {

        presPTP->secondsField += wrap;

    } else {

        presPTP->secondsField -= wrap;

    }

}