IEEE 1588/802.1 AVB Design Meeting

TITLE: Initial Description of Possible Use of Sync and Followup Messages with Peer-to-Peer Transparent Clocks in AVB

Revision: 0.0

AUTHOR(S) Geoffrey M. Garner (gmgarner@comcast.net)
Samsung (Consultant)
Tel: +1 732 758 0335

DATE: 21 Feb 2006

1 Introduction

During the February 21, 2006 IEEE 1588/802.1 AVB Design meeting, there was initial discussion of a possible way that AVB could use Peer-to-Peer (P2P) Transparent Clocks (TCs) with Sync and Followup messages. It had been stated previously that the use of separate Sync and Followup messages is undesirable with the expected message rates in AVB because the resulting load on the processor would be too large (AVB is expected to use an inexpensive processor such as the 8051). In particular, the AVB processor may take as long as 10 ms to process a Followup message in a TC node. This means that for a path through 7 TCs, it would require on the order of 70 ms for the Followup message to travel from the master to the slave. Note that the Sync messages are time stamped at each TC node as they enter and exit, are not further processed; therefore, a Sync message will travel from the master to slave in much less than 70 ms (the time for the Sync to travel from the master to the slave will likely be on the order of the AVB latency requirement; values in the range of 2 – 6 ms have been discussed). If the master sends Sync every 10 ms, this means that each TC would have to maintain state information on the residence times for multiple Sync messages at any given time (the final TC in the chain would have to save information for as many as 7 TCs). As a second alternative, TCs could maintain state information for fewer TCs if Followup messages were processed faster. As a third alternative, the Sync interval could be increased to 70 ms or more. All three alternatives were considered to be undesirable.

A possible approach for using Sync and Followup messages in AVB was discussed. In this approach, a Sync message is held at a TC until the Followup message arrives. When the Followup message arrives, its correction field is added to the correction field of the Sync, and the Sync is sent. The time the Sync is sent is noted; using this and the time the Sync arrived, a residence time for the Sync in the current TC node is computed. A new Followup message is generated, and the residence time for the current TC is placed in the its correction field.

With this approach, a Sync message will not get ahead of a Followup message for a previous Sync as long as the time between Sync messages is not shorter than the time required for processing a Followup message at a node (i.e., 10 ms). When a Sync message is sent, at least 10 ms will have elapsed since the previous Sync message, which means that the Followup message will have had enough time to be processed at the next downstream node.

The purpose of this document is to provide an initial written description of this approach. The description has been generated based on the verbal discussion in the Design Meeting. It is expected that this written description will be modified after discussion and review.

2 Ordinary or Boundary Clock

1. The master clock (i.e., port in the master state) sends Sync on the respective ports with the followup flag set and the correction field initialized to zero, and measures the time of departure of Sync on each port
2. The master (i.e., port in the master state) sends Followup on each respective port with the correction field initialized to zero and the preciseOrigin timestamp equal to the measured time of departure of the Sync message on that port
3. Each slave (i.e., port in the slave state) measures the time of arrival of the Sync message
4. When the slave (i.e., port in the slave state) receives the Followup message corresponding to a Sync message, it adds the correction fields in the Sync and Followup messages to the preciseOrigin Timestamp in the Followup message). [**Author's Note: TC Working Technical Description Version 12 indicates that the port in the slave state also adds the result of the upstream path delay calculation. It is not clear which path delay calculation this is referring to. The path delay on the link from the slave port to TC is obtained using the regular Delay_Req/Delay_Resp mechanism, and appears as a separate term in the slave clock offset calculation in IEEE 1588, version 1, clause 7.8.1.1. The path delays on upstream links between successive TCs are calculated using the ADelay mechanism, and are added to the Followup message correction field and subsequently to the Sync message correction field at the following node (see item 3.3).]**

3 P2P Transparent Clock

Notes: (1) In AVB, all TCs are Followup TCs (i.e., on-the-fly TCs are assumed not to be used)
(2) Items 6, 8 and 9 below are taken from the current TC Working Technical Description (Version 12; 17 Feb 2006; a portion of item 9 referring to End-to-End (E2E) TCs has been omitted because AVB will not use E2E TCs

1. Measures the time of arrival of a Sync message
2. Holds the Sync message until the corresponding Followup message arrives
3. On arrival of the Followup message, adds the correction field of the Followup Message to the Sync message
4. Sends the Sync message on the respective master ports, and measures its departure time on each port
5. Computes the residence time for the Sync message on each port, and places it in the correction field of a respective new Followup message generated for each port
6. Adds the path delay on the path from which the Sync message came to the correction field of the followup message
7. Sends the new Followup message
8. Performs path delay measurements to peer nodes using the ADelay mechanism.
9. Responds to Delay_Req received from a boundary clock or ordinary clock with Delay_Resp, in the same manner that a boundary or ordinary clock would respond (i.e., the Delay_Resp contains the delay_receipt timestamp for the corresponding received Delay_Req). **[Author's Note: It appears that, just as a TC responds to Delay_Req received from the slave port of a downstream OC or BC, it must send Delay_Req to the master port of an attached upstream OC or BC. It would then use the Delay_Resp message returned by the master OC or BC to computed the path delay on the link to the master. This path delay would be added to the correction field of the Followup Message that is generated.]**