Higher Layer Synchronization in an ECMA-368 Ultra Wideband Network

Margaret U. Anyaegbu[1], Alexander Weir[2], and Cheng-Xiang Wang[1]
[1]Joint Research Institute for Signal and Image Processing, School of EPS, Heriot-Watt University, Edinburgh EH14 4AS, UK
[2]TES Electronic Solutions, Edinburgh EH28 8LP, UK
Email: mua4@hw.ac.uk, alexander.weir@talk21.com, cheng-xiang.wang@hw.ac.uk

***Abstract*-An algorithm for implementing higher layer synchronization in ECMA-368 networks is being developed within the framework of the EUWB research and development project. In this paper, we adapt three synchronization algorithms used in wireless sensor networks to suit an Ultra Wideband (UWB) network and evaluate their performance in order to determine the best algorithm for a video streaming application scenario. We also propose some extensions to the ECMA-368 standard that would facilitate the implementation of these algorithms. For these purposes, we have simulated a UWB beacon group in OPNET. We observe that the algorithms that correct for clock drift are better suited for implementation in a UWB network.**

I. INTRODUCTION

Short-range, high-speed wireless connectivity is an emerging application in various commercial sectors (including personal computing, consumer electronics, and mobile communications), especially for real-time multimedia applications such as video streaming. In recent years, Ultra Wideband (UWB) has generated huge interest as the technology of choice for such applications due to its potential for enabling high data rates, advanced Quality of Service (QoS), and low cost systems (due to low complexity) [1]. The high data rate ECMA-368 UWB standard [2], developed by the WIMEDIA industrial consortium, is based on a Multi-Band Orthogonal Frequency-Division Multiplexing (MB-OFDM) physical (PHY) layer and is capable of supporting data rates up to 480 Mbps for a distance of up to 3 m. The maximum transmission range is 10 m, but this is only typically achievable at the lowest data rate of 53.3 Mbps and usually in an ideal operating environment.

Real-time video streaming is an important application for the EUWB research project [3], in which two scenarios are being considered: wireless aircraft cabin networks and wireless home entertainment systems. The generic network model for these scenarios consists of a video server streaming packets to multiple end devices over Ethernet and UWB links, via switches and access points, as illustrated in Fig. 1.

When video packets are transmitted in this way, jitter inevitably occurs as a result of variation in end-to-end delays experienced by consecutive packets from the same stream. This variation, typically caused by factors such as lack of transmission opportunity, prioritisation of different streams, packet retransmission or the lack of guaranteed bandwidth in wireless networks, reduces the QoS of the application at the receiver. A jitter buffer can be used to minimise the effects of jitter; however large delay variations require a larger, more expensive buffer which ultimately results in the introduction of a larger amount of constant latency at the start of the application that is unsuitable for real-time streaming.

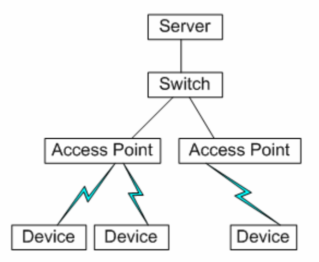

Figure 1. Generic network topology.

Synchronization of the application (higher layer) clocks at the sender and receiver can reduce the amount of jitter, thus relaxing the size of the jitter buffer and ultimately improving the QoS for real-time applications. Furthermore, when streaming applications are multicast to co-located devices (e.g. multiple loudspeakers in a home entertainment or wireless cabin network), there is an additional QoS requirement for synchronized media playback in the entire system. Higher layer clock synchronization is the only way to achieve this, as it can take into account the variation in end-to-end packet delays experienced at the different devices.

The ECMA-368 standard defines an optional higher layer synchronization mechanism that uses the services provided by the Medium Access Control (MAC) layer to accurately synchronize timers located in different devices. It involves the multicasting of synchronization frames, containing timing information, to devices requiring synchronization. In addition the MAC Layer Management Entity (MLME) is required to alert the higher layer whenever such frames are transmitted or received in order to achieve synchronization.

Although the synchronization process is described in [2], the actual implementation of the mechanism is beyond the scope of the standard. To the best of the authors' knowledge, no higher layer synchronization algorithms have been proposed in the literature for ECMA-368-compliant UWB systems.

There are two key requirements of the synchronization algorithm for the wireless cabin and home entertainment application scenarios. The algorithm must be scalable, such that its application can be extended from a simple beacon group of devices to the hybrid wired/ wireless network illustrated in Fig. 1. Secondly, the synchronization accuracy must not exceed 100 μsec across the entire network. In this paper, we consider the case of a UWB beacon group.

The rest of the paper is organized as follows. We briefly describe the key features of the ECMA-368 PHY and MAC layers in Section II. The synchronization algorithms and the OPNET simulation model are presented in Sections III and IV, respectively. Simulation results are discussed in Section V and the paper concludes in Section VI.

II. AN ECMA-368-COMPLIANT UWB SYSTEM

The 3.1 – 10.6 GHz UWB frequency spectrum is divided into 14 bands, each 528 MHz wide. This band spacing ensures that each OFDM symbol fits perfectly within a band. Three mandatory data rates (53.3, 106.7, and 200 Mbps) and five optional data rates (80, 160, 320, 400, and 480 Mbps) are specified in the ECMA-368 standard. Time and/ or frequency spreading techniques combined with convolutional Forward Error Correction (FEC) coding are used to vary the data rates and provide a robust signal at low transmitter power levels.

The MAC layer has a fully distributed architecture that supports ad-hoc, peer-to-peer networking. The channel time is divided into periodic intervals, called superframes. Each superframe is further divided into 256 Medium Access Slots (MAS), each of 256 μsec duration. The superframe is also split into a beacon period (BP) in which beacon frames are transmitted for the purpose of establishing and maintaining the network, and a data transfer period. During the data transfer period, devices may transmit their data, using either the Distributed Reservation Protocol (DRP) or the Prioritized Contention Access (PCA) mechanism. The start of a superframe is the Beacon Period Start Time (BPST) and the set of devices that share the same BPST and exchange beacon frames during the BP form a beacon group.

III. CLOCK SYNCHRONIZATION

A. *Definition of Terms*

A physical clock is a hardware device that periodically counts the oscillations of a crystal or quartz. After a specified number of oscillations, the clock register is incremented by one clock-tick, to represent the passing of time. A software clock or virtual clock is simply a transformation of the hardware clock. Hardware clocks do not oscillate at the same frequency over time. The rate at which the frequency changes is termed clock drift, and over time it causes the clock reading to gradually diverge from the "true" time reported by a standard reference time source such as Coordinated Universal Time (UTC). Clock offset is the instantaneous difference between the clock's reading and true time while skew refers to the frequency difference between the clock and true time [4]. If a clock is compared with another clock rather than with a standard time source, these terms become relative offset and relative skew, respectively. Two clocks are said to be synchronized if their relative offset is zero.

B. *Classification of Synchronization Algorithms*

Synchronization algorithms can be classified in various ways, including master-slave versus peer-to-peer, clock correction versus free-running clocks, internal versus external synchronization, sender-to-receiver versus receiver-to-receiver, and probabilistic versus deterministic algorithms [5].

In master-slave protocols, slaves synchronize to the clock of a designated master node, while peer-to-peer protocols allow a node to synchronize to any other node in the network. Clock correction protocols adjust the clocks after each synchronization round whereas in free-running clock protocols each node simply maintains a table of relative offsets that relates its local clock to other clocks on the network. Unlike internal protocols, external protocols have access to a trusted time reference source such as UTC. Sender-to-receiver synchronization is the conventional method in which a receiver achieves synchronization with a sender based on the timing information it receives from it, while receiver-to-receiver protocols perform synchronization between receivers by comparing the time at which they receive the same message. Lastly, probabilistic protocols provide a guarantee on the maximum clock offset permitted in the network, together with a failure probability while deterministic techniques guarantee an upper bound on the clock offset with certainty.

Due to the hierarchical nature of the generic network illustrated in Fig. 1 and the fact that direct communication between devices is not required, a master-slave synchronization approach with clock correction would be necessary to ensure that all devices on the network are synchronized. For the same reasons a sender-to-receiver protocol is implied, although a receiver-to-receiver protocol can be implemented provided that the time comparison occurs between the sender and receiver, as is the case of the Continuous Clock Synchronization protocol developed in [6]. Furthermore, the algorithm will support internal synchronization and use a deterministic approach to guarantee the accuracy of the system.

C. *Description of Three Selected Algorithms*

Based on the above reasoning, the Delay Measurement Time Synchronization [7] and Continuous Clock Synchronization [6] algorithms were selected. We further consider a Linear Rate Synchronization algorithm based on the notion of a virtual clock introduced in [6].

The Delay Measurement Time Synchronization algorithm combines a master's timestamp with delay measurements, in order to achieve synchronization of the slaves. Each slave takes two timestamps: one when it receives the preamble of the synchronization message, and the other after the message has been processed. The difference between these timestamps is a measure of the data transfer time plus the processing delay. Each slave also estimates the time taken to transmit the message preamble. Finally, each slave sets its clock to the sum of the master's timestamp, the data transfer time plus processing delay, and the preamble transmission time.

The Continuous Clock Synchronization algorithm, described in [6], uses an advanced rate-adjustment algorithm to spread clock correction over a finite interval in order to prevent time discontinuity caused by instantaneous clock correction. In this way, the local clock time is corrected by gradually increasing or decreasing the clock rate. Each node in the network has both a physical and virtual clock. In this sense, a virtual clock is simply a transformation function that corrects the skew rate of a physical clock. For the sake of simplicity, a linear function is used and clock correction is achieved by changing the parameters of this function after every synchronization round so that the slave's virtual clock matches the master's physical clock as closely as possible. The algorithm assumes that message reception is tight, such that a master node receives its own broadcast message at approximately the same time as the slave nodes [8]. Each synchronization frame contains the time at which the master node received the last synchronization message sent and by comparing this value with its own reception time for the same message, a slave is able to achieve synchronization with the master.

The UWB PCA and DRP data transmission mechanisms do not permit any node to receive its own broadcast frame, hence adapting the Continuous Clock algorithm to a UWB network requires a calculation of the expected reception time at the master node. We estimate the processing delay between the synchronization frame being scheduled by the MAC and the first symbol being received on-air by the remote device and add this to the master's transmission timestamp in order to obtain this value.

Another possible implementation of synchronization is based on a linear-rate function which relates a slave's clock to the master's clock. The master sends its timestamp to the slaves in the form of a command frame every synchronization round and each slave records its local clock time when it receives the synchronization frame, thus building a table of pair values in the form $[M_n\ S_n]$, where M_n is the local clock time at which the master sent the nth synchronization frame and S_n is the local time at which the slave received the same frame. After two rounds the slave can calculate the relative skew ξ and relative offset σ as follows:

$$\xi = (M_n - M_{n-1}) / (S_n - S_{n-1}). \quad (1)$$

$$\sigma = M_n - S_n\xi. \quad (2)$$

These values are then used to update the parameters of its virtual clock V which transforms the hardware clock H to mimic the behaviour of the master's clock as in (3).

$$V = \xi H + \sigma. \quad (3)$$

D. Implementation of Higher Layer Synchronization

The ECMA-368 standard includes three MLME primitives to support the higher layer synchronization function. The *mlme_hl_sync.request* primitive is sent from the higher layer to the MAC to initiate the synchronization mechanism, the *mlme_hl_sync.confirm* returns the result of the request, while the *mlme_hl_sync.indication* primitive indicates the transmission or reception of a sync frame.

Since the goal is to synchronize the clocks of higher layer protocols which may reside separately from the MAC, it makes sense for the MAC to be responsible for implementing the synchronization algorithm. The MAC layer can directly access the PHY layer clock for the purpose of generating timestamps; however since the higher layer clock values are required as parameters in the synchronization algorithm, there is a requirement for mapping higher layer clock values to PHY layer clock values. Furthermore, after synchronization is complete there is an additional requirement for transferring the clock adjustment to the higher layer. These requirements form the basis for our proposed modifications to the ECMA-368 standard.

The means of fulfilling the first requirement depends on whether the MAC and higher layer protocol reside on the same entity or on separate entities. If the latter case holds, then the higher layer clock values can be transferred to the MAC layer using the timestamp field of an empty Real Time Protocol (RTP) packet. In this case, the generation of the RTP packet will be triggered by the receipt of the *mlme_hl_sync.confirm* primitive at the higher layer. However if the MAC and higher layer protocol are on the same entity, then the MAC can directly access the higher layer clock value by simply reading the corresponding register. In either case the MAC layer reads the PHY layer clock value immediately it obtains the higher layer clock and then calculates the relative offset between the higher layer and PHY layer clocks. The MAC subsequently uses this offset to transform a PHY layer clock reading to the corresponding higher layer value. We also propose that the receipt of an *mlme_hl_sync.indication* primitive also triggers the transfer of the higher layer clock to the MAC via an RTP packet. On receipt of this, the MAC again reads the PHY clock and subsequently updates the relative offset. This is particularly important when a synchronization frame is received so that the slave node can transform its PHY layer reception timestamp value before clock adjustment computations are done.

Finally, we propose a *mlme_hl_clock_update.indication* primitive for transferring calculated clock adjustments to the higher layer. This primitive requires two parameters: offset and skew, for updating the parameters of the virtual clock. However the skew parameter is set to the default value of unity for algorithms such as the Delay Measurement Time algorithm that do not correct for drift.

IV. NETWORK SIMULATION IN OPNET

An OPNET [9] simulation model of the ECMA-368 standard has been developed for the purpose of research and development of UWB algorithms. Each ECMA-368 node consists of source and sink modules responsible for generating and consuming packets, respectively, a MAC Interface module for performing address resolution, and a simplified ECMA-368 MAC module. Each node also contains a higher layer clock and a PHY layer clock, and both are based on the drifting clock model presented in [10].

The system is assumed to be operating in an active quiescent state at the start of each simulation, i.e. operating channels have been selected, the nodes are initially synchronized, and a beacon group has been formed. Furthermore, hard DRP reservations have already been negotiated and the nodes are ready to send and receive data over these reservations.

V. Simulation Results and Discussions

A simple two node scenario (AP and device) was simulated. The AP acts as the master while the device is the slave, with clock drift rates of 0ppm and -20ppm respectively. The synchronization interval is 2.5 sec and the target accuracy is 100 µsec. The simulation was run for a period of 20 sec and the results are provided in Fig. 2.

As expected, without synchronization the relative clock offset between master and slave accumulates linearly, at a rate equal to the difference in their clock drifts. After each synchronization round, the Delay Measurement Time algorithm reduces the clock offset to zero but between rounds the offset starts accumulating again. Since the Continuous Clock and Linear Rate algorithms correct for clock drift, they perform better than the Delay Measurement Time algorithm and are able to maintain synchronization within the target accuracy for longer periods between synchronization rounds.

To identify the best algorithm for implementation, we consider the effects of packet loss and temperature variation. Packet loss results in missed synchronization rounds while the latter may change the rate of clock drift. In algorithms that do not correct for clock drift, the effect of packet loss is similar to the case of no synchronization as the clock offset will accumulate linearly, beyond the target accuracy level. Such algorithms would also be least resistant to large changes in clock drift because they would no longer be able to achieve the target accuracy using the same synchronization interval. Hence it is clear that the Linear Rate and Continuous Clock algorithms are better suited for UWB than the Delay Measurement Time algorithm.

The Continuous Clock algorithm has a longer convergence time compared to the Linear Rate algorithm, since its clock correction procedure is gradual rather than instantaneous. However once it achieves convergence, it yields a slightly more accurate synchronization since it incorporates an estimate of the processing delay.

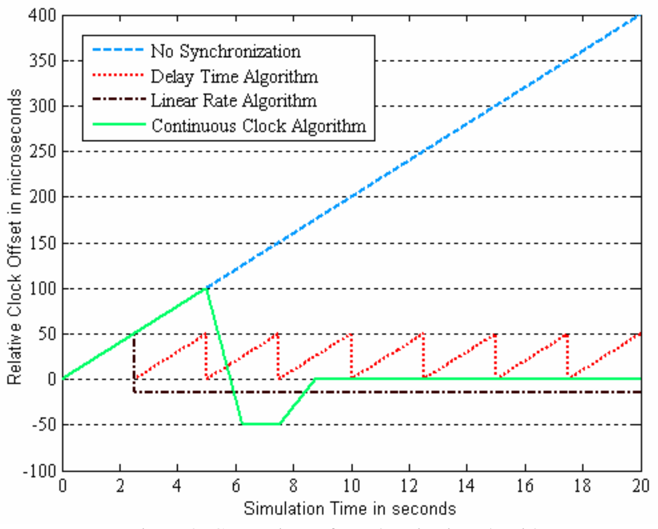

Figure 2. Comparison of synchronization algorithms.

VI. Conclusions

In this paper, we have proposed some extensions to the existing ECMA-368 standard to facilitate the implementation of the higher layer synchronization mechanism and demonstrated the performance of three candidate algorithms through network simulation. Our results show that drift-correcting algorithms perform better as they are able to maintain synchronization within the target accuracy for longer periods between synchronization rounds.

In the next phase of the project, the algorithms would be implemented on a development platform.

Acknowledgment

The authors gratefully acknowledge the EUWB consortium, EPSRC, and the Industrial Doctorate Centre in Optics and Photonics Technologies at Heriot-Watt University for funding this research. We also thank OPNET for providing the network simulation software free of charge under the University License agreement.

References

[1] M. Ghavami, L. B. Michael, and R. Kohno, *Ultra Wideband Signals and Systems in Communication Engineering (Second Edition).* West Sussex, UK: John Wiley & Sons, 2007.
[2] ECMA-368 Standard: High Rate Ultra Wideband PHY and MAC Standard, 2nd ed., Dec 2007.
[3] S. Zeisberg and V. Schreiber, "EUWB - Coexisting Short Radio by Advanced Ultra-Wideband Radio Technology," *Proc. ICT-Mobile Summit 2008,* Stockholm, Sweden, 2008.
[4] V. Paxson, G. Almes, J. Mahdavi, and M. Mathis, "Framework for IP Performance Metrics", RFC 2330, The Internet Society, May 1998.
[5] B. Sundararaman, U. Buy, and A. D. Kshemkalyani, "Clock Synchronization for Wireless Sensor Networks: A Survey", *Ad Hoc Networks*, vol. 3, no. 3, pp. 281–323, May 2005.
[6] M. Mock, R. Frings, E. Nett, and S. Trikaliotis, "Continuous Clock Synchronization in Wireless Real-Time Applications," *Proc. SRDS 2000,* Nürnberg, Germany, Oct. 2000, pp. 125.
[7] S. Ping, "Delay Measurement Time Synchronization for Wireless Sensor Networks," *Intel Research, IRB-TR-03-013*, June 2003,
[8] M. Mock, R. Frings, E. Nett, and S. Trikaliotis, "Clock Synchronization for Wireless Local Area Networks," *Proc. Euromicro-RTS 2000,* Stockholm, Sweden, June 2000, pp. 183.
[9] OPNET Product Documentation 15.0, www.opnet.com.
[10] Y. Quan and G. Liu, "Drifting Clock Model for Network Simulation in Time Synchronization," *Proc. 3rd International Conference on Innovative Computing Information and Control,* Liaoning, China, June 2008, pp. 385.