Performance Evaluation of Time Synchronization and Clock Drift Compensation in Wireless Personal Area Networks

Jonas Wåhslén Ibrahim Orhan Dennis Sturm Thomas Lindh
School of Technology and Health
KTH
Email:{jonas.wahslen; ibrahim.orhan; dennis.sturm; thomas.lindh}@sth.kth.se

ABSTRACT
Efficient algorithms for time synchronization, including compensation for clock drift, are essential in order to obtain reliable fusion of data samples from multiple wireless sensor nodes. This paper evaluates the performance of algorithms based on three different approaches; one that synchronizes the local clocks on the sensor nodes, and a second that uses a single clock on the receiving node (e.g. a mobile phone), and a third that uses broadcast messages. The performances of the synchronization algorithms are evaluated in wireless personal area networks, especially Bluetooth piconets and ZigBee/IEEE 802.15.4 networks. A new approach for compensation of clock drift and a realtime implementation of single node synchronization from the mobile phone are presented and tested. Finally, applications of data fusion and time synchronization are shown in two different use cases; a kayaking sports case, and monitoring of heart and respiration of prematurely born infants.

Keywords
Time synchronization, clock drift, data fusion, Bluetooth, IEEE 802.15.4.

1. INTRODUCTION
Correctly performed data fusion is crucial for applications in sports, medicine, health and many other fields. Samples of data from multiple sensors are received by a coordinating node such as a mobile phone. Data fusion requires that samples from multiple sensors have a time reference that is synchronized and comparable for all connected nodes. However, even small differences in clock frequency among the nodes may lead to unacceptable errors after some time. Compensation for clock drift is therefore a necessary component in time synchronization. Today, the main radio technologies for communication between wireless sensors and coordinating nodes are Bluetooth and IEEE 802.15.4. In this paper we evaluate and compare the performance of different time synchronization algorithms in Bluetooth piconets as well as ZigBee/IEEE 802.15.4 networks. In addition, we have developed and tested a new technique to compensate for clock drift to preserve accuracy in data fusion.

The paper is organized in the following way. Section 2 covers different methods for time synchronization and clock drift compensation. Performance evaluation results from Bluetooth piconets and IEEE 802.15.4 networks are shown in Section 3. Section 4 discusses applications in a kayaking sports use case and a neonatal care use case.

2. ALGORITHMS
This section briefly describes the algorithms for synchronization (Section 2.1) and clock drift compensation (Section 2.2) that are evaluated in Section 3.

2.1 Synchronization Algorithms
Two different approaches for time synchronization in wireless sensor networks are described in this section; firstly to coordinate the local clocks in each of the sensor nodes, and secondly to use the clock of a single central node, e.g. a mobile phone, to synchronize data. A comprehensive survey of time synchronization in wireless sensor networks can be found in [1].

2.1.1 Synchronizing the Clocks in Sensor Nodes
Algorithms for synchronization of local clocks in wireless sensor networks often apply a combination of methods for distributed systems [2] and the Network Time Protocol [3] for the Internet. The algorithm used in the performance evaluation in Section 3 is described in detail in [4]. The main idea is to synchronize the sensor nodes' (slaves') clocks to a mobile phone (master) clock in a Bluetooth piconet. Fig. 1 illustrates the algorithm: The master stores a timestamp, t_1, and sends a synchronization request message, $m_{request}$, to a slave that hosts a sensor. After receiving $m_{request}$, the sensor node inserts its local time, t_2, into the return message, m_{reply}, before transmitting it back to the master. The master generates timestamp t_3 when m_{reply} is received. The master can, based on these timestamps, determine the offset between its own clock and the slave's clock, and accordingly synchronize the two clocks. When the master during consecutive data acquisition receives a sample from the slave, including its local timestamp, it adds the estimated offset to obtain the common synchronized time. The master performs all offset calculations and implements the synchronization, slaves are passive and simply respond to requests.

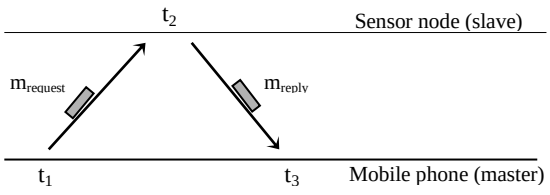

Fig. 1. The figure shows a synchronization message sent between a mobile phone and a sensor node, and the timestamps set by the master and slave.

Timestamps t_1 and t_3 are set by the master and timestamp t_2 is set the by slave. Let $t_{round}=t_3-t_1$be the estimated round-trip time and D_{min} the minimum delay time for $m_{request}$ from the master to the slave. The earliest time $m_{request}$ can reach the slave is t_2+D_{min}, and the latest time $m_{request}$ can reach the slave is $t_2+t_{round}-D_{min}$. The

maximum error in estimating the time for $m_{request}$ to reach the slave is $\pm(t_{round}/2\text{-}D_{min})$. The original idea in Cristian's algorithm [5] for probabilistic clock synchronization in distributed systems is that a server responds with its time, t_{server}, as a reply to a request from a client. The client sets its clock to $t_{server}+t_{round}/2$. Choosing $t_{round}/2$ as the delay between the readings of t_1 and t_2 minimizes the maximum error.

Instead of letting the slave act as a client and request the master's time, compute the offset and set its clock, the master performs all calculation and implements the synchronization. The estimated offset, *o*, between the clocks is the difference between t_1 and t_2 plus the delay between the readings of the clocks (half of the round-trip time), as shown in Fig. 2. The master adds this offset to the local timestamp inserted by the sensor node along with the data samples.

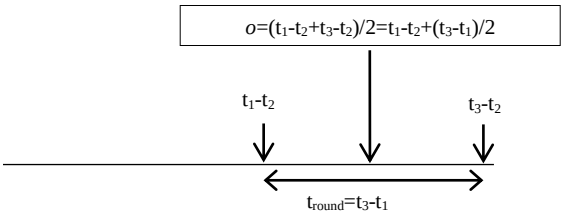

Fig. 2. The figure shows how the offset of the slave's clock relative to the master's clock and the round-trip time is estimated. The error limit for the offset is $\pm(t_{round}/2\text{-}D_{min})$.

An alternative method to coordinate the local clocks in sensor nodes is to use broadcast messages. This approach is further developed and evaluated in Section 3.2.

2.1.2 Single Clock Synchronization

Instead of coordinating the clocks in the sensor nodes, an alternative approach is to rely on the single clock in the receiving node, such as a mobile phone ([1], [6] and [7]). Fig. 3 shows an application in a mobile phone that reads samples from wireless nodes.

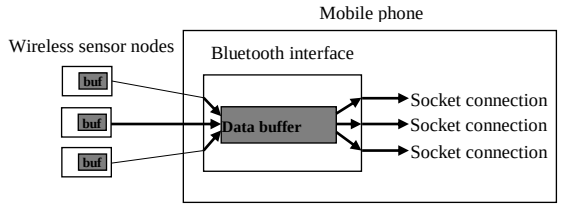

Fig. 3. An application in a mobile phone reads samples from three sensor nodes. Data from the sensors pass two buffers; the outgoing data buffer on the sensor, and the incoming data buffer on the mobile phone.

The accuracy of data synchronization based on timestamps by the mobile phone (Fig. 3) depends on a variable time period, Δt, between the actual sampling time on the sensor node and the time the sample is read from the buffer on the mobile phone. Let t_{mobile} be the time set by the application program on the mobile phone, and t_{sensor} be the time when the sample was acquired by the sensor node. If the clocks on the sensor node and the mobile phone are synchronized in time, then Δt for sample k will be $\Delta t(k)= t_{mobile}(k)-t_{sensor}(k)$. The main idea is to identify samples with the smallest Δt, and use the result to recalculate the timestamps for all samples. The method estimates the minimum Δt that consists of the fixed part of the delay i.e. processing time, sending time, propagation time etc. An algorithm that identifies samples with minimum Δt has been implemented [7]. It uses the observation that the last sample in the incoming queue has spent the shortest time in a buffer. The algorithm used in the performance evaluation in Section 3 can be summarized in the following steps for each connected sensor.

i) Let the samples be timestamped when read by the application program from the incoming buffer on the mobile phone.

ii) Identify a set of samples that is likely to have a minimum Δt.

iii) Calculate the average offset from this set of samples to correct the timestamps of the remaining samples.

The original algorithm in [7], which applied linear regression, has been improved to satisfy real-time requirements.

2.1.3 Clock Synchronization using Broadcast

An algorithm inspired by the Flooding Time Synchronization Protocol [8] for broadcast synchronization was implemented as following.

i) The coordinating node sends a broadcast message to all sensor nodes.

ii) A sensor node stores its local time when receiving the broadcast message.

iii) The synchronized time can be obtained, by just subtracting the sensor actual time with the time stored in the variable. All slave sensors will have the same time.

2.2 Clock Drift Compensation

The relation between two nodes local time, $t_{sensor1}$ and $t_{sensor2}$ can be modelled as a linear equation defined as

$$t_{sensor1} = o + dt_{sensor2} \qquad (1)$$

where *o* is the clock offset and *d* is clock drift. This clock drift is hardware-related and mainly caused by inaccuracy in the clock-crystal frequency. Small variations can occur due to for example power levels, temperature or voltage changes. A straightforward way to estimate the drift parameter *d* in Equation 1 is to retrieve timestamps from two sensors at two different time points (separated by a sufficient time interval) or using linear regression as in [7] and [8]. If multiple values are retrieved linear regression can be used. Resynchronization or an algorithm to compensate the drift may then be applied. Drift compensation can be implemented either by multiplying the local node by the estimated drift parameter, or by continuously adjusting the node's local time when necessary.

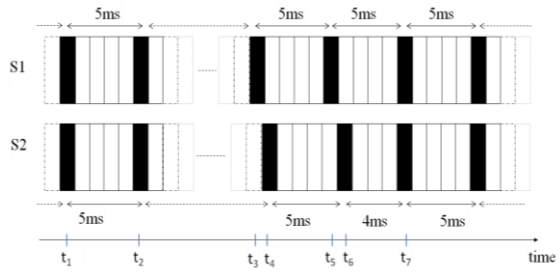

Fig. 4. The figure shows two sensors sampling at 200Hz at the same time (t_1, t_2). Due to clock drift the sampling rate is shifted (t_3, t_4 and t_5, t_6). The drift compensation function decreases the inter-sample time with 1ms for S_2 at t_6 when it detects a drift of at least 1ms. At t_7 the sensors sample at the same time again.

Clock drift compensation can be achieved by defining a virtual clock that runs with the estimated drift parameter *d* taken into account. We assume a resolution of 1ms for the local and the virtual clocks in the following. The sensor node clock with slowest clock (the lowest frequency) is used as a reference for the

other sensor nodes drift compensation. When the node detects that the virtual clock differs from the local clock more than 1ms, the virtual clock time is subtracted by 1ms.

Fig. 4 shows an example of how the drift compensation algorithm works. At t_1 and t_2 both sensor nodes, S_1 and S_2, are reading samples at exactly the same (synchronized) time with the sampling frequency f_s=200Hz (T_s=5ms). At t_3 the sample time for S_1 and S_2 are separated and partly overlapped at t_4. The time difference between the two local clocks at t_5 and t_6 has reached 1ms. The algorithm will now decrease the inter-sampling time T_s for S_2 by 1ms. The result is that the two sensors pick samples simultaneously at t_7. The algorithm will continuously monitor and control the clock drift.

3. PERFORMANCE EVALUATION RESULTS

This section presents results of performance evaluation of the algorithms outlined in the previous section. The testbed is described in Section 3.1 and results from tests in Bluetooth piconets and ZigBee/IEEE802.15.4 networks are shown in Section 3.2 and Section 3.3. Finally the algorithm for clock compensation is evaluated in Section 3.4.

3.1 Experimental Setup

The results presented in this section are obtained from a common testbed with a coordinator (master) and one up to six sensor nodes (slaves) from Shimmer Research [9]. These nodes are programmed in nesC (TinyOS 2.1.0 operating system) and are able to switch between a Bluetooth radio transceiver and a ZigBee/IEEE 802.15.4 radio transceiver.

3.2 Time Synchronization in Bluetooth Piconets

In this section we evaluate the performance of the two algorithms for synchronizing sensor nodes to a mobile phone (master) in Bluetooth piconets described in Section 2.1.

3.2.1 Synchronizing the Clocks in Sensor Nodes

The algorithm is evaluated by varying two parameters. Firstly, the number of connected sensor nodes, and secondly, the number of repeated loops of $m_{request}$ and m_{reply} (in Fig. 1), where each loop results in a set of timestamp t_1, t_2 and t_3. Two different cases are analysed regarding the way the sensor nodes are connected. Fig. 5 shows the synchronization error when nodes are simultaneously connected to the master during the synchronization procedure.

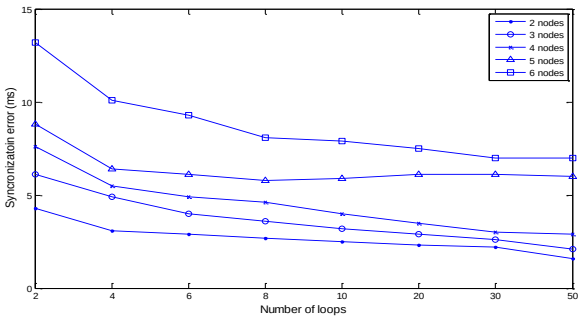

Fig. 5. Synchronization error in milliseconds when all nodes in a Bluetooth piconet are connected at the same time. The error decreases as the number of loops increases. The error increases when several nodes are connected.

In Fig. 6, each sensor node is synchronized to the mobile phone one at a time. The figures show the 99th percentile of the synchronization error in milliseconds for 1 up to 50 loops, where each test run is repeated a large number of times. From Fig. 5 and Fig. 6 it is obvious that the synchronization error decreases as the number of loops increases. No substantial improvement is gained by repeating the procedure more than 50 times, which takes approximately 1 second. It can often be interrupted after fewer loops. It is also evident that the error increases as the number of connected nodes increases. Furthermore, a substantially higher synchronization error can be observed when all nodes are connected at the same time (Fig. 5).

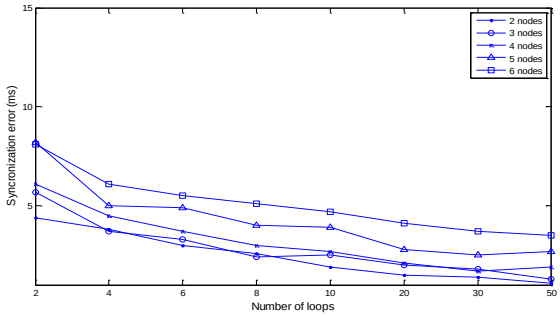

Fig. 6. Synchronization error in milliseconds when one node at a time is connected in a Bluetooth piconet. The synchronization error is approximately one half compared to Fig. 5, when the nodes all nodes are connected during the synchronization process.

This is a direct effect of the way Bluetooth controls multiple connections, where the master determines when a slave is permitted to transmit data by sending a null or pull packet to the slave. When a single sensor node is connected, the master only needs to check if that node has data to send. However, when multiple nodes are connected, the master will schedule the order that the nodes are permitted to transmit. A sensor node may have to wait before sending its sampled data. One reason is that the master visits every connected sensor (sends POLL or NULL packets) regardless of whether they have data to send or not. Another reason is that the scheduling is necessarily done in a round-robin fashion [4]. The additional error when all sensors are simultaneously connected during the synchronization procedure (Fig.5) is approximately twice the error when the sensors are synchronized one at a time (Fig. 6). For example, when 6 nodes are connected the time interval between every poll request from the master to a slave will be approximately 7ms.

3.2.2 Single Clock Synchronization from the Mobile Phone

The method described in Section 2.1.2 relies solely on the clock in the mobile phone for data synchronization and therefore does not require coordination of the sensor nodes' clocks. The synchronization error after 50 loops ranges from 1.1ms to 3.2ms (99th percentiles) depending on the numbers of nodes connected. This can be compared to 7ms error if all six nodes are connected simultaneously (Fig. 5) and 3.5ms error when the six nodes are connected one at a time (Fig. 6). The tests clearly show that the single clock synchronization approach (Section 2.1.2) results in significantly higher accuracy compared to synchronizing the clocks in multiple sensor nodes that are connected simultaneously. This is due to the fact that the response message m_{reply} (Fig. 1) with t_2 is blocked in the slaves' outgoing buffer until a later timeslot assigned by the master. The single clock synchronization method is of special interest when the sensor nodes' clocks are not accessible and if

resynchronization is needed, e.g. due to clock drift. In the original algorithm [7], linear regression based on the complete set of samples is used. This study shows that subsets of the samples can feed the algorithm, which decreases the processing time considerably.

3.3 Time Synchronization in ZigBee/IEEE 802.15.4 Networks

In this section we present the result of time synchronization in IEEE 802.15.4 networks based on the method in Section 2.1.1 and an alternative method using broadcast messages.

3.3.1 Synchronizing the Sensor Nodes' Clocks

Fig. 7 shows the synchronization errors using the same test setup as in Section 3.2 with the exception that the radio standard is IEEE 802.15.4, configured for contention-based access (CSMA/CA), instead of Bluetooth.

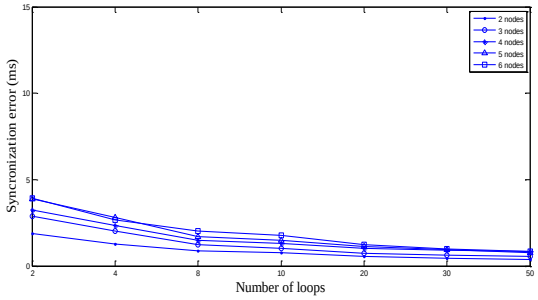

Fig. 7. The synchronization error in milliseconds for 1 to 6 participating sensor nodes and increasing number of loops in the synchronization procedure.

Clock synchronization according to the method in Section 2.1.1 gives considerably lower errors in IEEE 802.15.4 radio communication than in Bluetooth piconets after a few synchronization loops. Contrast to Bluetooth, contention-based access without a central scheduler means that a node is not forced to wait for a master poll request before transmitting the m_{reply} (Fig. 1). Applying CSMA/CA node merely checks if the media are idle prior to transmitting. The tiny synchronization messages will not result in major waiting times due to other nodes occupying the wireless channel. After 50 synchronization loops the error is approximately below 1ms even though all six nodes are active.

3.3.2 Broadcast Message Synchronization

The IEEE 802.15.4 radio standard supports broadcast messages multiple sensor nodes at the same time, which may be a feasible method for time synchronization. Our repeated tests have validated that measuring the time synchronization error using broadcast messages gives the same accuracy as using a hardware reset. Even though the coordinating node may have to wait for the wireless channel to become idle, the sensor nodes will receive the broadcast message whenever it is sent. Our tests show that every sensor node will have the same synchronized time with a resolution of at least 1 millisecond. The coordinating node's clock can also be included by sending a broadcast message from one of the sensor nodes. Since a broadcast solution provides accurate time synchronization and is straightforward to implement there is no need for the methods described in Section 2.1 in ZigBee/IEEE 802.15.4 networks.

3.4 Results of Clock Drift Compensation

The clock drift has been measured and the algorithm for drift compensation has been evaluated in a testbed with IEEE 802.15.4 compliant Tmote Sky motes [13] that have the same microcontroller, MSP430, as in the Shimmer nodes. A sensor node, connected to a PC, is acts as coordinator for five sensor nodes. The coordinator sends a broadcast message to reset the sensor nodes' local clocks. The sensor nodes are requested to send their local time periodically (every 30 seconds), used by the coordinator as input for monitoring and control of the clock drift. The clock drift compensation method in Section 2.2 is implemented. Fig. 8 shows the implementation of drift compensating function when sampling. A periodic timer is implemented for how often the sensor node should sample. Variable x is used for inter sample time and y is used for compensating drift. When the timer expires, a request is sent to read the AD-channel sample value, using the HplAdc12 interface [14] available in MSP430, and it will be stored. If a clock drift is calculated, the variable y will be set to 1 otherwise 0. Setting the value y to 1 compensates the drift by letting the periodic timer run for 1ms less.

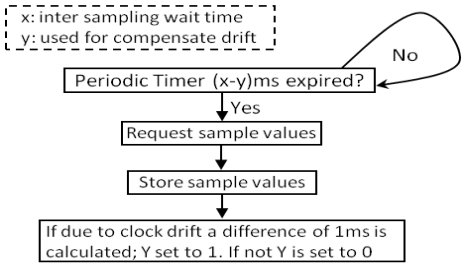

Fig. 8. A flow diagram for the drift compensating sampling algorithm.

Fig. 9 shows the maximum time difference between the two most drifting sensors (the sensor node with fastest and slowest clock) without drift compensation. The maximum time difference between 5 connected sensor nodes was approximately 5ppm. After 1 hour the time difference is 19ms.

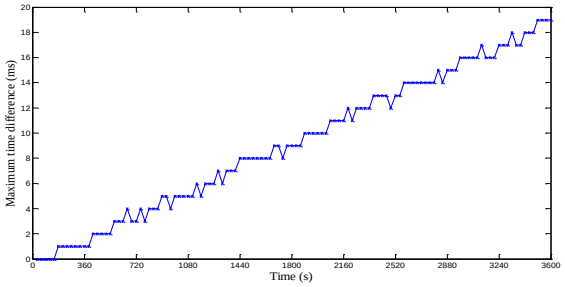

Fig. 9. The maximum time difference in ms between the two most drifting sensors (slowest and fastest clock) without drift compensating function.

Fig. 10 shows the effect of the drift compensating function. A maximum difference of 1ms is measured between two most drifting sensors measured at the testbed. The time from the two sensor node were transmitted every 30s during 1hour. The implementation of the algorithm results in a maximum clock drift of 1ms which will preserve data fusion for sampling rates up to 1kHz.

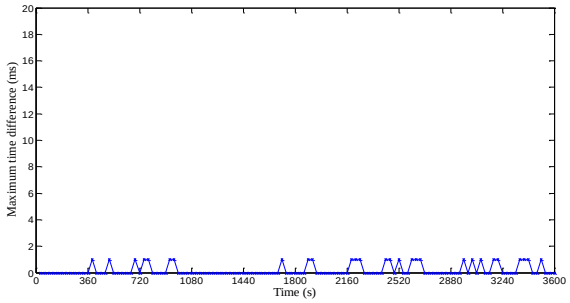

Fig. 10. The maximum time difference in ms between the two most drifting sensors (slowest and fastest clock) with drift compensating function.

3.5 Summary of Results

The performance evaluations in the previous sections show that the synchronization error is limited to around 1ms in ZigBee/IEEE 802.15.4 networks when a time synchronization protocol (Section 3.3.1) or broadcast messages (Section 3.3.3) are used. For Bluetooth, which so far is the prevailing communication technique between a mobile phone and wireless sensors, the errors increase as the number of connected nodes in a piconet increases. Our tests show that synchronization solely based on the clock in the mobile phone exhibits better performance than synchronizing the individual clocks in the sensor nodes (which sometimes is impossible). The algorithm to compensate for clock drift, due to variations in clock frequencies among the sensor nodes especially in low cost consumer equipment, has proven to be efficient (Section 3.4). Our study has resulted in performance limits (Fig. 5-7), in terms of synchronization errors, for prototype implementations of the algorithms in Section 2.

4. APPLICATIONS

Two real world use cases from two different fields, where time synchronization is important, are presented in this section.

4.1 Feedback in Kayaking Based on Synchronization

Flat water kayaking is an Olympic discipline in which a long and slim boat is propelled through the water by paddle equipped athletes. The motion in which the paddle transfers an athlete's force onto the water requires a complex, highly dynamic motor activity of the arms, shoulders, trunk and legs. Training supervision is ideally done by a coach, who primarily uses theoretical and personal practical knowledge and observation to derive subjective analysis and recommendations. Existing objective quantitative methods include time, distance measurement, heart rate measurement, GPS velocity tracking and video capturing. In a lab (e.g. on an ergometer) additional quantitative data can be accumulated by lactate and oxygen uptake measurement and motion analysis. All these methods are stand-alone technologies and can rarely be synchronized for a sophisticated data collection and analysis. Furthermore, there is only a small range of devices that can tolerate the conditions of on-water paddling; a desire for sophisticated on-water biomechanics measurement has been postulated by various researchers and coaches [10]. Based on the Mulle platform [11] a wireless sensor platform for on-water performance measurement on a kayak has been developed (Fig. 11).

Fig. 11. A Kayak system in action (left). Two paddle nodes (center, top) and the footstretcher (right, bottom) with an electronics box link via Bluetooth to a Java enabled mobile phone (right, top).

The system is designed to measure paddle and footstretcher force as well as motion parameters for real-time feedback (opportunity for motor learning through knowledge of performance) as well as post-processing. This platform as a successor of [12] consists of two Bluetooth enabled, battery powered paddle force measurement units with built-in accelerometer and gyro data acquisition as well as a Bluetooth node that includes nine axis motion sensors, a 10Hz GPS and six force transducers for point-of-pressure measurement for the individual foot on the foot stretcher.

The Bluetooth radio transmitted data is received on a regular mobile phone running custom J2ME software for data storage, visualization and analysis. Data from each of the three nodes is sampled at 100Hz and transmitted in packages of 32 byte (hull node), respectively 23 bytes (paddle nodes) to the phone. Since a crucial component of athletic performance is the coordination, the timing, of different movements, it is absolutely essential that knowledge of the temporal resolution and the time instance, at which a data sample on one sensor node with respect to the other two sensor nodes' clocks is obtained, is accounted for. Before the data is stored or analyzed all three sensor nodes' samples have to be aligned with respect to a common valid timeline.

One goal is to detect the timing between the force on paddle and by the legs on the footstretcher. This parameter can be used to analyse the performance of an elite athlete and to give feedback on synchronized or unsynchronized activation of upper trunk (paddle) and leg musculature (footstretcher). Furthermore, the very crucial parameter of boat velocity is determined by the GPS supported by an accelerometer (Fig. 12), both situated on and connected to the hull node.

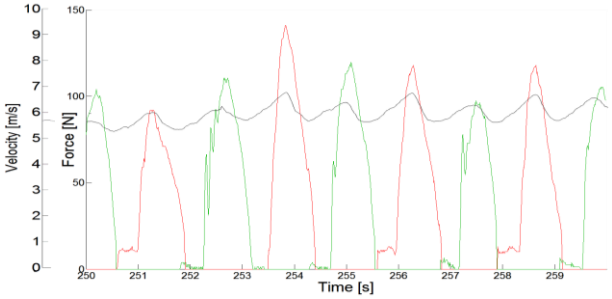

Fig. 12. The figure shows the forces from the right paddle node (green), the force from the left paddle node (red) and the velocity (black) measured on the hull node using the synchronization algorithm.

Synchronization is achieved by an algorithm executed on the mobile phone based on the method described in Section 2.1.1. The precision of this method has been determined to have a maximum synchronization error of 0.4±0.2ms, with a confidence grade of 99%. This has been deemed as sufficient for the current sampling rate as it is below 1/10 of the sampling period.

4.2 Monitoring in Neonatal Care

Cardiorespiratory monitoring in a Neonatal Intensive Care Unit (NICU), is one of the most vital monitoring system to presents secure the health of the infants [15]. Normally an ECG and a pulse-oximeter both connected via cables to the monitoring system are used. Fig. 13 shows information from a cardiorespiratory monitor (from the top); the ECG (green), the respiration (red), and the oxygen level from the pulse-oximeter (yellow).

Fig. 13. A photo of a cardiorespiratory monitor for an infant in neonatal care, which displays ECG, respiration and the oxygen level.

However the cables that connect the ECG and pulse oximeter to the cardiorespiratory monitor, complicates a natural and frequent skin-to-skin contact between the baby and the parents. Multiple studies have showed that kangaroo care, skin-to-skin contact between the infant and its parents, offers an environment that is as natural as possible for the infant to mature in. It improves the wellbeing not only for the infant but for the parents too. A wireless monitoring system can combine the goal with the requirement that an infant in NICU needs 24/7 monitoring of heart rate, respiration, oxygen level.

Since the monitoring is continuous over long time periods and the data rate is low, IEEE 802.15.4 radio communication seems to be an appropriate solution. A premature born baby may need extra oxygen for months after birth and therefore also monitoring of the oxygen level with a pulse-oximeter. False positive alarms, due to movements that lead to pulse-oximeter artefacts, are not uncommon and complicate a calm environment for the baby and the family. It is possible to minimize these alarms if the pulse-oximeter and the ECG are synchronized with each other [16]. Pulse-oximeter artefacts due to movements can then be identified and ruled out as true alarms. The tests presented in Section 3.3 show accurate time synchronization between two wireless sensor nodes can be maintained for hours.

5. CONCLUSIONS

Accurate time synchronization is essential for data fusion based on samples from multiple sensors. This paper has presented results of performance evaluation of major methods for time synchronization of sensor nodes in wireless personal area networks, mainly Bluetooth and ZigBee/IEEE 802.15.4. A new method for clock drift compensation is presented and evaluated. The performance results in this study are valuable guidelines when implementing synchronization algorithms for multi-sensor fusion, e.g. in sports and healthcare applications.

6. REFERENCES

[1] K. Römer, P. Blum and L. Meier, "Time Synchronization and Calibration in Wireless Sensor Networks", in Ivan Stojmenovic (Ed.): Handbook of Sensor Networks: Algorithms and Architectures, pp. 199-237, September, 2005.

[2] G. Coulouris, J. Dollimore and T. Kindberg, "Distributed Systems: Concepts and Design", Chapter 11, 4th edition, Addison-Wesley, 2006.

[3] D. Mills, J. Martin, J. Burbank and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, Internet Engineering Task Force (IETF), June, 2010.

[4] J. Wåhslén, I. Orhan and T. Lindh, "Local Time Synchronization in Bluetooth for Data Fusion Using Mobile Phones", International Conference on Body Sensor Networks (BSN 2012), Dallas, May, 2012.

[5] F. Cristian, "Probabilistic clock synchronization", Distributed computing (1989): pp. 146-158.

[6] M. Jadliwala, Q. Duan, S. Upadhaya, J. Xu, "Towards a Theory for Securing Time Synchronization in Wireless Sensor Networks", WiSec '09.

[7] J. Wåhslén, T. Lindh, M. Eriksson and I. Orhan, "A Novel Approach to Multi-Sensor Data Synchronization Using Mobile Phones", IJAACS, in print.

[8]

[9] M. Maróti, B. Kusy, G. Simon and A. Lédeczi, "The Flooding Time Synchronization Protocol", SensSys 2004, Balitmore, USA.

[10] Shimmer Research, http://www.shimmer-research.

[11] J.S. Michael, R. Smith and K.B. Rooney, "Determinants of kayak paddling performance", Sports Biomechanics, Vol. 8, No. 2, 2009.

[12] Mulle sensor node, http://staff.www.ltu.se/~jench/mulle.html

[13] D. Sturm, K. Yousaf and M. Eriksson, "A wireless, Unobtrusive Kayak Sensor Network Enabling Feedback Solutions", (BSN 2010), Singapore, June 7 - 9, 2010.

[14] Tmote Sky – IEEE 802.15.4 compliant sensor module from Sentilla (previously Moteiv).

[15] HplAdc12 interface for direct AD-sampling in the MSP430 MCU: http://tinyurl.com/7l7d54r.

[16] I. Murkovic, M.D. Steinberg and B. Murkovic, "Sensors in neonatal monitoring: Current practice and future trends", Technology and Health Care 11, 2003.

[17] L.K.L. Lum and P.W. Cheung, "Evaluation of Pulse oximetry with EKG Synchronization, IEEE Engineering in medicine and biology society 10th International conference, 1988.