Time Calibration in Experiments with Networked Sensors

Olivier Mehani, Ronnie Taib, Benjamin Itzstein

National ICT Australia
Eveleigh, Sydney, NSW, Australia
Email: {name.surname}@nicta.com.au

***Abstract*—Physiological sensors are widely used in user studies, often by practitioners with limited expertise in networking. However, large data volumes, and processing times often prevent the use of a single computer to collect the readings in real time. With multiple collection machines appear the problems of data aggregation and, more importantly, synchronisation. This paper describes how the OML reporting library allows solving the aggregation problem at low cost by introducing a lightweight instrumentation reporting to a centralised database. However, with unknown delays in network paths during aggregation and unreliable clocks on acquisition machines, synchronisation is hard to attain. We present a preliminary study of the theoretical feasibility of *post hoc* synchronisation corrections, supported by an experiment applying correction techniques to artificially impaired clocks and network transmissions. Based on the results of this experiment this paper highlights potential improvements.**

***Index Terms*—networked sensors, measurement, synchronisation, OML, TTP**

I. Introduction

Once confined to medical and psychological studies, physiological sensors are now widely used in studies ranging from theoretical human-computer interaction understanding to market research. This is due in part to the advancement of cognitive processes and their connection to physiological reactions, but also to the availability of affordable and easy-to-use sensors. It is common to find heart rate, temperature, galvanic skin response (GSR), electroencephalography (EEG), or eye tracking glasses (ETG) sensors used as base or control measures in many studies.

However, this trend comes with a side-effect of researchers finding difficulty to manage the data produced by the sensors. In particular, the following issues arise:

- Acquisition rates can be high, leading to large volumes of data to store;
- Sampling rates vary according to different time scales, leading to unaligned records;
- Sensors may be connected to different machines, leading to synchronisation issues;
- There is little support for cross-sensor synchronisation when using cheaper equipment.

A. Requirements

The overarching goal is to define a research framework for storing and synchronising physiological data acquired on a variety of equipment and machines. The main objective is to provide a good trade-off between cost, ease of use (practitioners are often not networking experts), and practical benefits to the researchers analysing the data. The requirements are to:

- Store high volumes of sensor data in a central location for easy analysis and combination;
- Be scalable to allow more sensors or acquisition machines to be added at any time;
- Be low cost and deployable on most current computers by non-experts.
- Provide synchronisation mechanisms, allowing for correction of clock discrepancies or network latencies between acquisition machines.

The latter point is crucial for the usability of the collected data, and has been extensively studied in distributed wireless sensor networks [1].

B. Network Reporting

The centralised data storage and scalability requirements inherently impose networking to be used for passing data from sensors or acquisition machines, to a central location. Several technologies address that specific issue with emphasis on different aspects.

The MQ Telemetry Transport (MQTT) open source protocol defines a lightweight broker-based publish/subscribe messaging protocol designed to be open, simple, lightweight and easy to implement [2], [3]. It is a data centric protocol allowing machines to exchange data, offering features like queuing messages when a registered subscriber is off-line, selectable levels of quality of service such as delivery and non-duplication guarantee. Part of the protocol is specifically designed for sensor networks, especially wireless networks.

While lightweight in principle, and offering programming interfaces for a multitude of languages, the protocol comes with many business-oriented features which may not be relevant or easy to deploy for user study experimenters. Furthermore, it focuses on message passing which means that additional technology (*e.g.*, IBM WebSphere) has to be integrated to store and format messages. Most importantly, while the quality of service can be tuned to regulate message order for example, the protocol does not seem to include any timestamping or specific support to synchronise messages at a fine level.

Figure 1. Participant wearing sensors.

Another option is the use of the OML framework [4]. While initially introduced for network experiments in testbeds, it can be used as a reporting system for any sort of set-up requiring moving sampled data from decentralised sensing devices to centralised storage databases. It offers automatic timestamping facilities. Moreover, its underlying data-transfer protocol has been implemented for multiple programming languages (*e.g.*, C, C#, Python), allowing for an easy integration with a wide range of sensors regardless of their software development kit of choice. The OML framework consists of a client library through which instrumented applications report sensor readings, and a data collection server which automatically stores them in a database for later analysis.

As practitioners, we found OML to be a promising solution. The server installation was easy and involved few dependencies. In this paper we illustrate a typical context of use: a user study involving a range of physiological sensors deployed on several machines. This paper also presents the results of a preliminary experiment highlighting the benefits of this approach for *post hoc* clock and network delay correction.

II. Context of Use

A. User Study Set-up

The data storage and synchronisation framework targets user studies where several physiological or behavioural sensors are in use simultaneously to monitor a participant under different conditions. To ground this paper in practice, we deployed OML for a study focusing on drivers' behaviour, conducted within a driving simulator (Figure 1). This specific example was selected as it includes a variety of sensors requiring distinct acquisition devices, and hence should reflect the constraints researchers may face in a wide range of user studies, whether carried out in the lab or *in situ*.

The study focuses on the physiological and behavioural reactions of drivers subjected to increasingly complex tasks while driving [5]. The various data streams need to be finely synchronised to allow multimodal analysis.

B. Sensors

The sensors used in the experiment were relatively low cost, and easy to use sensors. As can be seen in Table I, the sensors all had different sampling rates and numbers of channels. In fact, the sampling rates per channel varied, *e.g.*, blink detection

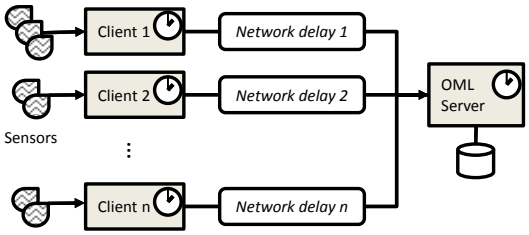

Figure 2. Typical user study set-up.

being slower than gaze direction changes. Moreover, some of the signals required heavy software processing (EEG and ETG in particular), so the data acquisition needed to be spread across several machines.

A more dramatic issue yet was linked to inconsistent acquisition methods (polling vs. event based), programming interfaces, and data formats reported by the devices. Most devices introduced internal sub-sampling, buffering and other mechanisms that may obscure the true time of each data point. Some devices provided timestamping, based on the local computer, other based on their internal clock, and others did not provide any timestamping at all. These issues were mostly addressed through the careful design of acquisition programs, able to unpack buffered data and re-timestamp it using the local machine time.

C. Aggregation

Once done, we assumed that all incoming signals were synchronised and timestamped according to their local machine clock. The next step was to ensure overall synchronisation across acquisition machines, and centralised storage. This is where the OML framework became crucial.

As illustrated in Figure 2, individual OML clients were attached to each device but shared the same experiment identifier within OML. Since OML covers data transmission and storage, as soon as the clients were started, each data reading was streamed to the OML server, and stored in a PostgreSQL database. This automatically provided us with centralised consistent access to all data streams.

In the following section, we examine timestamping mechanisms that can be used *post hoc* to detect and correct clock discrepancies between machines, as well as network delays.

III. Timestamp Correction

A. Principles

While the Network Time Protocol (NTP) can in theory be used to keep all acquisition machines synchronised, its deployment may not always be possible or desirable. One of the issues encountered is where a machine's clock receives a step update. It is also not the best solution for small or embedded measurements devices, as discussed in [1].

More specifically for experiments requiring fine synchronisation between input signals, absolute time is not paramount. Instead, strict order between clocks, and duration between events are of essence [6]. In the following sections, we explore

Table I
SENSORS USED IN THE EXPERIMENT, AND THEIR SAMPLE-ACCESS CHARACTERISTICS.

Sensor	Sampling rate	Data streams	Device
Eye tracking glasses	30 Hz	3 channels: gaze direction, pupil dilation, blinks	SMI ETG
EEG	128 Hz	14 channels	Emotiv Epoc
GSR and BVP	512 Hz	2 channels: skin conductance and blood volume pulse	Procomp Infiniti
Posture	500 Hz	16 channels: pressure, distance	2×Phidget 8/8/8
Simulator events	50 Hz	10+ channels	C# interface
Annotations (by experimenter)	125 Hz	1 channel: key logging	Keyboard

how to leverage OML timestamping mechanisms to approach these requirements.

B. Notations

For each reported sample, OML offers two timestamps, from different vantage points: the client and the server. The former is added by the measuring client at the time of recording of the sample. The timestamp accuracy is therefore only dependent on the local clock of that client. The latter is added upon reception of the sample by the server. Its accuracy is impacted by both the server's clock, and the network delay from the client. We formalise these relations here.

The real time of occurrence of an event e is t_e. This event can be observed by a node n (server or client). Each node n has a clock skew rate $\mathrm{d}\Delta_n(t)/\mathrm{d}t$, and a current drift from the real time of $\Delta_n(t)$. When node n measures the time at which e occurs, it reads

$$c_n(t_e) = t_e + \Delta_n(t_e). \tag{1}$$

When reporting an event e through OML, a client timestamps the sample with $c_n(t_e)$, then sends it to the server s. It is received at time $t_e^{r_n} = t_e + \Delta_{n,s}(t_e)$, where $\Delta_{n,s}(t_e)$ is the transmission delay between n and s, composed of the network and queuing delays on both sides. The server stores that reception time as an additional timestamp based on its own clock

$$\begin{aligned} c_s(t_e^{r_n}) &= t_e^{r_n} + \Delta_s(t_e^{r_n}) \\ &= t_e + \Delta_{n,s}(t_e) + \Delta_s(t_e + \Delta_{n,s}(t_e)). \end{aligned} \tag{2}$$

By introducing a sequence of synchronisation events $i = 1, \ldots, n$, simultaneously recorded by various nodes or sensors, we can study the error between timestamps for different clients. In particular, the client-to-client *timestamp* comparison,

$$C_{n_1,n_2}(i) = c_{n_1}(t_i) - c_{n_2}(t_i) \tag{3}$$

allows evaluating the synchronisation, or lack thereof, of clocks at nodes n_1 and n_2 at time t_i. The client-to-client *reception time* comparison,

$$S_{n_1,n_2}(i) = c_s(t_i^{r_{n_1}}) - c_s(t_i^{r_{n_2}}) \tag{4}$$

allows quantifying the impact of network delays from each client to the collection server.

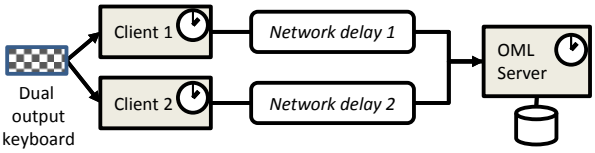

Figure 3. Experimental set-up: two clients observe the same (mechanical) keyboard event, and report it to the collection server. Depending on the test case, different impairments are introduced in the form of clock drifts or network delays (either static or incremental).

C. Experiment

An experimental set-up involving an OML server and two clients, each connected via a test router, was deployed to assess correction when one of the nodes (hereafter denoted *impaired*) experiences the following adverse conditions (see Figure 3).

- *base*: no clock drift, normal network delay
- *clockoffset*: static clock drift of -1 s
- *clockoffset_inc*: static clock skew -0.5 ms/s, using the `adjtime` Unix function with a target correction of 1 s
- *netdelay*: a one-way delay of 1 s is added at an intermediate router using `netem`, resulting in an average 2 s round-trip time (RTT, ±1 s). This is an extremely adverse condition, *e.g.*, compared to typical latency maxima of 470ms for international transmissions
- *netdelay_inc*: the one-way delay was manipulated to increase by a 250 ms step every 100 s

Two additional scenarios, labelled *all* (and *all_inc*) combined impairments: netdelay on one client, clockoffset on the other client (and their incremental flavours, respectively). In all cases, each client uses the `ping` utility to estimate the RTT to the server, d_i, which they also report through OML.

For each condition, a 5-minute test was run with a mechanically modified keyboard sending simultaneous signals to both clients. Every 100 s, the 26 alphabetical keys are pressed in sequence, providing a set of exact time references for both clients, since the signals are mechanically synchronised. The OML timestamps are recorded for both clients, $c_n(i)$, and the server, $c_s(i)$, and their analysis provides the basis of the synchronisation correction.

D. Time Correction

A time-transmission protocol (TTP) has been proposed in [7]. TTP allows a node to communicate its current clock

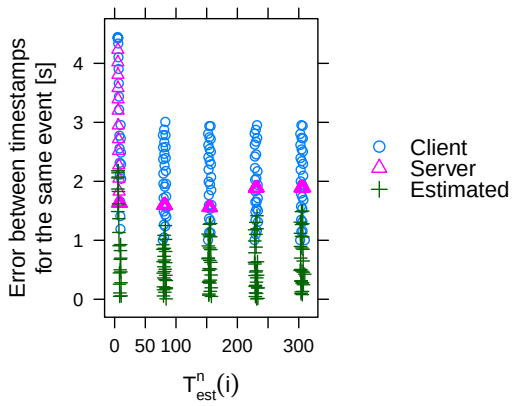

Figure 4. Error in timestamp for the same event as observed from different sources (client (3), server (4) and estimated) for the *all* scenario.

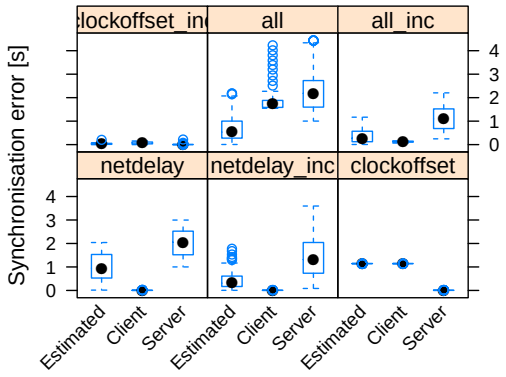

Figure 5. Errors throughout conditions

reading to a remote node. Network delays are accounted for by periodically measuring the RTT. TTP is used in [7] to derive a feedback probabilistic clock synchronisation algorithm. We use TTP differently in this proposal. We use a feed-forward approach, as suggested in [6], to derive estimated timestamps:

$$T^n_{est}(i) = c_s(i) - \frac{1}{i}\sum_{j=1}^{i} c_s(j) + \frac{1}{i}\sum_{j=1}^{i} c_n(j) + \frac{1}{i}\sum_{j=1}^{i} d_j \quad (5)$$

After deriving the corrected T_{est} for the events in each of our test cases, we compare the differences between the timestamps of the same event as observed from two different sources. Ideally, the difference should be null. Figure 4 shows these differences, for the scenario with both static drift and network delay added; for each key event, it compares client timestamps (3), server timestamps, and estimated timestamps. In this case, the TTP algorithm successfully absorbs some of the error in synchronisation, and performs better than either using plain client (2–4 s) or server (1–3 s) timestamps. The remaining error is however still not negligible, of the order of a second.

Figure 5 shows a summary of the errors for all test cases. In all cases, TTP allows mitigating the largest error, and performs better than the impaired timestamp ($c_n(i)$ in clock-offset experiments, $c_s(i)$ in netdelay experiments). However, the non-impaired timestamp, if known, still provides better event synchronicity. TPP is still useful, in case the magnitude and occurrence of impairments is not known, as it provides a reduction in errors.

IV. Conclusion

Motivated by the need for multi-client sensing in physiological studies and simplicity of deployment, we have argued that the use of a network reporting system can simplify the set-up of experiments using only off-the-shelf equipment. Using the OML reporting library, a complex experiment using various sensors could be easily set up, and the data centrally collected . However, multiple network clients pose the problem of time synchronisation, for the data to be meaningfully usable.

This paper proposed the use of TTP to provide practical *post hoc* timestamp correction, as well as an experimental set-up to study it. We set-up a series of experiments emulating various severe time impairments, and proposed the use of the previously introduced TTP as a feed-forward correction method. We showed that TTP managed to provide better results than the worst impaired system, thus validating the feasibility of *post hoc* clock correction based on OML timestamps.

In our experiment, strongly adverse conditions were simulated by combining clock offset with network delays of up to 2 ± 1 s RTT, but managed to reduce the errors between affected clients to below one second, which can also correct message ordering. This is a very encouraging result, hence future work will focus on improving our TTP-based approach further, as well as studying the performance of other feed-forward time correction algorithms. Ultimately, we hope to integrate such mechanisms directly into the reporting framework, saving the experimenter from some post-processing before their data can be analysed.

References

[1] K. Römer, P. Blum and L. Meier, 'Time synchronization and calibration in wireless sensor networks', in *Handbook of Sensor Networks*. 23 Sep. 2005, ch. 7. DOI: 10.1002/047174414x.ch7.

[2] IBM and Eurotech, *MQTT v3.1 protocol specification*, 2010.

[3] V. Lampkin, W. T. Leong, L. Olivera *et al.*, *Building Smarter Planet Solutions with MQTT and IBM WebSphere MQ Telemetry*, SG24-8054-00. 2012.

[4] O. Mehani, G. Jourjon, T. Rakotoarivelo and M. Ott, 'An instrumentation framework for the critical task of measurement collection in the future Internet', *Comput. Netw.*, vol. 63, 22 Apr. 2014. DOI: 10.1016/j.bjp.2014.01.007.

[5] R. Taib, J. Tederry and B. Itzstein, 'Quantifying driver frustration to improve road safety', in *CHI 2014*, Toronto, Ontario, Canada, 26 Apr. 2014. DOI: 10.1145/2559206.2581258.

[6] J. Ridoux and D. Veitch, 'Principles of robust timing over the Internet', *ACM Queue*, vol. 8, no. 4, 21 Apr. 2010. DOI: 10.1145/1755884.1773943.

[7] K. Arvind, 'Probabilistic clock synchronization in distributed systems', *IEEE Trans. Parallel Distrib. Syst.*, vol. 5, no. 5, May 1994. DOI: 10.1109/71.282558.