Fault-Tolerant Clock Synchronisation with Microsecond-Precision for CAN Networked Systems

Dongik Lee, Jeff Allan

A fault-tolerant clock synchronisation technique is presented. In a distributed system the discrepancy between a node's view of current time and the rest of a system can cause critical deadlines to be missed. It may also be the cause of many unknown system errors. In fact, many real-time applications, such as redundancy management, synchronous data acquisition and simultaneous triggering of actuators at several nodes, are impossible without such a global reference time. DRTS Ltd have developed and have protected a software-based fault-tolerant clock synchronisation technique for broadcast networks such as CAN. It provides a predictable and reliable service that enables networked system synchronisation to micro-second precision using negligible network bandwidth.

1. INTRODUCTION

Distributed real-time systems are now found in many industrial fields, such as process automation, oil and gas production, and automobiles. However, distributed environment presents significant challenges to the system designers:

- Communication networks inevitably introduce delays due to limited bandwidth and overhead; and
- It is difficult to achieve synchronised temporal behaviour and data consistency between computing nodes which are physically separated.

The key to solving these problems is the unification of the various representations of time across the system. For example, the former problem can be solved by using time-triggered communication protocols, such as TTCAN and TTP/C, where synchronised clocks are the fundamental requirement.

The quartz oscillators used in computers and networking equipment change with age and are affected by environmental variables, such as mechanical vibration and temperature. Consequently, clocks can drift up to several seconds per day. The discrepancy between a node's view of current time can cause critical deadlines to be missed leading to system failure. It may also be the cause of many unknown system errors.

In this paper, a software-based fault-tolerant clock synchronisation algorithm for CAN networked systems is presented. The proposed algorithm is based on a master-slave structure and the well-known *a posteriori* technique (Gergeleit & Streich, 1994). The proposed algorithm deterministically guarantees an upper-bound for the clock skew and the number of messages required for synchronisation. The key advantage of the proposed method is the capability to tolerate faulty master clocks in a systematic and predictable way.

2. THE BENEFITS OF SYNCHRONISED CLCOKS

The benefits of synchronised clocks for commercial computer networks, such as financial transactions and stock trading, have been understood well. Skoog (2001) described clock synchronisation as "a key factor in success or failure of a networked system". As a result, most of computer networks are equipped with time servers and synchronisation algorithms, such as Network Time Protocol (NTP).

Clock synchronisation techniques are also beneficial for many industrial applications:

Time-Triggered Communications

In safety-critical applications, such as X-by-wire cars, randomly varying message latency causes significant concern. In response to this concern, time-triggered protocols based on scheduling of the communication resources are becoming a common solution for safety-critical and hard real-time systems. For the correct operation of a time-triggered network, it is essential to provide a system-wide time reference to enable a consistent identification of the time at which timeslots trigger.

Control Systems

A distributed control system may suffer significant time-varying delays (i.e., jitter) between the sampling of the sensors and the reaction of the actuators. This jitter influences the system's stability as well as changes the system characteristic into a time-varying one for which theoretical results for time-invariant systems cannot be used. A main cause for the jitter is polling of the sensors by the controller (Eidson & Cole, 1998). Using synchronised clocks the control objective can be achieved at a lower sampling rate and communication bandwidth.

Redundancy Management

In safety-critical applications, some degree of hardware redundancy is commonly used to meet the system requirement. Usually, the redundancy management strategy is based on a voting mechanism requiring high degree of data consistency. Without synchronised clocks, voting cannot work properly.

Synchronous Data Acquisition

The existence of global time reference is the fundamental requirement for data acquisition systems (DAQs) for distributed environment. For instance, many DAQs use timestamps to induce a total ordering on the events that occur at different nodes. If the clocks in the system are not synchronised, it is possible for the event to be seen as like affected by the causes from the "future".

3 . AN OVERVIEW OF CLOCK SYNCHRONISATION TECHNIQUES

3.1. Clock Synchronisation Blocks

Clock synchronisation is a technique to generate an approximate system-wide time reference, the so called "approximate global time", using local clocks, so that at any instant any two non-faulty clocks agree on the current time within a known bound. Several researchers (Schneider, 1986; Anceaume & Puaut, 1998) stressed that any software clock synchronisation algorithms consist of the three blocks for:

- The detection of resynchronisation time—to trigger each node to start the clock synchronisation algorithm;
- Reading remote clock values—to obtain information about the remote clock values; and
- Clock correction—to establish an approximate global time reference, and to calculate the correction term.

A clock synchronisation algorithm having these three blocks can be implemented as in figure 1, where F_c is a "convergence function" to bring clocks closer together. adj_p^{k+1} is the amount by which the requesting clock *p* differs from correct value at the resynchronisation round of (k+1). Different choices for these blocks result in different clock synchronisation algorithms. For the surveys on various techniques to implement these blocks, see Schneider (1986), Ramanathan et al (1990), and Anceaume & Puaut (1998).

k=0;
$adj_p^{\,0}$=0;
do forever
  detect resynchronisation event at time R^{k+1};
  $adj_p^{\,k+1} = F_c(p, x_1^{\,k+1}, ..., x_N^{\,k+1}) - p$;
  k=k+1;
end

Figure 1. Clock synchronisation blocks.

3.2. CAN-Based Algorithms

Different clock synchronisation algorithms can be designed according to the network used. CAN has a number of unique characteristics. In particular, a set of error handling facilities and simultaneous delivery offer the potential for achieving precise clock synchronisation. On the other hand, it has relatively low bandwidth which favours simple algorithms.

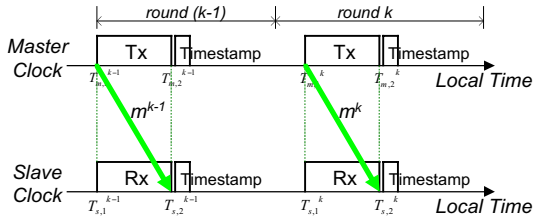

Figure 2. Clock synchronisation based on the *a posteriori* technique.

Gergeleit & Streich (1994) proposed a clock synchronisation technique based on a master-slave structure and a simplified "*a posteriori" agreement* technique (see figure 2). A clock in the system is designated as the master, which periodically broadcasts a synchronisation message that provides a reference time value. In this method, timestamps are taken right after a message is delivered, rather than before broadcasting the message. In the synchronisation round k, the master broadcasts a synchronisation message m^k which contains its timestamp taken at $T_{m,2}^{k-1}$ when the previous synchronisation message m^{k-1} was delivered to the slaves. Every slave in the system simultaneously receives this message at $T_{s,2}^{k}$, and takes a timestamp right after the reception. Each slave clock then calculates a correction term using the difference between the timestamps $T_{s,2}^{k-1}$ and $T_{m,2}^{k-1}$. The key advantage is the high precision that does not depend on message latency. However, the lack of capability to tolerate a faulty master is the major concern.

Eriksson et al. (1996) proposed a token-based approach to provide a fault-tolerance capability with Gergeleit & Streich (1994). In each resynchronisation round a different clock is designated as the master, and then the other clocks synchronise to the selected master. However, it uses no mechanism to detect faults in the current master. In addition, if f successive clocks are faulty, then the system can be left without synchronisation during the period of fR.

The *a posteriori* technique proposed by Verissimo & Rodrigues (1992) has also been applied to CAN networked systems (Rodrigues et al., 1998). This algorithm is based on a fully distributed structure which is inherently fault-tolerant. However a drawback is the complexity resulting from the distributed structure. The number of messages required is N(N+2) for an N-node system.

4 FAULT-TOLERANT CLOCK SYNCHRONISATION FOR CAN NETWORKED SYSTEMS

As discussed in the previous section, a major concern with typical synchronisation algorithms for CAN is the vulnerability to a faulty master clock. This section describes a novel CAN-based clock synchronisation algorithm which is capable of tolerating faulty clocks with relatively simple structure.

Table 1. Summary of CAN properties relating to the clock synchronisation (taken from the work by Rodrigues et al., (1998).

A1: Validity—If a correct node broadcasts a message, then the message is eventually delivered to a correct node.
A2: Best-effort agreement—If a message is delivered to a correct node, then the message is eventually delivered to all correct nodes, if the sender remains correct.
A3: Simultaneous agreement—If the sender remains correct, the last retransmission of the same message is delivered to all correct nodes at real time values that differ, at most, by a known interval.

4.1. CAN Properties and Assumptions

The algorithm is based on the CAN properties in table 1. These properties, which are achieved by the built-in error handling mechanism of the CAN protocol, provide fault-tolerant broadcast. That is, within a CAN network a message is accepted either by all nodes or by no node. Property A3 is crucial for achieving a high precision of synchronisation using the *a posteriori* technique. Properties A2 and A3 remove the possibility of Byzantine faults.

4.2. Outline of the Proposed Algorithm

The main feature of the proposed algorithm is outlined as follow:

Centralised structure: A master-slave structure is employed in order to create as simple algorithm as possible. The master-slave structure can drastically reduce the number of messages if a broadcast network is used.

Multiple-master: A multiple-master method is used to tolerate a faulty master. The novelty of the suggested method lies in the use of two groups of master candidates—'Master Candidates Group' (MCG) and 'Substitutes Group'—to reduce bus traffic caused by a master selection process.

Resynchronisation detection: A start message is used for triggering a synchronisation round.

Clock correction: All clocks in the system synchronise to the time of a selected master clock. The *a posteriori* technique is used.

4.3. Subsets of Clocks in the System

The major drawback with a multiple-master technique is the need for a master selection mechanism. Selection algorithms are usually complicated and introduce extra loading on the system in terms of the number of messages and the processing time. The number of messages required for the selection mechanism increases with the size of the multiple-master cluster. The larger size of the multiple-master also leads to the higher complexity in the selection mechanism, which is not desirable.

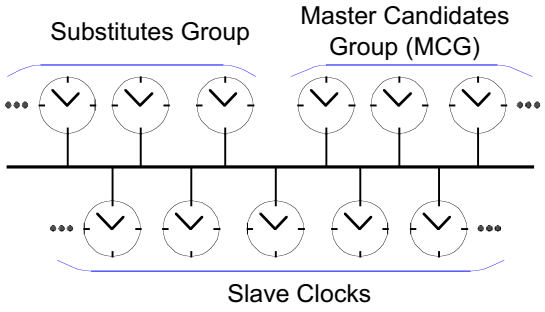

Figure 3. Subsets of clocks in the system.

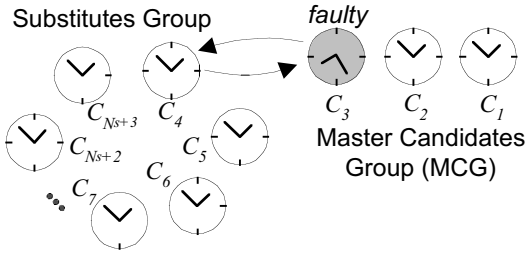

Figure 4. Replacement of a faulty candidate clock.

In this work, to overcome these problems, all clocks in the system are divided into three subsets (figure 3). At each synchronisation round, only clocks in the MCG take part in the selection of a master. Clocks in the substitutes group do not take part in the selection, and are only for replacing faulty clocks in the MCG. The rest of the clocks in the system are considered to be slaves, which have to synchronise to the selected master clock and are not required to broadcast any message for clock synchronisation.

An example given in figure 4 explains how the MCG and the substitutes group work in order to update the MCG. Each node of the MCG examines its clock value by comparing with the current master clock time. When a clock in the MCG is found to be faulty, a non-faulty clock in the substitutes group becomes a new member of the MCG. In figure 4, for example, the faulty candidate C_3 is replaced with C_4.

Clock C_3 takes the place of C_4, rather than being removed from the system.

The number of clocks for each group can be chosen by the system designer to achieve the desired level of fault-tolerance. The size of MCG to tolerate f_m faults is given by

$$N_m = 2f_m + 1 \qquad (1)$$

Note that f_m denotes the maximum number of *new* faulty clocks of the MCG that can arise in a *single* resynchronisation round. Thus, the complexity of the selection mechanism is not directly proportional to the total number of faulty clocks assumed in the system. On the other hand, the size of the substitutes group, N_s, depends on the total number of faulty clocks, f, to be tolerated in the system. It seems useful to choose N_s as a multiple of f_m, to achieve η-modular set of redundant clocks to substitute for faulty master candidates; that is,

$$N_s = \eta f_m,\ \eta = 1,2,... \qquad (2)$$

Since $f_m << f$, the proposed algorithm can achieve a desirable degree of fault-tolerance using a simple selection mechanism and a lower number of message exchanges.

4.4. Master Clock Selection

Figure 5 shows the entire steps for the suggested algorithm. It is assumed that at most $f_m = 1$ new faulty clock can be found in the MCG in a single synchronisation round, and thus three clocks (C_1, C_2 and C_3) for the MCG are needed. In this example clocks C_1 and C_3 are assumed to be the best and the faulty, respectively. Arrows represent a broadcast of message with unknown latency.

Selecting a master is performed in the first two steps of figure 5. The selection function (Fv) is based on the timestamps (T_1, T_2, T_3) taken by each candidate clock on the arrival of a start message (m_{start}). The CAN properties described in table 1 guarantee that all the candidates receive the start message simultaneously. As soon as a timestamp has been taken, each candidate, including the sender of start message, broadcasts a time message that contains its timestamp, and then waits for other candidate's time messages. According to A2 in table 1, all the correct candidates will obtain an

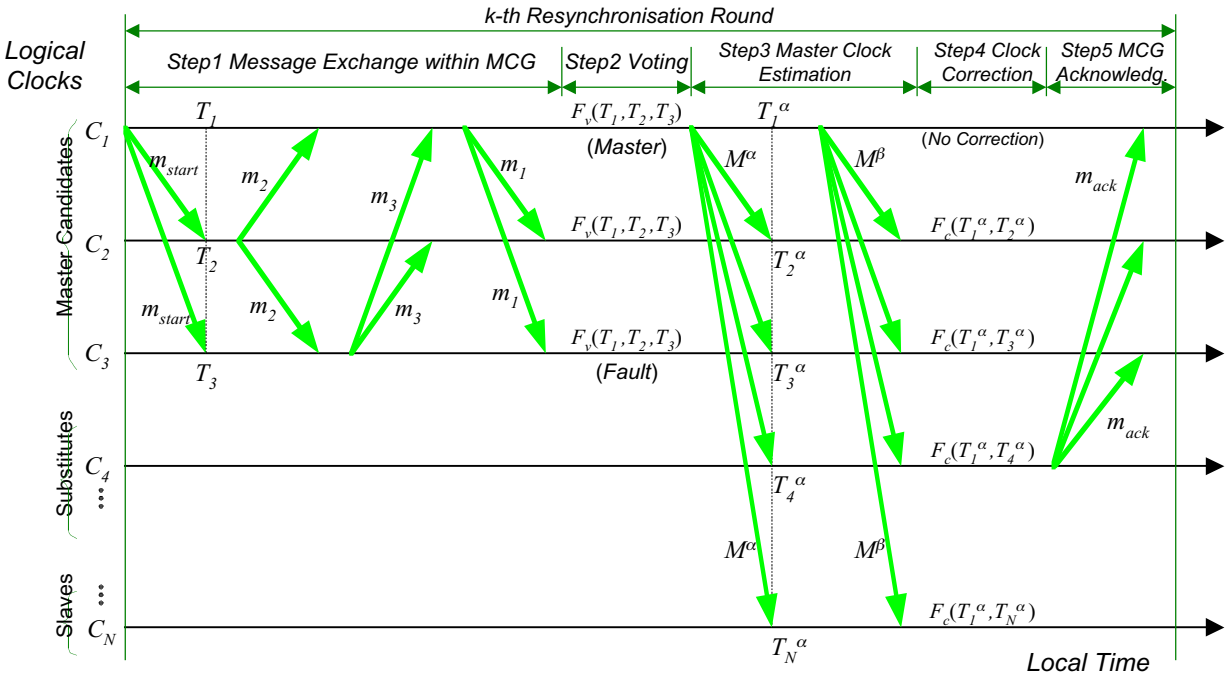

Figure 5. A timing diagram model of the synchronisation algorithm.

identical set of timestamps, and thus, will vote for a common clock as the master.

In addition to the selection of a master, the selection mechanism also identifies any faulty clocks in the MCG. Any candidates whose time differences with the median value are larger than a predefined threshold will be considered faulty. The faulty clocks abandon themselves as candidates for a master and move to the substitutes group, whilst the selected master builds a list of new candidates for the following synchronisation round.

A key advantage with the proposed selection mechanism is its robustness. Since all the timestamps needed for selection are taken at once, any delays (or even missing deadlines) in sending them do not lead to synchronisation failure. On the start messages, the fastest clock in the MCG may broadcast a start message. However, the selection mechanism still gives a correct result even though the resynchronisation round starts earlier due to the fastest clock (or a faulty clock in the worse case), since the selection algorithm relies on the fact that a start message has arrived rather than the time when it was generated.

4.5. Clock Correction

Except for the selected master, all clocks in the system synchronise to the master clock. Step3 and step4 in figure 5 are related to the clock correction mechanism. The correction term is calculated in a similar way to the master selection mechanism. In this work two messages, M^{α} and M^{β}, are successively broadcast.

4.6. Substitution of Faulty Candidates

Each synchronisation round ends by updating the MCG with a set of non-faulty clocks (step5). The key to this process is the acknowledgement messages sent by the substitutes. Only the substitutes requested by the current master send the acknowledgement messages containing binary information, i.e., *accept* or *refuse* depending on the sender's clock status.

4.7. Analysis

It is not feasible to describe in this paper the details of analysis on the achievable synchronisation precision and the number of messages. To summarise, the worst case synchronisation skew between any two clocks is given by:

$$\delta = 4\rho R + \xi \qquad (3)$$

where, ρ, R, and ξ denote drift rate, resynchronisation period, and reading error, respectively. To tolerate f_m faulty candidates in a single resynchronisation round, the total number of messages for the synchronisation algorithm is given by:

$$2f_m + 4 \le n_{msg} \le (\eta + 2)f_m + 4 \qquad (4)$$

where, η=1,2,…is a parameter to be selected by the system designer according to the desired degree of fault-tolerance. Note that eqn (4) represents the number of *messages* rather than the number of bits. Therefore, the maximum bandwidth used by the synchronisation messages can be obtained by assuming every message has 8 bytes of data.

5. EXPERIMENTAL RESULTS

The performance and effectiveness of the suggested method have been assessed and demonstrated on a test system. See figure 6 (a) and (b). It consists of a set of micro-controllers, a CAN network, DRTS X-View monitoring software, and a three-actuator rig. In this work, resynchronisation period (*R*) and bus speed are chosen by 1sec and 250Kbps, respectively.

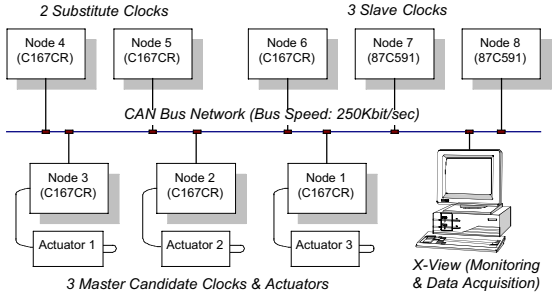

Figure 6(a) Schematic overview of the experimental setup.

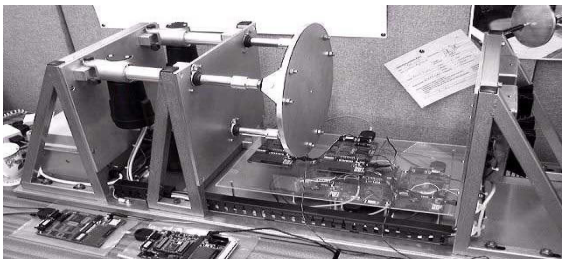

Figure 6(b) Picture of the three-actuator rig.

5.1. Synchronisation Precision

The drift rate of a physical clock used in this experiment is $\rho \approx 1.5\mu$sec. The best synchronisation precision can be seen at the time when every node adjusts its local clock, that is $\delta = \xi$ with R=0 and $\xi = 4\mu$sec. ξ can be reduced to 1μsec at bus speed 1Mbps. The worst case clock skew is seen at the right before starting a new resynchronisation cycle. Figure 7 shows that the worst case clock skew is approximately $\delta = 10\mu$sec.

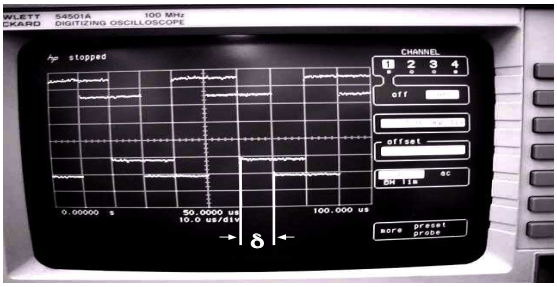

Figure 7. Screen shot of the oscilloscope showing the worst clock skew between two clocks is $\delta = 10\mu$sec.

5.2. Fault-Tolerance

The main issue with the fault-tolerance mechanism of the suggested algorithm is replacement of a faulty candidate clock with a non-faulty substitute clock. The ability of tolerating faulty clocks is illustrated in figure 8 by showing the transitions of each clock status, which are Master, MCG, or Substitute. At t=12sec, the manual reset button of clock 1 is pressed, and as result its clock status changes from MCG to Substitute. On the other hand, clock 4, of which status was Substitute, becomes a new member of the MCG instead of clock 1. Faults are injected to the rest of clock nodes in the similar way.

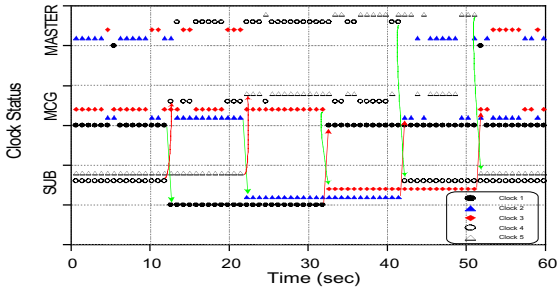

Figure 8. Transitions of clock status.

5.3. Deterministic Communications with Standard CAN

The CAN with over 250 million nodes installed world-wide has achieved widespread acceptance. Basing a time-triggered strategy on this protocol that can use established technical resources and also existing hardware, is seen to have enormous advantages.

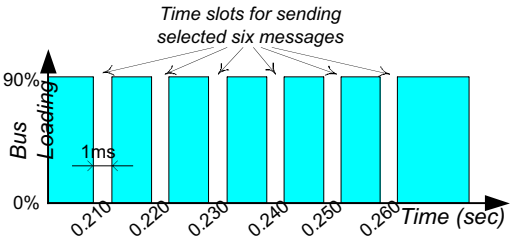

Figure 9. Profile of background bus loading used to measure the worst case latency at load level 90%.

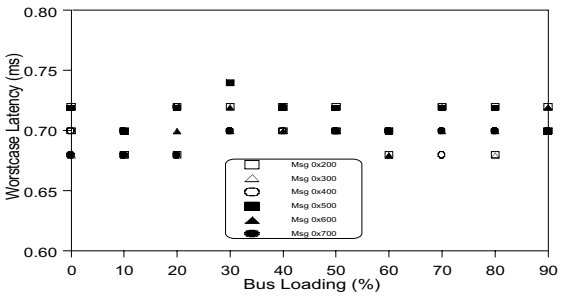

Figure 10. Worst case message latency using the clock synchronisation technique.

This experiment demonstrates the latency with a time-triggered strategy where a unique time slot is designated for each message to broadcast (see figure 9). Background bus loading of 90% is eliminated during these timeslots. If any clock among either the load generators or

the message senders fails to synchronise with the rest clocks in the system, the message latencies must vary depending on the bus load conditions and the priority position.

As illustrated in figure 10, applying the suggested synchronisation technology to a standard CAN network enables deterministic communication. A typical true latency of message with 8-byte data at bus speed 250Kbit/sec is 520μsec. Adding to this value with the total processing delay (180μsec with this experimental setup) gives the latency around 700±20μsec. Jitters by ±20μsec are mainly due to the granularity of logical clocks used for the measurement. The jitters mostly do not exceed the clock granularity at the higher bus loading, indicate that any single message has not violated the predefined time slots designated for other messages. As result, even under high bus loads (i.e., 90%) every message is delivered with a latency equal to the time taken by the highest priority message (i.e. as if there was no bus load).

5.4. Synchronised Motion Control

One of the most important aspects of safe design is simplicity. The reliable and accurate clock synchronisation technique results in a very simple control system since the complex mechanism to deal with the jitter in control delays are avoided by using a time-triggered architecture based on synchronised clocks.

In this example, the control objective is to control the three motors (see figure 6a) such that they follow the demand positions with a highly synchronised motion. Each motor is controlled by its local controller. To achieve this objective, each controller needs to compare the actuator positions measured simultaneously. Therefore, in addition to the message latencies, data consistency errors on sampling of the sensors can also be a critical problem. In this work, the feedback loop is implemented by a simple Proportional (P) control law with a sampling period of 40msec. The background bus load level is 90%.

The results in figure 11 indicates that each actuator moves very closely from each other when clocks are synchronised. In contrast, the result without clock synchronisation illustrates that the excessive disagreement on the actuator positions led to the actuators being stuck.

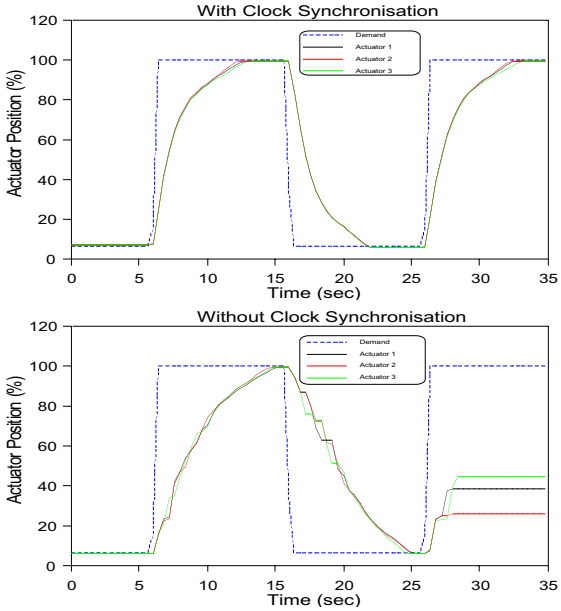

Figure 11. Step responses of the actuators with/without clock synchronisation.

6. CONCLUDING REMARKS

A deterministic and fault-tolerant clock synchronisation algorithm has been presented. The algorithm exploits a unique clustering method and the *a posteriori* technique. This approach is easy to implement. It can be used immediately and no new hardware is necessary. The experimental results demonstrate that the proposed approach enables time triggered communications that can be implemented in standard CAN nodes. It has also been demonstrated that the synchronised clocks can achieve the three-actuator control requirements with a simple P-controller at a lower sampling rate.

The key features of this algorithm are summarised as follow:

- **Precision**: Approximately 1μsec at 1Mbps, or 4μsec at 250Kbps.

- **Simplicity**: A master-slave structure which is simple to implement in any embedded real-time applications using CAN.
- **Robustness**: Synchronisation services remain available in the presence of clock faults or a partly disconnected network.
- **Network load efficiency**: Low bus bandwidth is used; e.g., <0.1% at 1Mbps, <0.4% at 250Kbps.
- **Predictability**: The key parameters can be precisely determined in the design phase.
- **Flexibility**: Slave nodes as well as master candidate nodes can be added to or removed from the network on-fly.
- **Low cost**: A software based algorithm implemented on standard CAN that can provide a cost-effective solution to safety-critical applications.

Note that the proposed algorithm can be applied to any broadcast protocols as long as the assumptions described in table 1 are satisfied. For example, the proposed technique, especially the fault-tolerance mechanism, can be used for sensors/actuators complying with the IEEE-1588 standard to improve the reliability of the global time reference (Allan & Lee, 2003).

REFERENCES

[1] Allan, G. & Lee, D. (2003). “A solution for fault-tolerant IEEE1588”, *Workshop on IEEE1588 Standard*, September, Gaithersburg, USA.

[2] Anceaume, E. & Puaut, I. (1998). “Performance evaluation of clock synchronization algorithms”, *Tech. Report N3526*, Unite de recherche INRIA Rennes, IRISA, Campus universitaire de Beaulieu, 35042 Rennes Cedex, France.

[3] Eidson, J. & Cole, W. (1998). "Ethernet rules closed-loop system", *InTech*, pp.39-42, June.

[4] Eriksson, C., Thane, H. & Gustafsson, M. (1996). “A communication protocol for hard and soft real-time systems”, Pr*oc. IEEE European Workshop on Real-Time Systems (EURWRTS)*, L’Aquila, Italy, June.

[5] Gergeleit, M. & Streich, H. (1994). “Implementing a distributed high-resolution real-time clock using the CAN bus”, *Proc. 1st iCC*.

[6] Gusella, R. & Zatti, S. (1989). “The accuracy of the clock synchronization achieved by TEMPO in Berkeley UNIX 4.3BSD”, *IEEE Tr. Software Engineering*, **15** (7), pp.847-852.

[7] IEEE-1588 (2002). “Standard for a precision clock synchronization protocol for networked measurement and control systems”, http://ieee1588.nist.gov/.

[8] Ramanathan, P., Shin, K.G. & Butler, R.W. (1990). “Fault-tolerant clock synchronization in distributed systems”, *IEEE Computer*, **23**(10), pp.33-42.

[9] Rodrigues, L, Guimaraes, M. & Rufino, J. (1998). “Fault-tolerant clock synchronization in CAN”, *Proc. IEEE Real-Time Systems Symposium*, Madrid, Spain, December.

[10] Rodrigues, L. & Verissimo, P. (1992). “A posteriori agreement for clock synchronisation on broadcast networks”, *Tech. Report RT/62-92*, INESC, Portugal.

[11] Schneider, F.B. (1986). “A paradigm for reliable clock synchronization”, *Tech. Report TR-86-735*, Cornell University, USA.

[12] Skoog, P. (2001). "The value of network time synchronization", *Cisco World*, May. http://www.ciscoworldmagazine.com/monthly/2001/05/time.shtml.

Dongik Lee
Dependable Real Time Systems Limited
The Innovation Centre
217 Portobello Road
Sheffield, S1 4DP
United Kingdom
Phone: +44-(0)114 224 2253
Fax: +44-(0)114 223 2301
E-mail: dongik@drts.co.uk
Website: http://www.drts.co.uk

Jeff Allan
Dependable Real Time Systems Limited
The Innovation Centre
217 Portobello Road
Sheffield, S1 4DP
United Kingdom
Phone: +44-(0)114 223 2301
Fax: +44-(0)114 223 2301
E-mail: Jeff.Allan@drts.co.uk
Website: http://www.drts.co.uk