A Brief Discussion of Security Aspects of Clock Synchronization in Networked Control Systems

Eloy Martins de Oliveira Junior, Marcelo Lopes de Oliveira e Souza
Instituto Nacional de Pesquisas Espaciais – INPE – Av. dos Astronautas, 1758, São José dos Campos, SP, Brasil

***Abstract* — Current systems such as satellites, aircrafts, automobiles, traffic controls, turbines, wind power generators and smart grids are becoming increasingly complex and/or highly integrated as prescribed by the SAE-ARP-4754 Standard. Such systems integrate computations, communications and real time controls via networks among other key architectures and technologies to form networked control systems (NCS). Such architectures and technologies usually require accurate clock synchronization among its nodes and devices for correct operation. So, any accidental or intentional fluctuation beyond a tolerance in clock synchronization can cause faults in such systems. This paper presents a brief discussion of security aspects of clock synchronization in networked control systems. This highlights how a networked control system using time triggered architectures and technologies is affected by the de-synchronization of a clock caused by an External Malicious Agent (EMA); and this includes some simulations to illustrate them. The paper concludes by suggesting some countermeasures, based on the discussions and simulations presented.**

***Keywords* — Clock Synchronization, Networked Control System, Security.**

I. INTRODUCTION

Currently, the largest trend in real time applications is to integrate computations, communications and real time controls in different levels of operation, using a large number of actuators, sensors and controllers implemented in intercommunicating processors. This can be done according diverse architectures. This paper emphasizes the time triggered architectures and technologies, where sensors, actuators, and controllers are connected via a network to form a networked control system (NCS).

These systems require predictability in the logical domain and in the temporal domain [2]. The temporal requirements may become very strict, thus demanding clock synchronization among the nodes. The most common approach to achieve this is to use a master clock with high precision and synchronized with other clocks in the nodes.

Clock synchronization is a critical part of a networked control system. It is an important topic of conception and design, mainly in time triggered technologies. In time triggered architectures and technologies, every node (or subsystem) has its own clock.

Eloy M. Oliveira Junior, eloy@dem.inpe.br; Marcelo L. O. Souza, marcelo@dem.inpe.br, Tel +55-12-32086240.
This work was supported by Course and Division of Space Mechanics and Control-DMC from the National Institute for Space Research - INPE, and CAPES of Brazil.

In normal operation, the goal of clock synchronization is to ensure that the operations follow in the correct order, so all nodes need synchronization among the clocks of the entire system. Any fluctuation (de-synchronization) beyond a tolerance in this function can cause a fault in the control system. The fluctuation of a clock may be caused by a natural cause (imperfections of clock) or by an External Malicious Agent (EMA). Consequently, there is a lot of concern, not only with the accuracy of the clocks, but with the security of the entire distributed control system.

The main imperfections of clocks are: drift, offset, fluctuation (jitter) and state error. These imperfections are caused by environmental changes such as variations in temperature and voltage, aging of crystal, in case of quartz clock, and some others reasons. However, to correctly understand and to ignore unnecessary details and focus on the essential features of the design, the abstraction of clocks is necessary. Abstractions are often applied in a hierarchical fashion, where each layer of abstraction relies on the essential features of the abstraction level below, and hides unessential details from the lower level [3]. Many models of abstractions are possible, depending on circumstances. This paper aims to show the security aspects for clock synchronization of networked control systems. To better understand the security aspects, we divided them into three main abstraction layers: physical, hardware, and software; as shown in Figure 1.

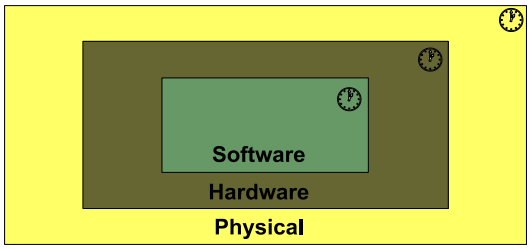

Figure 1. Main Abstraction Layers in Clock Synchronization.

Figure 1 shows the following main abstraction layers:

- Physical representation is the layer where we have physical materials and properties, *e.g.* crystal quartz clocks.
- Hardware representation is the layer where we have the oscillator mechanism(s) of clock(s).
- Software representation is the layer where we have the logical clock(s), *i.e.*, the clock(s) controlled by software.

In this paper, we make a brief discussion about security aspects of clock synchronization of networked control systems, only in the software abstraction layer. However, the software layer is dependent of the hardware and physical

layers. So, the software layer is the most susceptible to attacks by an External Malicious Agent (EMA). This paper aims to show how the clock de-synchronization affects the control of NCS, and how this de-synchronization can be caused by an EMA. To do that, in Section II, we discuss basic concepts about clocks and its imperfections, methods, architectures, and algorithms to achieve clock synchronization, and the networked control system model. In Section III we discuss the security aspects of clock synchronization, and the possible attacks to clocks by an External Malicious Agent (EMA) that can degrade networked control systems. In Section IV we show some simulations. In Section V we discuss the results. In Section VI we offer some conclusions.

II. BASIC CONCEPTS

In this section, we show the basic concepts about clock synchronization and networked control systems.

Clocks Fault Modes

A physical clock and, therefore, logical clocks, have some imperfections. These imperfections can be caused by environmental changes such as variations in temperature and voltage, aging of crystal, in case of quartz clock, or can be caused by an External Malicious Agent (EMA). Figure 2 shown the main imperfections of a clock.

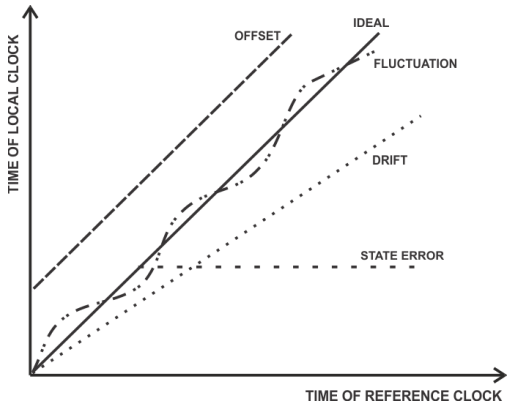

Figure 2. Main Imperfections of a Clock.

These imperfections are known as drift, initial or instantaneously offset, fluctuation and state error:

- ***Clock Drift:*** Clock Drift is when a local clock has a frequency of oscillation greater or less than another local clock and/or a reference clock; i.e. the drift is the rate of change between the two clocks.
- ***Offset:*** There are two types of offset: the initial offset is the difference between the initial times of local clocks and/or of a reference clock; the instantaneous offset is the difference between the instantaneous times of local clocks and/or of a reference clock.
- ***Fluctuation:*** The fluctuation or jitter is the uncertainty in the measurement of the clock.
- ***State Error:*** The State Error is when a local clock stops the measurement of the progression of time; i.e., the local clock stops on a fixed value. The State Error can be considered a fault or a failure.

These imperfections of a clock are impossible to eliminate. Thus, there is a need to use methods to achieve the clock synchronization within a tolerance and to minimize the effects of these errors of a clock.

There is a lot of methods for clock synchronization. These methods follow the clock synchronization architectures.

Clock Synchronization Architectures

This paper presents two of these architectures. Most methods and algorithms to perform clock synchronization are based in one of them.

1) Centralized Architecture: The centralized architecture uses a master clock, *i.e.*, the slave clocks receive periodically a time of a master clock and then the slave clocks compare and adjust their times.

Figure 3 shows this architecture.

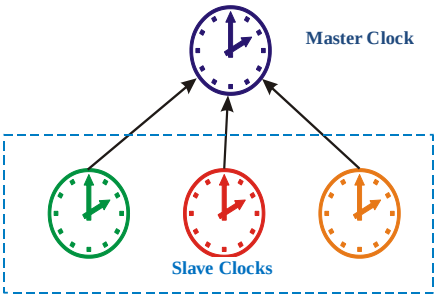

Figure 3. Centralized Architecture.

This architecture is widely used in many systems. There is an IEEE 1588-2008 Standard [4], initially used for clock synchronization over Ethernet protocols that is now migrating to other protocols and applications, such as power systems. This architecture is mostly applied in systems that make use of clock synchronization based on the GPS (Global Position System).

The master clock usually is of greater quality, to establish an accurate time base. It can be: an atomic clock, an international standard time reference (UTC – International Time Coordinated or TAI – International Atomic Time) or, in many cases, the GPS, which contains an atomic clock.

This architecture, besides not being tolerant to Byzantine errors, has the disadvantage of having a common point of failure; *i.e.*, if the master clock fails then all slave clocks lose their reference for synchronization, thus compromising the system. This disadvantage has been minimized with: 1) the use of master clocks of greater quality; 2). other efforts to minimize the problem of common point of failure; both at the expense of increasing the cost of the system. Both have increased the reliability of this architecture over the years.

2) Distributed Architecture: The distributed architecture does not use a master clock to synchronize the system. To achieve clock synchronization, this architecture creates a

global time. The global time is achieved by one mathematical equation involving a subset of the set of clocks.

This architecture is frequently used in communications via a databus, where the global time must be achieved for correct operation of network. Figure 4 shows this architecture over a databus.

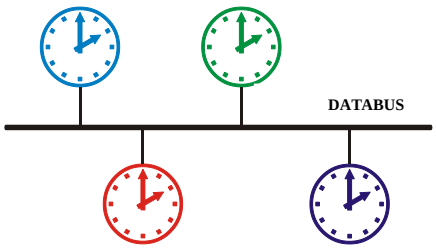

Figure 4. Distributed Architecture.

Figure 4 shows four local clocks where each of them calculates its correction. These calculations by each clock make the convergence to a global time, i.e., all clocks converge to the same time within a tolerance.

This architecture has the advantage of allowing byzantine fault tolerant clock synchronization algorithms. This increases the reliability of the system but it increases the traffic of data and adds a cost on the precision achieved among the set of clocks. Kopetz [5] presents a table of additional costs on precision due to the use of byzantine fault tolerance in an averaging clock synchronization algorithm.

The FTM (Fault-Tolerant Mid-Point) Algorithm, created by Lundelius *et al.* [6], is an example of a byzantine fault tolerant algorithm for a distributed architecture. To ensure that all nodes have a consistent view of time, the re-synchronization of clocks is needed regularly (periodically). The consistent view of time by all clocks is called a (virtual) global clock. For this, the algorithm follows a logical sequence. Each node applies this sequence with the aim of reaching a correction term.

Networked Control Systems

Current systems such as satellites, aircrafts, automobiles, traffic controls, turbines, wind power generators and smart grids are becoming increasingly complex and/or highly integrated as prescribed by the SAE-ARP-4754 Standard. Such systems integrate computations, communications and real time controls via networks among other key architectures and technologies to form networked control systems (NCS). Such architectures and technologies usually require accurate clock synchronization among its nodes and devices for correct operation. The introduction of communication lines brings a lot advantages, and disadvantages. This paper uses a NCS with a TDMA (Time Triggered Multiple Access) network to discuss security aspects of clock synchronization, as shown below.

III. SECURITY ASPECTS

Security becomes an importance aspect when clock synchronization is used in commercial applications, in public networks or even in critical control applications which make use of a network.

The security aspects are becoming an important research area. In previous sections, we introduced the main imperfections of clocks. In general, the problem of clock synchronization of logical clocks caused by natural imperfections is solved by algorithms. However, it is obvious that the overall functionality of those systems can be degraded or even disabled if the mechanism of synchronization of clocks is attacked [7].

The attacks to the mechanism of synchronization of clocks depend of the clock synchronization architecture. In general, each architecture and system have their own vulnerability list, although some attacks are common for many clock synchronization architectures, as detailed below.

Clock Attacks in Centralized Architectures

The centralized architecture is widely used in many systems. There are some researches in the literature about the security aspects of systems using the IEEE 1588-2008 Standard [4], such as in [7]-[9].

Table 1, extracted from [7], shows a list of attacks to clock synchronization and the results of these attacks.

TABLE I – LIST OF ATTACKS TO CLOCK SYNCHRONIZATION

	Attack	Result of Attack
1	Denial of service	no service available
2	Byzantine master	complete loss of control
3	Interruption of control loop	deviation determined by precision of local clock
4	Removal of packets from control loop	deviation determined by precision of local clock
5	Packet manipulation	complete loss of control
6	Packet insertion	offset up to sync cycle depending on implementation
7	Selective packet delay	offset up to sync cycle

Source: [7]

In the list of Table 1, we are more interested in the attacks that result in a complete loss of control or instability. According to Table 1, the 2) byzantine master, and 5) packet manipulation, can cause it directly; and the 3) interruption of control loop, and 4) removal of packets from control loop, can cause it indirectly. Let us discuss them briefly.

1) Byzantine attack: In the centralized architecture, we have a master clock. This means that all local clocks (slave clocks) track the time of a master clock; see Figure 3. If the master clock suffers any fluctuation, then all local clocks are affected. If the External Malicious Agent (EMA) can manipulate or even degrade the master clock, then the clock synchronization process is impaired. The nodes lose the time reference and, in consequence, this causes the degradation of control.

2) Packet manipulation: The clock synchronization via software is dependent of time data exchanges among the local clocks and the master clock. These data are encapsulated in packets to transmit the messages of time. These data, encrypted or not, can be intercepted and manipulated by an

External Malicious Agent (EMA). This manipulation may cause a loss of clock synchronization and, in consequence, this causes the degradation of control.

3) Interruption of control loop: This attack aims to interrupt the clock synchronization process, *i.e.*, this attack interrupt the time data exchange between the clocks. The clocks begin working in open loop and natural fluctuations of clocks induce a loss of synchronization. This attack can cause an indirect degradation of control, if the node of control loses its clock synchronization.

4) Removal of packets of control loop: This attack is similar of interruption of control loop. The removal of packets causes an interruption of clock synchronization, and the effects are the same of the previously discussed attack.

Clock Attacks in Distributed Architectures

The applications that make use of this architecture are growing. This architecture does not use a master clock in the clock synchronization process. An example of a system which uses this architecture is the Time Triggered Protocol (TTP) [10]. There, clock synchronization is assured by the Byzantine Theorem [11] expressed by Equation (1):

$$n \geq 3f + 1 \quad (1)$$

where, *n* is the number of clocks in the system; and *f* the number of clocks with errors. Through this rule, it is possible to achieve clock synchronization even in the presence of (*f*) errors.

The distributed architecture is vulnerable to the same attacks mentioned above in the centralized architecture, such as: 1) byzantine attack, 2) packet manipulation, 3) interruption of control loop, and 4) removal of packets of control loop. The difference is that, in this architecture, the attacks do not focus on the master clock, but focus on the hypotheses of the Byzantine Theorem (1); *i.e.*, they aim to make them invalid. If this happens, then the clock synchronization process is impaired, and the virtual global clock is not maintained.

There are other possibilities for improving the security in clock synchronization. For each system, an investigation on the means to reach higher security is necessary.

IV. DESIGN OF SIMULATIONS

In networked control systems, mainly in real time ones, using a protocol that requires clock synchronization is crucial to have good clock synchronization. To exemplify how the imperfections (caused by Nature or by an External Malicious Agent (EMA)) of clock can cause a degradation of control, we studied by means of simulation the case of a networked control system using a TDMA (Time Division Multiple Access) philosophy for the communication protocol.

For this simulation, we used the TrueTime/Matlab/Simulink environment [12]. We simulated two sets of controls, *i.e.*, a system with two control loops connected by a common databus (TDMA). The sensors, actuators/plants and controllers were connected via the databus. The controller used was a PID (Proportional, Integral and Derivative). The actuator/plant was a second order marginally stable continuous time system, according to the following transfer function:

$$G(s) = \frac{1000}{s(s+1)} \quad (2)$$

The controller and sensor nodes had logical clocks given by the virtual computer of the TrueTime Kernel; and they used the databus to exchange data and time measurements between them. The actuators/plant used the databus only to receive the control data. Each control node implemented a periodic control task and a periodic clock synchronization task. Each sensor node implemented a periodic task for sending the measured data to the controller; and a periodic task for synchronizing the clock. Each actuator/plant was activated by events when the control task arrived in the actuator by the databus. All nodes had an interruption caused by data arrived from the databus. The reference time is the virtual time given by the logical clock of the Matlab/Simulink environment. In the TrueTime Kernel it is possible to manage the virtual clocks. So, it is possible to insert faults (imperfections) and correct them. The EMA (External Malicious Agent) node implemented one task for changing, maliciously, the virtual clock in Sensor 1. The model of the simulated NCS is given at Figure 5:

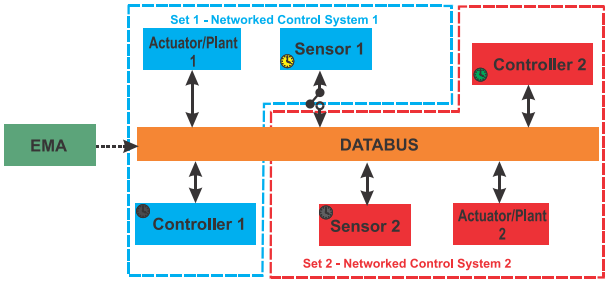

Figure 5. Networked Control System Model.

For the clock synchronization, we used a FTM (Fault-Tolerant Mid-Point) algorithm, also known as Welch-Lynch algorithm [6]. It provides fault tolerance for byzantine clock synchronization of distributed systems. To ensure that all nodes have a consistent view of time (global time) we need to re-synchronize the clocks regularly (periodically). For this, the algorithm follows a logical sequence. Each node applies this sequence with the objective of reaching a correction term. With this correction term, the deviations caused by the drift of the clocks are adjusted so that all system clocks are within a certain precision.

V. RESULTS

We simulated two control subsystems sharing the same databus as shown in Figure 5. In blue, we have control loop

set 1 and in red, we have control loop set 2. The sensors, actuators/plants and controllers are connected via the databus.

The EMA, in green, represents the External Malicious Agent, which was responsible by inserting a sum of 0.1 second in the time measurement of sensor 1 at instant 0.1 second. This attack is called packet manipulation. The objective of the simulation is to show how this attack: 1) directly degrades the synchronization between Controller 1, Sensor 1, Controller 2 and Sensor 2 even with all using a FTM algorithm; and 2) indirectly degrades the system step response.

Figure 6 shows the timeline of all clocks in the databus, and Figure 7 shows a detail of Figure 6. The yellow line represents the time measurement of Sensor 1, under attack.

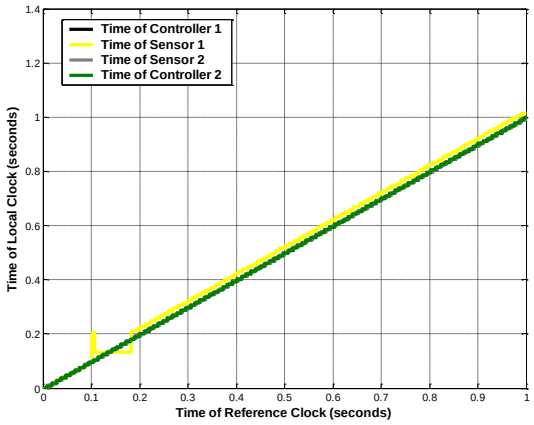

Figure 6. Timeline of set of clocks.

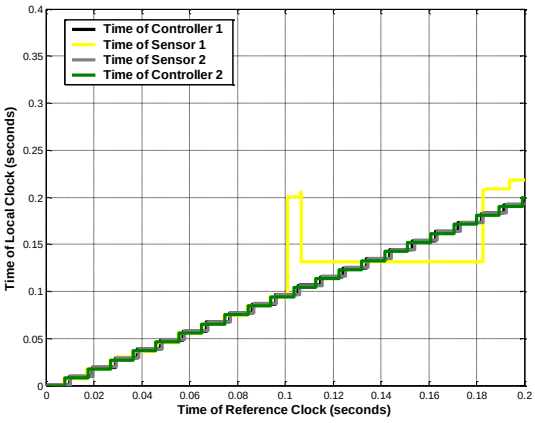

Figure 7. Detail of timeline of set of clocks.

We can observe in Figures 6 and 7, at the instant 0.1 second, that the time measurement of sensor 1 was increased of 0.1 second. This error is corrected by the FTM algorithm near 0.2 second. Even so, this correction affects the control.

Figure 8 show the PID control of sets 1 and 2. In blue is the PID control 1, and in red is the PID control 2. We observe that control 1, with error in sensor 1, is affected after the instant 0.1 second. Control 2, without error, keeps the same step response. We can even observe a gap in control in the blue line between instants 0.1 and 0.2 second.

This happens because then the clock of sensor 1 is wrong. So, sensor 1 thinks it is ahead of time. The transmission of sensor 1 stops until it recovers the clock synchronization. Due to this, the control loop is temporarily opened, as we can see between instants 0.1 and 0.2 second. Figure 9 shows this effect on the system step responses.

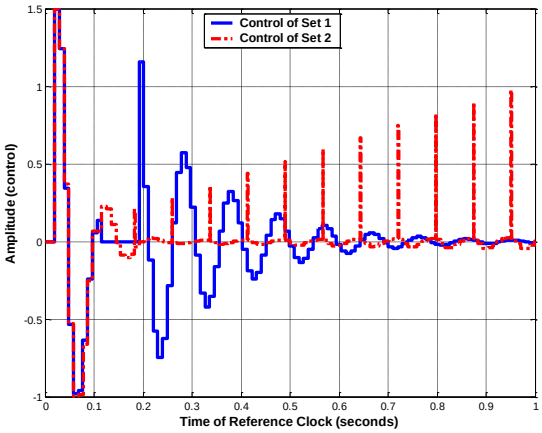

Figure 8. Control Laws.

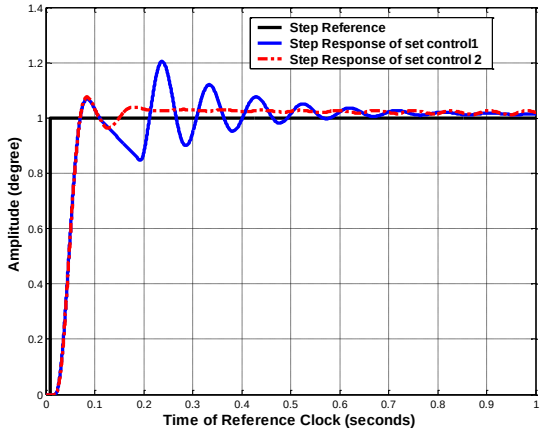

Figure 9. System Step Responses.

Between instants 0.1 and 0.2 second, the set of control 1 operates in open-loop. So, until the system recovers the clock synchronization, the system remains marginally stable. Figure 10 shows the difference between the step responses of set of control 1, in blue, and set of control 2, in red.

To compare how the control is degraded by the attack on the clock of sensor 1, Fig. 11 shows the ideal difference between step response of set of control 1, in blue, and set of control 2, in red, without attack to sensor 1.

Summarizing, this attack was simple and introduced an offset error in the clock of sensor 1. This caused a temporary opening of control loop 1 in a TDMA network. The effect was not very destructive for the controlled system because of the FTM algorithm. The algorithm was a countermeasure for this attack, and recovered the synchronization before major damages happened.

This example showed clearly: 1) the degradation of control loop 1 induced by an attack to the clock of sensor 1;

2) the correction by the FTM algorithm reducing the bad effects on the system.

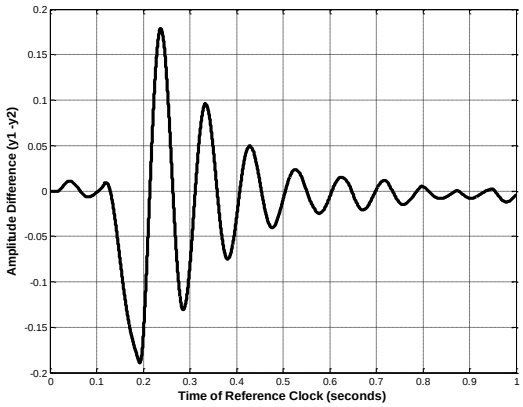

Figure 10. Difference of Step Responses of Controls 1under attack and 2.

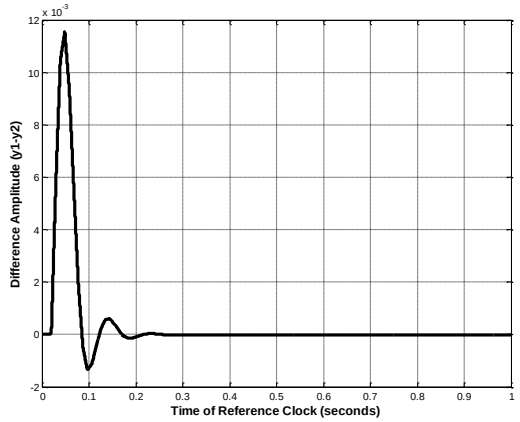

Figure 11. Ideal Difference of Step Responses of Control 1 and Control 2.

VI. CONCLUSIONS

The security aspects of clock synchronization are becoming important for research and development of complex and integrated systems. Technically, the problem of clock synchronization caused by its natural imperfections can be solved within a tolerance by algorithms and methods. But socially, the overall functionality of those systems can be degraded or even disabled if the synchronization of the clocks is attacked.

The attacks of clocks can be caused by numerous ways, such as byzantine attack, packet manipulation, interruption of control loop, and removal of packets of control loop. These attacks aim to manipulate the times of clocks of systems, and trick them. The clock attacks depend of the clock synchronization architecture. So, during design, each system shall be analyzed: 1) to establish its own vulnerability list; and 2) to choose its security mechanism to be implemented to minimize these vulnerabilities.

In a NCS with TDMA communication, the clock synchronization is crucial to achieve logical and temporal correctness. If the temporal correctness is attacked, then the overall functionality is degraded.

To exemplify the attack to clocks, we simulated an attack to a NCS with 2 loops. The preliminary results based on that case study suggest that: 1) the packet manipulation attack to clock of sensor in a NCS affects the clock synchronization, and then, the communication and control; 2) in the TDMA protocol, an attack to the time management and/or clock synchronization opens the control loop temporarily; 3) the NCS needs countermeasures for attacks to the clock synchronization; 4) The FTM algorithm used prevents the worst effects on the control system; and it is good countermeasure. Others are suggested below.

Suggestions of Countermeasures

To reach the required security goals, we suggest installing various countermeasures on various levels [7], as:

- Cryptography of time messages transmitted, i.e., introduction of cryptographed measures of time in the clock synchronization process;
- Firewalls to protect the transmission media;
- FDIR means to: Detect an error in the master clock, Isolate and Identify the error in the master clock, and Reconfigure the architecture by choosing another master clock;
- Algorithms to identify and prevent any abrupt change in time;
- Security measures by hardware timestamp, i.e., include a new layer in the clock synchronization process.

REFERENCES

[1] SAE, “Certification Considerations for Highly-Integrated or Complex Aircraft Systems,” Aerospace Recommend Practice ARP-4754, SAE, Nov. 1996.

[2] J. A. Stankovic, “Misconceptions about real-time computing – a serious problem for next-generation systems”, IEEE Computer, vol. 21, no.10, pp.10-19, October 1988.

[3] D.G , Messerschmitt, “Synchronization in digital system design”, IEEE Journal on Selected Areas in Communications, vol. 8, p. 1404, October 1990.

[4] 1588-2008 - IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, c1 – 269, July, 2008.

[5] H. Kopetz, “Real-Time Systems: Design for Distributed Embedded Applications”, 1 ed., Kluwer Academic Publishers, 1997.

[6] J. Lundelius, and N. Lynch, “A new fault-tolerant algorithm for clock synchronization”, Third Annual ACM Symposium on Principles of Distributed Computing, Vancouver, Canada, 1984.

[7] G. Gaderer, A. Treytl, T. Sauter, “Security aspects for IEEE 1588 based clock synchronization protocols”, IEEE International Workshop on Factory Communication Systems - WFCS 2006, Torino, Italy, September 2006.

[8] J.-C. Tournier, O. Goerlitz, “Strategies to secure the IEEE 1588 protocol in digital substation automation”, Fourth International Conference on Critical Infrastructures, CRIS 2009, Sweden, Linköping, April 2009.

[9] A. Treytl, G. Gaderer, P. Loschmidt, N. Kerö, “Investigations on security aspects in clock synchronized industrial Eethernet”, 38th Annual Precise Time and Time Interval (PTTI) Application and Planning Meeting, Washington, USA, December, 2006.

[10] TTTech Computertechnik. “Time-Triggered Protocol TTP/C high-level specification document protocol”, version 1.1. Ed. 1.4.3. Vienna, 2003. (D–032–S–10–028).

[11] L. Lamport, R. Shostak, M., Pease. “The Byzantine Generals Problem”, ACM Transactions on Programming Languages and Systems, Vol. 4, No. 3, July 1982, Pages 382-401, 1982.

[12] M. Ohlin, D. Henriksson, A. Cervin, “TrueTime 1.5 – Reference Manual”, Department of Automatic Control, Lund University, Sweden, Jan. 2007.