Survivability of SCADA Control Loop

José M. Camacho , *Universiteit Twente*

Abstract—The endorsement of information technologies for critical infrastructures control introduces new threats in their security and surveillance. Along with certain level of protection against attacks, it is desirable for critical processes to survive even if they succeed. A stochastic Petri Nets-based model of a SCADA control loop is presented in this paper, modelling both, the system functioning and attacks. Together with the model, a distribution-based method to assess survivability is introduced and applied to the depicted model.

Index Terms—Survivability, SCADA Control Loop, Stochastic Petri Nets.

I. INTRODUCTION

THE development of information and communication technologies in the latest years has led to an increasing amount of novel applications. Even traditional proprietary systems such as *supervisory control and data acquisition* (hereafter SCADA [1]) of critical infrastructures have shifted away from expensive vendor-specific solutions to commodity off-the-shelf software and hardware products.

In spite of such a change brings about many benefits like systems interoperability, costs reduction and faster personnel training, as a direct consequence of general purpose technologies endorsement, vulnerabilities and operational problems are automatically inherited by target systems (e.g. SCADA).

Those new potential threats are a major concern nowadays since, in contrast to other communication networks as the Internet, the disruption of service in a SCADA is critical and can turn out to be harmful to the physical system under control and to people depending on it. Imagine for instance, the consequences of a misbehavior in a system controlling a *dump* or a *dike*. Therefore, it is important to guarantee a certain quality of service under any condition.

Research and investments have been done to that extent. Yet, literature survey yields the conclusion that most efforts in dependability focus on systems protection against random failures (namely *reliability*) [2], [7] . When it comes to securing systems, prevention in the face of malicious attacks is the priority [6]. However, little can be found regarding *survivability*.

Hereby, survivability is understood *as the capability of a system to fulfill its mission, in a timely manner, in the presence of attacks, failures, or accidents* [3], [5], [12]. Graceful and quick recovery of system response is crucial in many cases. In addition to scarce studies on survivability, most dependability models in related work are rather simplistic [7], [8].

The main idea of this paper is to model specifics of the SCADA control loop (see Fig.1) and the possible impact on its performance of several attacks. Presented results are aimed to provide insights into the system's misbehaviour and importantly, how the control loop recovers in the presence of those attacks.
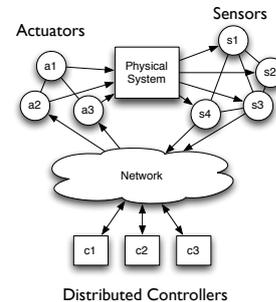

Fig. 1.   Schematic Control-Loop

In order to fulfil those goals, *Stochastic Petri Nets* (SPNs) are chosen as modelling technique. SPNs are a widely used graphical and analytical tool. Roughly speaking, SPNs consist of *places* and *transitions* interconnected by *arcs*. The state of the model is defined by the number of *tokens* held in places. Each tokens distribution is known as a *marking*. Transitions fire at a given rate when enabled, moving tokens from one place to another. Detailed documentation on SPNs can be found in [9], [10], [11]. Measurements from the SPNs models were obtained by using a software with support for SPNs called SMART [15].

This paper's contribution is threefold. Firstly, a model of the entire SCADA control loop using SPNs is presented in Section II. Secondly, a method to assess survivability by means of common analysis techniques of SPNs is described in Section III-A. Finally, results and conclusions are discussed in Sections III-C, III-D and IV.

II. SCADA CONTROL-LOOP

A. Overview

SCADA systems are widely deployed and mostly follow the structure depicted in Fig.1. Information is collected from the physical system by the sensors (e.g. temperature read). Acquired data is digitalised and sent through a communication network. Upon arrival to the controller, that data is processed and the control logic makes a decision. Depending on that decision, a controller may wait for more incoming information from the sensors or it may order some actuators to perform required actions (e.g. raise the amount of water in the cooling system).

Despite their clearly defined role, these three differentiated parts can be implemented in many ways (e.g. network can be either wired or a wireless [14]). Modelling will abstract from implementation details and focus on behavioural aspects. Upcoming sections introduce a model for the main parts of the system along with possible points of failure.
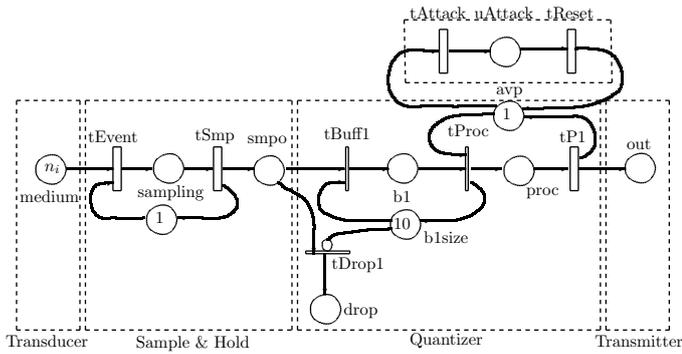
Fig. 2.   SPN Model of a Generic Sensor.

Fig. 3.   SPN Model of a Communication Channel.

Fig. 4.   SPN Model of a SCADA Controller.

B. Sensor Model

A typical sensor is made of four clearly differentiated parts, see Fig.2. A transducer block which generates an electric signal related to the measured magnitude according to a sensitivity law [4] and *conditions* that signal (e.g. noise filtering). Next step is to pass such a signal through an analogue to digital converter (i.e. A/D converter) which in turn has two stages: a *sample and hold* circuit that picks up taps (sampled values) from the signal and a *quantizer*, which processes them to determine their digital representation. Last block is typically a transmission block that reports the measured values through the communication network.

The sensor model in Fig.2 is *excited* by changes in its measured magnitude, those are modelled as the timed transition, `tEvent`. The place `sampling` represents the mentioned sample and hold circuit, from which only one tap at a time is generated. The A/D converter has a buffer for up to 10 taps, which can be queued at `b1` before they are quantified by the processor.

The marking of place `avp` dictates whether the processor is available or not. An attack can be easily modelled as follows: the token in `avp` is moved to `uAttack` as soon as the `tAttack` transition fires. Thus, if no token is in `avp`, the system stops processing samples. This mainly represent attacks that halt the processing unit but do not affect the information already saved in the system.

After quantization is finished (i.e. `tP1` has fired), the digital sample is relayed to the transmission system, which sends the sample to the control unit throughout the communication network.

C. Channel Model

Once samples are digitalised, they are sent to the controller in order to determine whether actions must be taken to the actuators or not. To that extent, sensors feature communication capabilities. That is modelled in Fig.3 as a finite transmission buffer (`buffer2`) bounded to #b2size packets (tokens). Packets are transmitted as `tTrx` transition fires.

Afterwards, packets travel through the channel and two situations may occur:

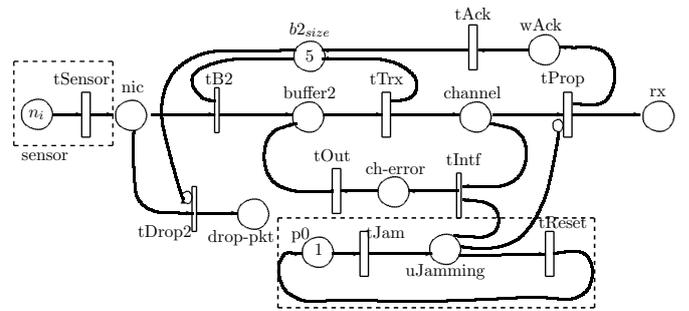1) A packet can be sent successfully, hence after some propagation time (modelled as `tProp` transition) it
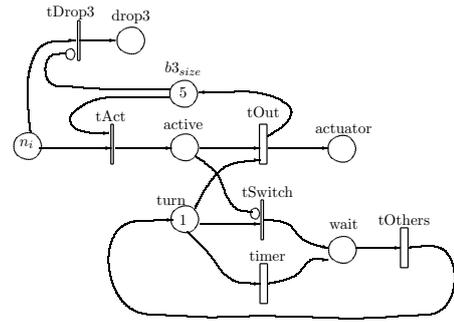
arrives to the receiver, which acknowledges the packet after `tAck` fires.

2) In contrast, if some interference (malicious or not) destroys the information crossing the medium, i.e. there is a token in `uJamming` place, then, `tIntf` fires immediately and eventually the sender notices the timeout of the packet (i.e. transition `tOut` fires). A timeout indicates to the transmitter that a packet has not been acknowledged. The packet is then put back in the queue to be re-transmitted.

Buffer space is freed only when packets have arrived and have been acknowledged.

D. Control Unit and Actuators

For completeness, the SCADA controller has been modelled (see Fig.4), even though survivability of the controller is left as future work. The controller acts as a polling station which follows a Round Robin serving policy. As soon as no tokens are left in the place `active` or the transition `timer` fires, the system starts processing samples coming from other sensors. If `tOut` fires, a command is sent to the actuators.

III. SURVIVABILITY EVALUATION

A. Methodology

In the following, we asses whether the presented models are survivable in presence of attacks. Our approach is based on the fact that the models will eventually reach a steady state if no attacks occur. Thereafter, the state probability distribution remains steady.

Sensor Rates		Channel Rates	
tEvent	1	tSensor	0.27
tSmp	0.2	tTrx	0.012
tP1	0.27	tProp	0.005
tReset	$15 - 45 \cdot 10^{-5}$	tAck	0.002
		tOut	0.002
		tRJam	$25 \cdot 10^{-5}$

To our method, attacks only happen when the system is already in the steady state. The effects of an attack in the model are an immediate change in the marking (e.g. some places flush their tokens immediately) and the state probability distribution goes on a transient state.

If the recovery mechanism is in place, intuitively the state probability distribution of the model should evolve along the time towards the distribution in the steady state (i.e. when no attack has occurred). Hence, a system is said to be survivable if its state probability distribution converges to the distribution without attack.

Since it is not possible to know a priori in which marking the model is left after the attack takes place, the former survivability assessment must be done for any possible existing marking. In addition, as introduced in Section II-B, only the tokens held in places that model *buffers* are not removed by the attack. The latter constrains the amount of existing markings after attack. Moreover, for simplicity in the measure, model's state probability distribution is measured only at those places, in other words, the evaluation is based on the probability distribution of the number of tokens in the buffers.

B. Evaluated Metrics

In this paper two metrics are analysed,

- *Time to converge*: If $f_i(t, n, n_0)$ is defined as the function that represents the evolution over time (t) of the probability of having n tokens in the buffer i given that when the attack occurs there are already n_0 tokens in, then the *time to converge* is the time interval that $f_i(t, n, n_0)$ needs to meet the value of the probability of n tokens in buffer i in steady state, with 0.5% tolerance and $0 \le n, n_0 \le size(\text{buffer } i)$.
- *Time to recovery*: is defined as the maximum among the times to converge for $0 \le n \le size(\text{buffer } i)$, tokens in the buffer, for all possible initial markings n_0.

C. Analysis of the Sensor

1) Sensor Performance without Attack: Fig.5 illustrates the evolution of the probability distribution of the number of tokens in b1 when the system has just started to work and zero tokens were there at the beginning. For instance, the probability of having 0 tokens in b1 is the graph marked with a 0 on the right side. As it can be observed this probability distribution stabilises after the system has been running for about 0.15 *sec*. Other experiments not shown here yield
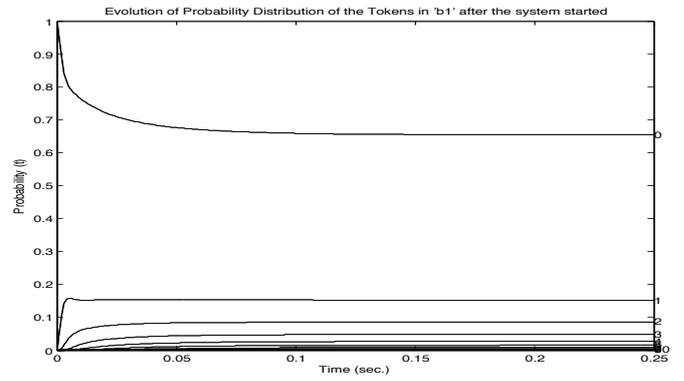

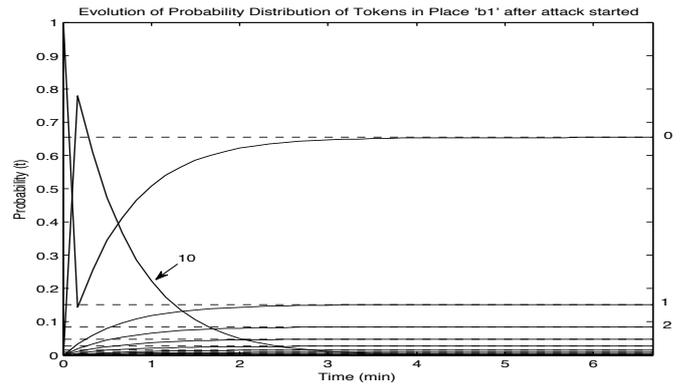Fig. 5. PDF of tokens in b1 evolving in time for a scenario without attacks during system startup.

Fig. 6. Transient analysis of the system after attack. Shows how the probability distribution of tokens in b1 converges to the values without attack.
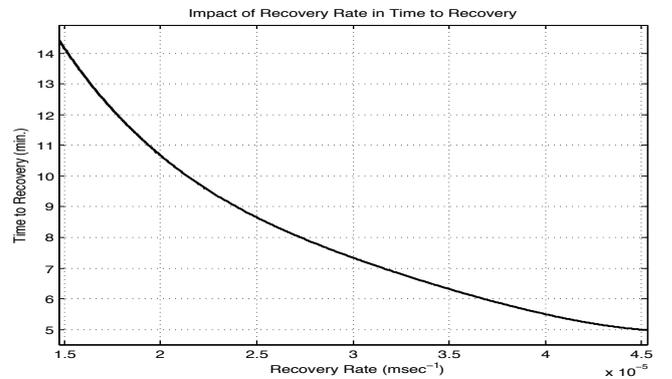

Fig. 7. Recovery Rate vs. Time to Recovery

that the steady state distribution is independent of the initial number of tokens in b1. The values for the firing rates can be found in Table I.

2) Sensor Performance under Attack: Dashed lines in Fig.6 depict the target distribution as in Fig.5. Continuous lines show the transient probabilities, for the case of zero tokens in the buffer b1 when the attack starts (e.g. most likely case). Probability distribution converges asymptotically towards the target distribution. In agreement with the notation introduced before, the curve denoted with a 0 on the right side corresponds to the defined function $f_1(t, 0, 0)$, the one denoted as 1 to $f_1(t, 1, 0)$

and so forth.

Time to converge (and therefore time to recovery) of those functions was measured for different existing markings $(f_1(t, n, 0), f_1(t, n, 1), \dots)$. For the sensor, results shown that the time to recovery of the system is, in this case, independent of the existing marking, with value $8.7\,min$.

Further, this time to recovery is always given by the time to converge of the probability curve for $n = 10$. This is consistent with the evolution shown in Fig.6 since the probability curve for $n = 10$ rises dramatically right after the attack is performed. The latter is due to the system is not processing samples before the processor is reset, hence it is more and more likely to have a full buffer until `tReset` is likely to fire.

The role of `tReset` in the survivability of the system leads to another set of experiments whose outcome is displayed in Fig.7. This graph shows the dependency between the time to recovery of the system and the reset rate of the processors. The faster the processor is back to work, the faster the recovery. On the other hand, if the reset rate is too slow, the time to recovery rises asymptotically towards infinity, so to speak, the system is not able to survive to the attack in time.

D. Analysis of the Channel

A similar analysis was done for the channel. Time to recovery was measured for different markings n_0 in the places `buffer2`, `wAck` and `ch-error` (see Fig.3) upon attack. In contrast to the sensor case, time to recovery varies depending on `b2`'s marking after the attack. Results are provided as a cumulative distribution function which represent $Pr\{t_r \leq t\}$, i.e. the probability of the time to recover (denoted as t_r) to be quicker than a time t. The curve is represented in Fig.8.

E. Limitations of this Approach

The SCADA control loop presented here is a detailed model of the entire system. A potential limitation is due to the fact that models are very detailed, so that the underlying state space becomes too large quickly. Putting the depicted three parts together and only one attack generates already an underlying CTMC of $652,288$ states. The addition of simultaneous attacks or redundancy will increase this amount exponentially, turning the models to be highly impractical.

Furthermore, SMART does only support exponential distribution of firing rates. Therefore, consequences of the attack in the system have to be assumed as immediate. In other words, analysis of the system degradation when is working normally and an attack starts cannot be carried out with this tool.

IV. CONCLUSIONS

A detailed model of the schematic SCADA control loop has been analysed. The level of detail present in this models allows to tune and evaluate the effect of different parameters in the performance of the system and in its survivability in particular.

The introduced method for survivability assessment provides a straightforward way of evaluating whether a system is able to recover from a given attack.
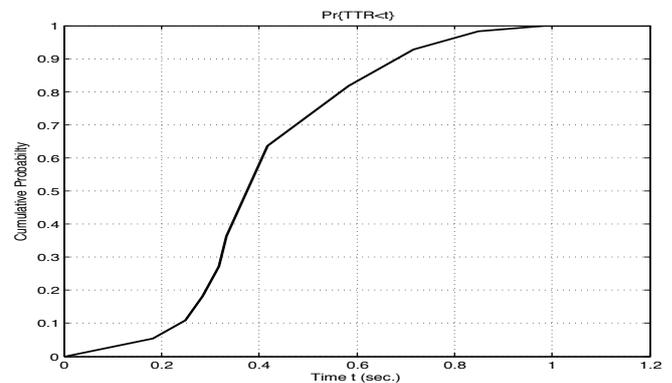

Fig. 8.   Cumulative Probability Distribution of Time to Recovery

Only hard attacks have been analysed in this paper, soft attacks are left for future work. Another interesting potential research lines for the future work are the addition of multiple processors and transmitters and the impact of combined attacks in the model. Furthermore, modelling of attacks over the controller is also pending.

REFERENCES

[1] K. Stouffer, J. Falco, K. Kent, *Guide to Supervisory Control and Data Acquisition (SCADA) and Industrial Control Systems Security.* Recommendations of the National Institute of Standards and Technology, Special Publication 800-82, Initial Public Draft.

[2] A.A. Cárdenas, S. Amin, S. Sastry, *Secure Control: Towards Survivable Cyber-Physical Systems.* The 28th International Conference on Distributed Computing System Workshop.

[3] L. Cloth and B.R. Haverkort, *Model Checking for Survivability*, Proceedings of the Second International Conference on the Quantitative Evaluation of Systems (QUEST'05).

[4] J. Fraden, *Handbook of Modern Sensors*, AIP Press.

[5] N.R. Mead, R.J. Ellison, R.C. Linger, T. Longstaff, J. McHugh, *Survivable Network Analysis Method*, Technical Report CMU/SEI-2000-TR-013, ESC-2000-TR-013, Carnegie Mellon Software Engineering Institute, September 2000.

[6] C.W. Ten, C.C. Liu, G.Manimaran, *Vulnerability Assessment of Cybersecurity for SCADA Systems*, IEEE TRANSACTIONS ON POWER SYSTEMS, VOL. 23, NO. 4, NOVEMBER 2008.

[7] H. Yu and S. Lloyd, *Petri Net-based Closed-Loop Control and On-Line Scheduling of the Batch Process Plant*, 1106 UKACC international Conference on CONTROL '98, 1-4 September 1998, Conference Publication No. 455, 0 IEEE, 1998.

[8] F. Sheldon, T. Potok, A. Krings and P. Oman, *Critical Energy Infrastructure Survivability, inherent limitations, obstacles, and mitigation strategies.* International Journal of Power and Energy Systems, PowerCON Special Issue 2004.

[9] B.R. Haverkort, *Performance of computer communication systems.* Wiley, New York, 1998. ISBN 047 1972282. 490 pages.

[10] A. Bobbio, *System Modelling with Petri Nets*, Istituto Elettrotecnico Nazionale Galileo Ferraris Strada del le Cacce 91, 10135 Torino, Italy, from System Reliability Assessment, Kluwer p.c. pp 102-143 (1990).

[11] M.V. Iordache, P.J. Antsaklis, *Supervisory Control of Concurrent Systems, A Petri Net Structural Approach*, Birkhuser Press.

[12] J.C. Knight and K.J. Sullivan, *On the Definition of Survivability*, Department of Computer Science, University of Virginia, December 2000.

[13] C.L. Su and Y.C. Chang, *A SCADA System Reliability Evaluation Considering Performance Requirement*, 2004 International Conference on Power System Technology - POWERCON 2004, Singapore, 21-24 November 2004.

[14] Rockwell Automation, *SCADA System Selection Guide*, June 2009.

[15] G. Ciardo, A.S. Miner, *SMART: The Stochastic Model checking Analyzer for Reliability and Timing*, pp.338-339, The Quantitative Evaluation of Systems, First International Conference on (QEST'04), 2004.