COMPUTATIONAL INTELLIGENCE IN THE TIME OF CYBER-PHYSICAL SYSTEMS AND THE INTERNET-OF-THINGS

Cesare Alippi, Seiichi Ozawa

Abstract The emergence of non-trivial embedded sensor units and cyber-physical systems and the Internet of Things has made possible the design and implementation of sophisticated applications where large amounts of real-time data are collected, possibly to constitute a big data picture as time passes. Within this framework, intelligence mechanisms based on machine learning, neural networks and brain computing approaches play a key role to provide systems with advanced functionalities. Intelligent mechanisms are needed to guarantee appropriate performances within an evolving, time variant environment, optimally harvest the available and manage the residual energy, reduce the energy consumption of the whole system, identify and mitigate occurrence of faults, provide shields against cyber-attacks.

The chapter introduces the above aspects of intelligence, whose functionalities are needed to boost the next generation of cyber-physical and Internet of Things applications, smart world generation whose footprint is already around us.

1. Introduction

Advances in embedded systems and communication technologies and the availability of low-cost sensors have paved the way to a pervasive presence of distributed applications in our everyday lives as well as in diversified segments of the market. In this direction, Cyber-Physical Systems (CPS) i.e., hardware/software systems interacting with the physical world are providing the technological framework to support such epochal shift in the human-machine interaction.
A cyber-physical system is typically composed of a network of heterogeneous units strongly interacting with the physical environment they are deployed in. In CPS individual units interact with the physical world, creating the basis for "smart solutions" which revolutionize scenarios from health to industry, from workplace to home, from transportation to entertainment, ultimately leading to an enhanced quality of life. Designing such systems means actively participating in a new "digital revolution" that enables augmented interaction with the real world.

Cyber-physical systems are present at home, at work and provide the core technologies to design smart homes, buildings and cities, enable the Internet of Things (IoT), support smart energy production, environmental protection, precise agricul-

Cesare Alippi
Politecnico di Milano, Italy
Università della Svizzera Italiana, Switzerland
Cesare.Alippi@polimi.it

Seiichi Ozawa
Kobe University, Japan
ozawasei@kobe-u.ac.jp

ture, management and metering, facilitate smart transportation and healthcare just to provide a very concise list. The expected evolution of the field, as also perceived by the industry [1, 2], will focus on the integration of hardware and software technologies to support application reconfiguration, enable autonomous operations, make native the access to the Internet, and extend the usage and operational models by introducing intelligent resources and application management mechanisms.

We all agree that addressing fundamental architectural, technological and standards challenges, e.g., the energy consumption of the transistor, the communication level and the IPV(6) protocol, will improve the units efficiency and networking ability [3]. However, fundamental advances in highly performing hardware per se will not be enough to drastically change the way embedded applications impact on our lives. In fact, we should also design methodologies that, by adaptively optimizing the use of existing embedded resources, provide the application with intelligent functionalities granting *adaptation abilities* to environmental changes, adaptive *fault detection and mitigation facilities* and sophisticated adaptive *energy-aware modalities* to prolong the system lifetime [4]. Intelligence is also needed to prevent attacks from malicious software (malware) designed to steal sensitive information from users, take control of systems, impair or even disable the functionalities of the attacked devices, distort money from users and so on. Recently, malware Mirai was shown to be able to take control of IoT devices and carry out a major cyber-attack [5]; this represent a vulnerability issue that has to be promptly addressed.

Current research addresses intelligent aspects mostly as independent research lines either without any functionality harmonization effort or with little emphasis on how to integrate the different –challenging– facets of these fields within a solid framework. Moreover, not rarely, strong assumptions are made to make derivations amenable at the cost of a high loss in performance, efficiency –and sometimes credibility– whenever real world applications are taken into account. In particular we assume *infinite energy availability*, in the sense that energy and power consumption is not an issue, *stationarity/time invariance*, implying that the process generating sensor data (the physical world and the interaction with the sensor transducer) is not evolving with time, *correct data availability* claiming that acquired data are complete and correct, and *secure* operations assuming that cyber-attacks are not carried out. By relaxing above assumptions we enter in a new realm requesting presence of intelligence in all computational architectures; these aspects are addressed in subsequent sections.

2. System architecture

The physical ICT architecture we consider here reflects those considered for CPS and IoT [6]. As such, it is a very variegated one composed, in its most gen-

eral form, of heterogeneous hardware and software platforms. End point units of the (internet) connection communicate with servers (possibly also acting as gateways) in a star, field bus, or general topology depending on the particular application at hand. Computational complexity and hardware resources (e.g., memory capacity, energy availability) are application-specific, with units that, not rarely, are operating system-free. In other cases, units possess a simple operating system (e.g., RTOS), a more complex one (e.g., Android) or an operating system specifically developed for limited-resource devices, such as Contiki [7], ARMmbeb [8] or, again, specifically targeted to IoT such as the Google Android Things [9].

An end unit can mount application-specific sensors and/or actuators, with the interesting case where humans can act both as sensors and actuators. Fog and cloud computing processing architectures can be elements of the overall architecture: the final architectural decision depends on the expected and the maximum response time tolerated by the application.

Where intelligence should be located in the architecture? This complex question receives a very simple answer: it depends on the energy availability and the computational and hardware/software resources needed by the application and the intelligent functionalities to carry out their tasks. Once intelligent functionalities are taken into account, we should consider hierarchical processing solutions with intelligence –for a given functionality- distributed along the processing architecture. Within this framework, low performing end units provide (very) simple functionalities but lack of a comprehensive, global view of the problem. As such, we should expect decisions taken in a hierarchical way, with the effectiveness of the decision increasing with the availability of a larger set of data/features and the possibility to execute a more complex algorithm. In fact, more processing demanding algorithms can be run, e.g., to identify a better solution to a specific problem, in systems where larger computational power/energy is available. Result outcomes from the processing stage are then sent back downward the communication chain to reach end units.

3. Energy harvesting and management

In cyber-physical systems where energy availability is an issue, an accurate and sound management of energy in addition to energy harvesting represents a major necessity. Due to its relevance the aspect has been widely addressed in the related literature [4] that, however, leaves major investigation areas open.

The key point of harvesting is to maximize energy acquisition that, scavenged from an available source of power, is stored either in an energy accumulator (e.g., a battery or a super-capacitor) or directly consumed by the electronic system. Likewise, the major goal of energy management is to intelligently control energy consumption by acting at the hardware and software levels, with most sophisticated decision strategies based both on available and forecasted energy availability. Machine learning and, more in general, computational intelligence techniques, are suitable methods to be considered here since physical descriptions about the har-

vestable energy is unavailable due to time variance of the environment and accurate information about the current power consumption is available through measurements only.

3.1 Energy harvesting

In CPSs energy can be harvested by relying on different technologies, e.g., from photovoltaic or Peltier cells to wind and flow turbines and, again, by relying on piezoelectric solutions. The optimal solution for a generic CPS application depends on the available –and harvestable- energy, no matter whether the available power is high or not. "We get what we have, and that has to be enough" says an old leitmotiv. However, of particular relevance for the high density power harvestable are solutions coming from small photovoltaic cells. They can be both deployed outdoor and indoor, with polymer flexible cells that, though less performing compared to the crystalline or amorphous counterparts, are foldable and assume any shape to fit with the available surface. This flexibility makes these harvesting solutions appealing with IoT since the cell can be designed to be deployed directly on the target object. Photovoltaic cells –but the same can be stated for wind and flow turbines- can greatly take advantage of intelligent solutions in the sense that we can maximize the acquired power through adaptation mechanisms. Adaptation is needed every time the energy source, here the light, provides a power density that evolves with time, e.g., due the presence of clouds, dust or water drops on the cell surface as well as due to varying incidence angles. Defined v_p to be the controllable voltage imposed at the photovoltaic cell, the harvestable power depends e.g., on the particular effective solar radiation as per figure 1a, where curve A is associated with a stronger power availability compared with case B. v_p can be controlled over time by means of a Control Power Transfer Module (refer to figure 1b) that grants the harvester to maximize the extracted power to be sent to the storage mean. Details can be found in [4]. Adaptation can be seen as an online learning procedure where the optimal controlling parameters at time k+1 can be achieved through a gradient ascent algorithm as

$$v_p(k+1) = v_p(k) + \gamma \frac{d(i_p v_p)}{dv_p} = v_p(k) + \gamma\left(i_p + \frac{d(i_p)}{dv_p}\right) \quad (1)$$

where γ is a small constant accounting for the step taken along the gradient descent direction. Such of a solution requires acquisition of current i_p and v_p over time through suitable sensors.

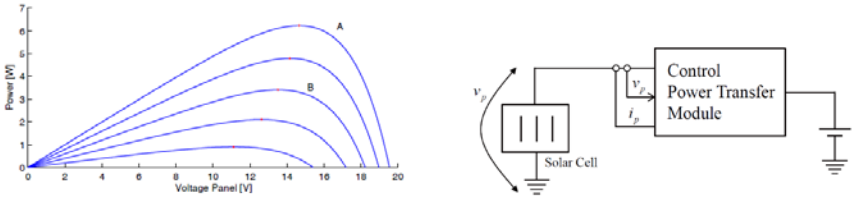


Figure 1: a) the characteristic curves for a photovoltaic cell: the acquired power depends on the applied controlling voltage v_p. *b) The Control Power Transfer Module identifies the optimal voltage according to (1) and applies it to the cell; other architectures can be considered, e.g., see[4].*

3.2 Energy management and research challenges

Energy management is a very important issue in any cyber-physical and IoT system given the fact units are mostly battery powered and need to be kept as simple as possible to reduce their cost.
Energy management can be carried out both at hardware and software/application level by leveraging on

- *Voltage/frequency scaling*. By scaling power voltage and clock frequency the power consumption of the device reduces. In fact, for a CMOS technology the power consumption scales quadratically with the voltage and linearly with the working frequency [4]. Machine learning and fuzzy logic techniques can be adopted e.g., to profile the application at compile time and identify both at compile and run time when and how the control variables should be scaled. Evolutionary computation algorithms can also be considered at compile time to identify the optimal setting of controlling parameters over execution time.
- *Adaptive sampling*. A great energy saving can be achieved by implementing an adaptive sampling strategy where the sampling frequency is adapted according to the current needs of the application. By reducing the sampling frequency –sometimes below the value granting signal reconstruction - we can also reduce the bandwidth needed for communication [10]. Machine learning and fuzzy logic can be fruitfully considered here to control the energy consumption of the system by adaptively acting on the sampling rates of sensors by also taking into account predictions for both the residual and the harvestable (in the future) energy.
- *Keep the solution simple*. This design strategy is always up-to-date in the sense that, in general, complex solutions require a high energy to carry out the due computation which, most of time, is not needed. In fact, in presence of uncertainty affecting the measurements and with the optimal application to be executed on the CPS unknown, it does not make much sense to implement too complex solutions. Machine learning and statistical methods should be investigated to assess the loss in performance associated with a given solution by

also taking into account existing uncertainty and available hardware resources.

- *Consider incremental applications*. Whenever performance accuracy is an issue we can tradeoff accuracy for energy, in the sense that if a higher accuracy level is needed than we tolerate the algorithm to be more complex and energy-eager. Identification of the tradeoff between accuracy performance and energy savings can be carried out with optimization algorithms, e.g., those based on evolutionary algorithms.
- *Duty cycling*. The more you sleep (i.e., the device enters low and deep power sleep modalities) the less energy you consume. By implementing duty cycling at the processor, sensor and memory levels we can significantly control energy consumption. Identification of timing to switch on and off different hardware elements as well as selection of the optimal sleep modality represent an application-dependent complex optimization problem whose optimal solution can be found at compile time with both machine learning and evolutionary computation algorithms.

4. Learning in nonstationary environments

In designing cyber-physical and IoT applications we mostly assume that the process generating the sensor datastream is either stationary (i.e., data or features extracted from the signal are independent and identically distributed (i.i.d.) random variables) or time invariant (the signal does not show an *explicit* dependency on time) [11]. However, such assumptions are hardly met in real applications and represent, in the best case, a first order approximation of the reality. In fact, sensors and actuators are naturally subject to ageing and the environment evolves per se, e.g., think of seasonality, day-night cycles or unpredictable effects affecting a plant. This problem is rarely addressed in existing CPS or IoT applications mostly due to the intrinsic difficulties and challenges that learning in nonstationary environments pose. In the extreme cases where the change intensity impairs the performances of the system, designer implement strategies permitting the application to be remotely updated. However, adaptation to the change in stationarity should be anticipated and managed as early as possible in order to meet the expected quality of service.

In the literature, the difficult problem of detecting time-variance is generally transformed into the detection of changes in stationarity, through suitable transformations, with only few studies addressing time-variance directly at the acquired datastreams level. As such, in the sequel, we focus on the change in stationary problem.

The current literature about learning in nonstationary environments either proposes passive strategies, with the application undergoing a continuous adaptation (passive adaptation modality or passive learning) e.g., see [11-12], or active ones, with adaptation activated only once a trigger detects a change (active adaptation modality or active learning) [11, 13].

Let's assume that the embedded application can be controlled through a vector of parameters θ so that the application code is described by means of the differentiable family function $f(x,\theta)$, x representing the input vector, e.g., containing sensors data. At time t the vector of parameters θ_t, is associated with algorithm $f(x,\theta_t)$. The notation becomes easier to follow if we imagine that function $f(x,\theta_t)$ is a linear filter or a neural function. Differentiability is a convenient assumption to ease the presentation here but it is not strictly requested. In fact, we just require the application to be updated in response to a need.
Every time we are provided with output y_t (measurement) in correspondence with input x, the discrepancy $L(y_t, f(x,\theta_t))$ can be used to update the application. $L(.,.)$ is any appropriate figure of merit used to assess such discrepancy, e.g., a mean square, a Kullback-Leibler distance, etc. Passive and active approaches differentiate on the way the application update is carried out. More in detail,

4.1 Passive adaptation modality

In passive solutions the parameters describing the application are always updated following a compulsive update modality. The classic example is that of linear filters which update online the filter parameters based on the value of the discrepancy $(y_t, f(x,\theta_t)$, mostly based on a least square loss function. More in general, the updated parameters are

$$\theta_{t+1} = \theta_t - \gamma \left.\frac{\partial L(x,\theta)}{\partial \theta}\right|_{\theta_t} \tag{2}$$

where γ is a small and application dependent scalar value controlling the step taken along the gradient descent direction.
Passive solutions do not suffer from false positives and negatives in detecting a change in stationarity/time variance, since the learning-based system/application continuously updates parameters over time by integrating the available novelty content into the model. From this perspective the partial derivative in (2) can be intended as the operator extracting the novelty information content from current data: if there is novelty, the model needs to be updated. As a consequence, passive models are rather sensitive to noise in incoming data: after parameter convergence the application continues to update the parameters so as to track the particular noise realization. A vast literature on passive approaches exist (also called online learning in the neural networks field) and the interest reader can focus on and implement those results in her/his CPS/IoT application [4, 11, 14]. However, the computational complexity of a passive approach might be inappropriate for embedded applications given the continuous need to update the model, operations which might end up in a prohibitive energy consumption and processing time whenever the update phase is energy/computation eager. Ensemble solutions and complex neural networks are examples in this direction where the cost we have to pay for high accuracy and flexibility is computational.

4.2 Active adaptation modality

In the active adaptation strategy, the presence of a change-detection trigger activates the due application reaction following the detected change in stationarity, e.g., by updating the learning-based system. This means that the application running on the embedded device undergoes an update/reconfiguration phase to track the change in stationarity only when triggered by the change detection module. The change detection module –or Oracle- operates by inspecting features φ extracted from the input data or preprocessed variables [4,13] to assess the presence of a change. In other terms the Oracle ω acts as the indicator function

$$\omega(\varphi) = \begin{cases} 1, if\ change\ in\ stationarity\ is\ detected \\ 0, otherwise \end{cases}$$

and the update equation becomes

$$\theta_{t+1} = \theta_t - \gamma\, \omega(\varphi)\, \frac{\partial L(x,\theta)}{\partial \theta}\Big|_{\theta_t} \tag{3}$$

In its basic form the Oracle is based on a threshold; in more sophisticated versions the Oracle also relies on a confidence level –e.g., as it happens in statistical tests- or, even, takes control of the occurrence of false positives by setting its expectation to a predefined value.
Adaptive strategies at cyber-physical systems following the active adaptation modality are known as “detect & react” approaches; such solutions have much focused on classifiers in the related literature, though results are more general [4].

It should be commented that if the computational load of passive solutions is negligible they should be preferred than active ones unless the application is interested in knowing that a change in stationarity occurred (the change in stationarity might be associated with a faulty sensor for instance, as investigated in the next section). If the computational complexity of the update phase is not negligible, also in relationship with the dynamics of the change, we might prefer active solutions. However, active approaches suffer from false alarms (false positives) introduced by the change-detection triggering mechanism hence inducing the application to update even though not strictly needed. Fortunately, in CPS applications false positives do not negatively affect performance but only introduce an extra, not requested, computation.
One should expect that if the physical environment is changing with a low frequency then an active approach might be more appropriate than a passive one. However, again, one should balance the application update phase with its computational complexity and the level of time-variance exposed by the environment the system interacts with. This problem becomes even more relevant and up-to-date in CPSs, IoT, and Smart-X technologies, producing high-dimensional datastreams where it is expected the computational load to be high. Since active and passive

solutions represent extreme strategies, current solutions are investigating hybrid approaches aiming at taking major advantages from them.

4.3 Research challenges

Several research challenges should be addressed in order to support a quick and effective design of cyber-physical and IoT technologies

- *Design methodologies.* Neither investigations nor methodologies are available to shed light on the relationships among the effectiveness of active/passive approaches w.r.t. the speed/nature of the change and the computational complexity of involved methods. Such investigations are fundamental to permit embedded applications to detect possible changes in the environment and react accordingly to keep the quality of service at the appropriate level.
- *Design of distributed decision making applications*. There are no computationally-light change detection mechanisms for distributed embedded systems able to control the false positive rates. The challenge here is to provide distributed –possibly autonomous- decision making strategies, reasonably based on machine learning methods.
- *Approximate computing.* What is largely needed are strategies permitting the harmonization of the learning in nonstationarity environment functionality for distributed embedded systems, possibly within an approximate computing framework. In this case, the approximation level introduced by the hardware as well as that introduced by the adoption of incremental software should be traded-off with accuracy performance as coming from active or passive learning strategies. It is expected that the optimization problem identifying the most suitable level of approximation over time can be carried out with evolutionary computation algorithms.
- *Addressing subtle changes.* Design of learning-based methodologies to detect and anticipate subtle drift changes are needed, e.g., to deal with aging at the sensor and actuators level. In fact, slowly developing changes are hard to detect in the sense their magnitude is small and can be detected by current change detection tests only in the long time period. However, it is expected that availability of datastreams should permit to run machine learning tools to estimate the current behavior of the features and build predictive models to assess the level of time variance.

5. Model-free fault diagnosis systems

Cyber-physical applications are mostly data-eager, in the sense that application decisions and behaviors are strongly driven by the information content extracted from a generally large platform of sensors. This dependency on sensor data can be however very critical, since sensors and real apparatus are prone to faults and malfunctioning that, in turn, negatively affect the information content carried by data and used by the application to make decisions [15, 16]. The problem am-

plifies when low cost sensors are considered and/or sensors are deployed in harsh and challenging environments (e.g., think of a body network where sensors, external buses and connectors are subject to mechanical stress and environmental challenges). As such, faulty sensors detection (also including sensors working in suboptimal conditions) and mitigation are intelligent functionalities that must be included in the design of any CPS to prevent propagation of erroneous information to the decisional level [17]. At the same time, we comment that a generic method designed to detect a fault occurrence will also detect any deviation in the information carried by data, e.g., caused by changes in the environment the sensor is deployed in (time variance), and react erroneously: a novel family of methods are hence requested to detect changes in the information content carried by data, disambiguate between faults and violation of the time invariant hypothesis as well as identify, isolate and possibly mitigate the occurrence of faults. These tasks are carried out by Fault Diagnosis Systems (FDSs).

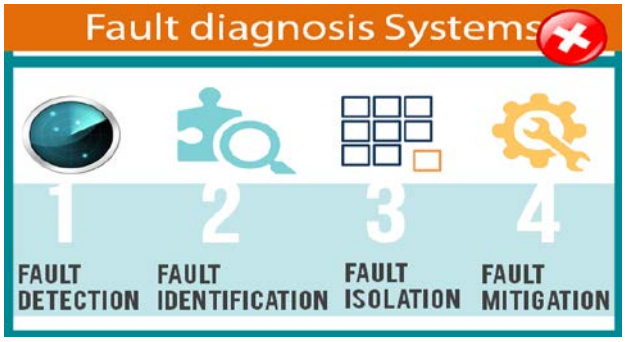


Figure 2: A full fault diagnosis system is characterized by four phases: fault detection aiming at identifying the presence of a fault; fault identification characterizing the type and nature of the fault; fault isolation, localizing the fault and fault mitigation, whose goal is to reduce the impact of the fault on the system

5.1 Model-free fault diagnosis systems

Since CPSs rely on a rich and diversified set of sensors produced by a plethora of companies, it is not possible to request an accurate physical model describing their modus operandi. In this direction, model-free Fault Diagnosis Systems are requested to detect, identify and isolate the occurrence of faults without assuming that their signatures, the nature of uncertainty and their ruling equations are available.

Research on standard (not model-free) FDSs has provided major breakthroughs in past decades by yielding several methodologies for detecting and diagnosing faults in several real world applications e.g., see [17, 18]. However, the effectiveness of a traditional FDS is directly proportional to the available information and priors about the given system, in the sense that availability of 1) the equations ruling the

interaction between the cyber system and the physical phenomenon, 2) information about the inputs and the system noise and 3) availability of the "fault dictionary" containing the characterizations of feasible faults, permits the FDS to operate in optimal conditions. Even though some information might become available in some very specific applications, in general it is hardly usable in cyber-physical applications since the characterization of the CPS interaction, the nature of existing noise and uncertainty, and the signature of expected faults are missing. Moreover, we cannot spend huge efforts in designing a FDS for any CPS-based application if the requested procedure is too complex and articulated since costs and time-to-market are fundamental requirements in designing successful applications. Instead, we would appreciate a methodology able to automatically learn the FDS directly from the data the application receives (computational intelligence-based model-free approach: no available model for the system under investigation, no fault dictionary or fault signatures, no information about the nature of uncertainty). In other terms, all unknown needed entities are learned from available data through computational intelligence and machine learning techniques.

The particular computational intelligence technique depends on the information available to solve a specific sub-problem. For instance, we can use machine learning and fuzzy systems to detect faults; the same technologies can be used to design a fault dictionary, identify the type of fault through a classifier or its magnitude through inference. Fault localization can take advantage of a statistical/probabilistic or fuzzy logic framework whereas mitigation can be based on machine learning and fuzzy systems.

Existing model-free FDSs automatically learn the nominal and the faulty states from sensor data-streams and take advantage of existing temporal and spatial relationships among sensors to detect a possible occurrence of faults. The learned relationships are then used to characterize the system nominal state as well as detect any deviation from such a nominal behavior, diagnose the causes of the deviation, identify the nature of the faults, isolate faulty sensors and –possibly- mitigate their effects. It must be pointed out that most of existing solutions either apply the learning mechanism only to a particular aspect of the FDSs (e.g., the fault dictionary), or solve specific applications: very few –preliminary- model-free methodologies have been proposed in the literature, e.g., see [19]. Such solutions aim at characterizing the relationships present in the acquired data-streams to autonomously learn the nominal state and construct, whenever possible, the fault dictionary during the operational life of the system for fault detection, isolation and identification purposes.

However, despite of these encouraging results major investigations must be accomplished to reach the maturity level needed to support an automatic design of a model-free FDS for networked cyber-physical applications that is automatic, effective, controls false positives/negatives and is computationally light.

5.2 Research challenges

In order to support the next generation of any cyber-physical application we need to address some open research issues

- *Multiple faults*. The "single fault" assumption has to be relaxed to host multiple "concurrent" faults, possibly also of transient type, as it is the case in cyber-physical/human applications. In fact, once a fault occurs it is likely that a domino effect will arise and a subset of sensors affected. Graph-based machine learning techniques are expected to be the right tools to address this research topic.
- *Disambiguation module.* Once a change in stationarity is detected we need to run powerful methods able to disambiguate among changes in the environment, faults and false positives introduced by the change detection method. Given the nature of the problem and the type of information available, it is expected that machine learning and fuzzy tools should be the appropriate techniques to be applied here.
- *Unbalanced data.* Faults are rare events; as such it is hard to have many data coming from the "faulty class". This implies that we need to provide machine learning methodologies to design effective FDSs starting from few and unbalanced data.
- *Modeling aspect*. We need to provide novel design methodologies for model-free FDSs. Such methodologies should be able to automatically configure the FDS for the given application after having profiled sensor data. It is expected that machine learning techniques should be the right tools to be used here to design such of a system.

6. Cyber-security

As mentioned in the previous sections, CPS and IoT technologies will make our life easier and richer in opportunities. However, at the same time, such technologies are prone to cyber-attacks designed to steal precious information and harm people, companies, and governments. In late September 2016 bot nets infected by the infamous IoT malware Mirai [5] caused a severe Distributed Denial of Service Attack (*DDoS*) that prevented major sites to correctly provide services to clients.

Cyber-attacks using IoT devices are expected to steadily increase by taking advantage of network scalability and vulnerability of IoT devices, an aspect that is far from being fixed. Moreover, malware Mirai that was originally designed to infect Linux-based IoT devices is still evolving, with its offspring targeting Windows computers and possibly, in the future, also Android smartphones.

Cyber-attacks to CPS are of deeper concern once applications target our daily life. Consider, for instance, an elderly-care smart home with a solo senior person whose behavior is continuously monitored for safety purposes. The monitoring system consists of some surveillance cameras and various IoT sensors such as door lock and proximity sensors deployed in rooms and corridors. Once the moni-

toring system detects an emergency situation an alert is issued and sent to caregivers and doctors. Following a malware infection, the system control is completely taken by the attacker who might decide to activate e.g., a DDoS attack, hence leaving the person safety issue unattended. Needless to say, CPS designed to protect the person will, in the best case, not work properly. It comes per se that smart cyber-security systems need to be considered in order to prevent such situations.

6.1 How can CPS and IoT be protected from cyber-attacks?

In order to effectively protect a given CPS or IoT device from cyber-attacks we need to consider several cyber-security levels. The first security level should be designed to protect IoT sensor units, the second one has to shield the server where sensor data are processed and control signals issued to be delivered to actuators. Finally, the third level is designed to protect the whole CPS application by monitoring trends of cyber-attack activities on the Internet.

When thinking about the first level of security we should keep in mind that the number of IoT units could be very large and heterogeneous in hardware and software. Here, given the constrained hardware resources, it is mostly unrealistic to implement a complex form of security such as installing anti-virus software or embedding a network intrusion detection mechanism in hardware. The protection measures depend then on the computational resources of the end device. One solution here would be to utilize an anomaly (fault) detection mechanism as those used in fault diagnosis systems or an Oracle, as discussed in the learning in nonstationary environment sections.

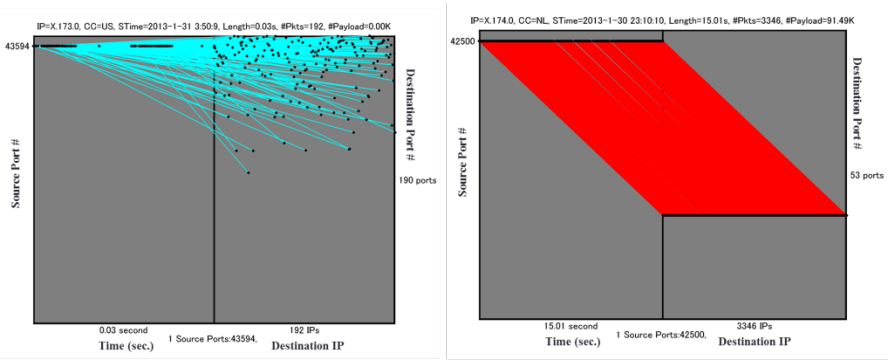
Various software products are available to implement the second level of security. However, there is still plenty of room to design machine learning-based algorithms to identify malwares and detect unknown cyber-attacks using zero-day vulnerability. Many approaches have been proposed so far in this direction, e.g., see [20][21] for details.

The third security layer is also very important and can be seen at the application level. Usually, the number of new malware victims gradually increase, infection then rapidly spreads following some triggering events associated with malware evolution (the open source code of the malware is made available) and, finally, we end up in a pandemic situation. Early detection of malware can then mitigate the infection as presented in the next subsection case study.

6.2 Case study: Darknet analysis to capture malicious cyber-attack behaviors

In order to protect users form cyber threats it is important to characterize malware behavior, with malware instances caught both within one's local network domain and the Internet as a whole. A classification system can then be designed to monitor the current system behavior in order to detect cyber-attacks. One way to observe large-scale events taking place on the Internet is to design a network probe inspecting activity in the *darknet*. The darknet is defined as the set of un-

used address-space of a computer network which should not be used and, as such, not have any normal communication with other computers. Following the definition, it comes out that almost all communication traffic along the darknet is therefore suspicious; by inspecting such of a traffic we can grasp information related to cyber-attacks. Observable cyber-attacks are mainly activities of random scanning worms and DDoS backscatter messages. However, since the darknet can receive packets from the whole Internet space, by operating in the darknet we can monitor the large-scale malicious activities on the whole Internet.



(a) DDoS attacks (b) Scanning

Figure 3: Examples of tiles to represent two typical darknet traffic patterns

Figure 3 shows the darknet traffic patterns representing 2 cases of typical malicious behaviors: (a) DDoS attacks and (b) scanning. The vertical axes of the plots represent the port number of a source host and a destination IP (darknet side). The horizontal axis is divided into two parts: the left one refers to the time needed to send a packet from a source host while the right half refers to the destination IP. Therefore, the plot shows that a packet is delivered from a point in the left side to a point in the right one. According to the nature of cyber-attacks, traffic patterns differentiate. For instance, figures 3 shows a typical DDoS attack and a typical scanning activity.

We discovered that a darknet traffic pattern can be successfully described by the following 17 features related to the statistics of darknet packets [22]:

(1) #Total Packets
(2, 3) Avg and Std of Time Spans of Packets
(4) #Source Ports
(5, 6) Avg and Std of #Packets from Source Ports
(7) #Destination IPs
(8, 9) Avg and Std of #Packets from Dest. IPs
(10) #Destination Ports
(11, 12) Avg and Std of #Packets from Dest. Ports
(13) #Protocol Types
(14, 15) Avg and Std of Payload Sizes
(16, 17) Avg and Std of Spans of Dest. IP Numbers.

Once a darknet traffic pattern for a specific source host is transformed into the 17-dimentional feature vector, machine learning techniques can be applied to clus-

ter data based on the similarity of traffic patterns. Figure 4 illustrates results of darknet traffic patterns inspected in March 2014.

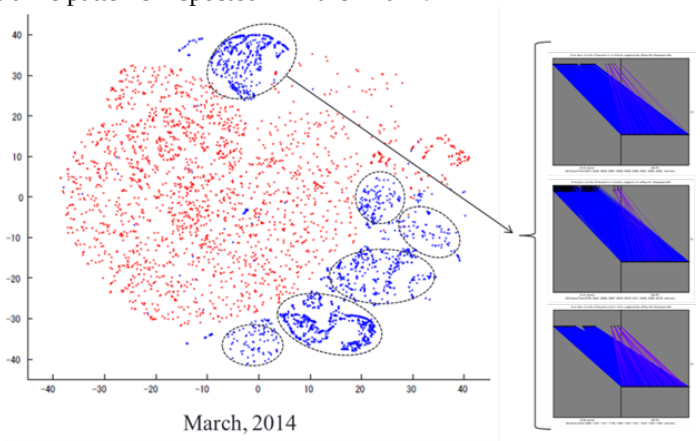


Figure 4: A distribution of darknet traffic patterns. A red point corresponds to a source host whose packet traffic is classified as a DDoS. A blue point refers to a non-DDoS backscatter activity, mostly associated with scanning.

As seen from figure 4 darknet traffic patterns are clearly separated into some clusters of DDoS and non-DDoS (mainly scanning) attacks by *t*-Stochastic Neighbor Embedding (*t*-SNE) [23].

In order to identify a particular type of cyber-attack classifiers must be designed, e.g., SVM and random forests.

The abovementioned 17 features intentionally eliminate detailed information such as port numbers and header information of a TCP packet. This type of abstraction is important to define a broad feature space covering unknown malicious activities. Once unknown darknet traffic patterns are identified using an anomaly detection method, a more precise analysis such as identifying a specific malware type is conducted. In a way what done within the cyber-attack protection intelligent functionality is similar to some steps introduced by fault diagnosis systems where, at first malware is detected, secondly, identified and, third, mitigation strategies taken into account.

Acknowledgement

The authors thank to Dr. Ban Tao, Dr. Junji Nakazato (National Institute of Information and Communications Technology, Japan), and Mr. Jumpei Shimamura (Clwit Inc.) for providing the darknet traffic data and their well-experienced expert knowledge on cybersecurity. This chapter partially contains the experimental results in the research project, supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research (B) 16H02874.

References

[1] International Data Corporation (IDC), “Final Study Report: Design of Future Embedded Systems”, 2012.
[2] D.Evans, “The Internet of Things. How the Next Evolution of the Internet Is Changing Everything”, CISCO White Book, 2011.
[3] P.Marwedel, Embedded Systems Design, Springer, pp.241, 2011
[4] C. Alippi. Intelligence for Embedded Systems: A Methodological Approach. Springer, Switzerland, pp.283, 2014
[5] US-CERT - United States Computer Emergency Readiness Team, Alert (TA16-288A) - Heightened DDoS Threat Posed by Mirai and Other Botnets. November 2015. Tech. rep. URL: https://www.us-cert.gov/ncas/alerts/TA16-288A
[6] C.Alippi, R.Fantacci, D.Marabissi, M.Roveri, "A Cloud to the Ground: The New Frontier of Intelligent and Autonomous Networks of Things", IEEE Communication Magazine, Vol.54, No 12, pp. 14-20, December 2016
[7] Contiki: The Open Source OS for the Internet of Things. URL: http://www.contiki-os.org
[8] ARM, Introduction to the mbed OS 5 handbook. URL: https://docs.mbed.com/docs/mbed-os-handbook/en/latest
[9] Google Inc. Android Things. https://developer.android.com/things
[10] C.Alippi, G.Anastasi, M. Di Francesco, M.Roveri: An Adaptive Sampling Algorithm for Effective Energy Management in Wireless Sensor Networks with Energy-hungry Sensors, IEEE-Transactions on Instrumentation and Measurement. Vol. 59, Issue 2, pp. 335 – 344, February 2010
[11] C.Alippi, R.Polikar, Guest editorial, IEEE Neural Networks and Learning Systems, Special issue on “Learning in nonstationary and evolving environments”, Vol. 25, No. 1, pp. 9-11, January 2014
[12] L. I. Kuncheva, Classifier ensembles for changing environments, in Proc. 5th Int. Workshop Multiple Classifier Systems, pp. 1-15, 2004
[13] C. Alippi; G. Boracchi, and M. Roveri, Hierarchical Change-Detection Tests, IEEE Transactions on Neural Networks and Learning Systems, Vol.28, No.2, pp. 246 – 258, 2017
[14] J. Gama, I. Žliobaite, A. Bifet, M. Pechenizkiy, and A.Bouchachia, A survey on concept drift adaptation, ACM Computing Survey, vol. 46, no. 4, pp. 1-44, April 2014
[15] M. Basseville, I. V. Nikiforov, Detection of Abrupt Changes: Theory and Application, volume 104. Prentice Hall Englewood Cliffs, pp.529, 1993
[16] R.Isermann, Fault-diagnosis systems: introduction from fault detection to fault tolerance, Springer, 2006.
[17] V.Reppa, M. M. Polycarpou, C. G. Panayiotou, Distributed Sensor Fault Diagnosis for a Network of Interconnected Cyber-Physical Systems, IEEE Transactions on control of networked systems, Vol.2, No.1, pp.15-23, 2015
[18] C.Alippi, S.Ntalampiras, M.Roveri, Model-free fault detection and isolation in large-scale cyber-physical systems, IEEE Transactions on Emerging Topics in Computational Intelligence, pp.61-71, February 201

[19] C.Alippi, M.Roveri, F.Trovo', A Self-building and Cluster-based Cognitive Fault Diagnosis System for Sensor Networks, IEEE Transactions on Neural Networks and Learning Systems, Vol. 25, No.6, pp. 1021-1032, June 2014
[20] R.Sommer, V.Paxson, Outside the Closed World: On Using Machine Learning for Network Intrusion Detection, Proceedings of the 2010 IEEE Symposium on Security and Privacy, pp. 305-316, 2010.
[21] C.Sinclair, L.Pierce, S.Matzner, An Application of Machine Learning to Network Intrusion Detection, Proceedings of 15th Annual Computer Security Applications Conference. pp. 371-377, 1999.
[22] N.Furutani, J.Kitazono, S.Ozawa, T.Ban, J.Nakazato, J.Shimamura, Adaptive DDoS-Event Detection from Big Darknet Traffic Data, Neural Information Processing, LNCS 9492, Springer, pp. 376-383, 2015.
[23] L.Maaten, G.E.Hinton, Visualizing Data Using t-SNE, Journal of Machine Learning Research, Vol. 9, 2579-2605, 2008.