Handling Failures in Cyber-Physical Systems: Potential Directions

Taylor Johnson and Sayan Mitra
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
Email: johnso99@illinois.edu, mitras@crhc.uiuc.edu

***Abstract*—The strong coupling of software and physical processes in the emerging field of cyber-physical systems (CPS) motivates the development of new methods to respond to failures in both the cyber and physical domains. To this end, we propose a study of existing work on handling failures from various disciplines. If these models and methods are applicable to CPS, appropriate extensions should be made to apply them. However, if they are not, then we should head off into uncharted territory, developing new methods, which we suggest to be drawn from fields such as formal methods and verification.**

I. Research Problem

The interdisciplinary research problem we propose is to develop failure detection and mitigation strategies that can apply to broad classes of novel CPS, building upon the vast existing literature. Failure detection has been widely studied in a variety of engineering and computer science disciplines, such as control theory [1], reliability [2], artificial intelligence [3], distributed systems [4], among many others. Upon analyzing these models and results, their applicability to CPS can be determined, leading to extensions of previous results or novel methods.

Classes of failures: Since these systems are composed of software interacting with the physical world, many classes of faults exist. On the cyber side, there are timing failures of real-time programs and operating systems, in addition to crash failures, and simply software bugs. On the physical side, there are actuator, control surface, and sensor failures, aside from of course necessary robustness given the potential operating environments of a system. Between these two worlds is the potential for communication failures, such as message drops and omissions, or worse, adversarial man-in-the-middle attacks perhaps culminating in Byzantine failures.

Graceful degradation: Any of these failures can result in a degradation of physical state, and thus potential violation of safety or liveness, which in the context of safety-critical systems must be handled appropriately to prevent catastrophic failures. To avoid over-engineering these systems and as more systems begin to utilize commercial, off-the-self (COTS) components, we must realize that we cannot design systems that do not fail, but instead should direct our attention to designing systems that fail gracefully. Gracefully failing could abstractly be described as maintaining critical safety properties while allowing progress to slow or stop until the system has been able to recover by some autonomous or directed mitigation response, and then proceeding onwards.

Ideally this response would also guarantee some reduced operation for liveness. In decreasing the impact of a fault, the main constraint is to realize there exists a fault and apply a mitigating action before degradation of safety and liveness. If detection can occur faster, then mitigation can begin sooner. Perhaps however, some systems do not need mitigation, and can continue to operate at some level before automatically recovering, such as in self-stabilizing algorithms. Might such concepts be useful in systems that have both physical and software components?

II. Past Work and Potential Directions

The most applicable methods in the controls literature to CPS are probably model-based techniques. The recent research on invertibility [5] and observability [6] could lead to new methods for fault detection, in addition to classical methods such as residual and signature generation [1]. Distributed systems failure detectors give processes information about failures of other processes in the system [4]. They provide algorithms for solving canonical problems such as consensus, leader election, and clock synchronization in the presence of certain types of failures, and also establish lower bounds about impossibility of solving those problems with certain resource constraints.

Given that effectively all CPS must maintain some notion of the current state of the system with regards to time to be able to interact with the physical world, the real-time systems community has analyzed faults. Giotto [7] and its extensions allow for analysis of programs to ensure no timing failures (missing deadlines) can occur in the virtual machine these programs are

executed on. Etherware [8] utilizes a middleware layer and shows the ability of a distributed real-time control system to maintain liveness and safety in spite of communications link failures. The Simplex-architecture supervisory control allows for the automatic mitigation of certain faults by concurrently executing several controllers, one of which is thoroughly tested, and then choosing the control output from the safe controllers if the other controllers issue commands that would take the system to an unsafe set of states [9].

Formal methods and verification will provide useful tools for solving these problems. Sha motivated in [10] the need for formal methods in detecting and mitigating faults in what are now termed CPS, and more recently again in [11] as there are countless further directions to explore. While thus far we have primarily considered mitigating faults to avoid violation of safety properties, we propose also a study of optimizing liveness properties in addition to maintaining safety.

III. Inspiration from Applications

Consider the seemingly disjoint problems of collision avoidance between platoons of cars or among uninhabited aerial vehicles (UAVs) where the state variables of concern are positions, and in avoiding cascading failures in today's electrical or tomorrow's smart grid where the concern is to prevent failures from spreading due to overloading. While these are clearly different problems at a low level, at a higher level, in each case the faster detection occurs, the faster a mitigation response can be performed, such as stopping the vehicle or disconnecting from the grid. This fast response time could prevent the fault from causing a degradation of system properties that leads to catastrophic failure. The mobility allowed in the collision avoidance example gives much freedom for how to respond to such failures and perhaps similar degrees of freedom can be found in more constrained systems such as the power grid. Particularly as distributed power generation grows in popularity, compositional methods for handling cascading failures must be investigated. In some systems, a degradation of state, such as moving from very safe states to less safe states, could potentially be used to detect faults–this is similar to how the Simplex architecture switches between controllers.

In the context of a flock of agents, one agent could broadcast that it has a failed actuator or other local problem, and nearby agents could respond accordingly by moving away. An alternative approach would come from distributed systems, relying on a heartbeat to detect whether some agent has failed–if we have not received a response for a long time, we can suspect the agent with higher probability. However, relying on failure advertisement or heartbeats are not always good responses, as in the case of adversarial faults.

IV. Conclusions

The complexity of this problem, solely in the classes of faults, is great. In addition to exploring the applicability of the existing literature and the new proposed methods to CPS, we must also utilize first principles such as abstraction to deal with this complexity, as otherwise we will wind up over designing these new systems. Some interesting questions are, given a system and a model, (a) When is it possible to detect failures? (b) Can we bound time to detection? (c) How do we minimize time to detection? (d) Can we utilize physical state to more easily or quickly identify failures?

References

[1] P. M. Frank, "Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy—a survey and some new results," *Automatica*, vol. 26, no. 3, pp. 459–474, 1990.

[2] F. P. Preparata, G. Metze, and R. T. Chien, "On the connection assignment problem of diagnosable systems," vol. EC-16, pp. 848–854, Dec 1967.

[3] R. Reiter, "A theory of diagnosis from first principles," *Artificial Intelligence*, vol. 32, no. 1, pp. 57–95, 1987.

[4] T. D. Chandra and S. Toueg, "Unreliable failure detectors for reliable distributed systems," *J. ACM*, vol. 43, no. 2, pp. 225–267, 1996.

[5] L. Vu and D. Liberzon, "Invertibility of switched linear systems," *Automatica*, vol. 44, no. 4, pp. 949–958, 2008.

[6] C. N. Hadjicostis, "Non-concurrent error detection and correction in fault-tolerant discrete-time lti dynamic systems," *IEEE Transactions on Automatic Control*, vol. 48, pp. 2133–2140, 2002.

[7] T. Henzinger, B. Horowitz, and C. Kirsch, "Giotto: a time-triggered language for embedded programming," *Proceedings of the IEEE*, vol. 91, no. 1, pp. 84–99, Jan 2003.

[8] G. Baliga, S. Graham, L. Sha, and P. Kumar, "Etherware: domainware for wireless control networks," May 2004, pp. 155–162.

[9] D. Seto, B. Krogh, L. Sha, and A. Chutinan, "The simplex architecture for safe on-line control system upgrades," in *Proc. American Control Conference*, Philadelphia, PA, Jun. 1998, pp. 3504–3508.

[10] L. Sha, "Using simplicity to control complexity," *IEEE Software*, vol. 18, no. 4, pp. 20–28, 2001.

[11] L. Sha, S. Gopalakrishnan, X. Liu, and Q. Wang, "Cyber-physical systems: A new frontier," in *Sensor Networks, Ubiquitous and Trustworthy Computing, 2008. SUTC '08. IEEE International Conference on*, June 2008, pp. 1–9.