Critical Infrastructure Systems and the Internet of Things

Phillip A. Laplante
Pennsylvania State University
plaplante@psu.edu

***Abstract* — Over the last few years there have been many publications in the technical literature and popular press heralding spectacular applications of the Internet of Things (IoT). In fact, the IEEE has several publications and a major strategic initiative involving the IoT and the Reliability Society has been actively involved in this work [1]. The US National Institute of Standards and Technology (NIST) also has a significant initiative on IoT, and I have been participating in this work.**

Based on my own work with NIST and my work in licensing of software engineers in the US e.g. [2], I want to discuss some considerations for IoT applications that involve or interact with critical systems.

***Keywords* — internet of things, IoT, critical systems**

I. Introduction

IoT systems contain specialized sensors, computing devices and actuators operating in some environment. Jeff Voas has proposed a more formal model for an IoT based on a set of primitives [3]:

1. sensor -- an electronic utility that digitally measures physical properties such as temperature, acceleration, weight, sound (e.g. cameras and microphones),
2. snapshot -- an instant in time,
3. cluster -- a grouping of sensors that can appear and disappear instantaneously,
4. aggregator – a software implementation based on mathematical function(s) that transforms sensor data into intermediate data,
5. weight – the degree to which a particular sensor's data will impact an aggregator's computation,
6. communication channel – any medium by which data is transmitted (e.g. wireless or wired),
7. external utility (eUtility) – a software or hardware product or service which executes processes or feeds data into the overall dataflow of the NoT,
8. decision trigger -- creates the final result or results from data concentrations and any other data needed to satisfy the purpose and requirements of a specific NoT [3].

This formal framework is already proving to be quite useful in characterizing systems, helping identify gaps during requirements elicitation, proving properties of the system, and helping engineers to design test cases, among many other uses.

Although relatively new, there are many deployed instances of IoTs. Typical applications include smart homes, smart cities, automobiles, and critical infrastructure systems such as water treatment, power generation and distribution. Many of these systems are experimental and small in scale as platform builders and end users are still learning about the challenges of building real IoT systems. From my discussions with implementers of real, industrial strength IoT systems, I learned that one of the biggest challenges is the difficulty in meeting the hyped capabilities promised by pundits. For example, achieving desired levels of availability and reliability for even the simplest IoT system is quite challenging. So when we consider and IoT for a critical infrastructure IoT systems we need to be very careful about the gap between theory and reality since the consequences of failed critical systems can be significant.

II. Critical Systems

The notion of a "critical" system is subjective, even if we base our definition on the degree to which a failure affects humans. For example, it is clear that a system failure at a nuclear plant could have mortal consequences. But it is not so clear for some kind of financial system hack that results in significant loss of funds– while no physical injury results, the impact on the lives of those who lost their savings would be significant, even "critical." For the purposes of this discussion, let's consider critical systems to include:

- telecommunication infrastructure,
- water supply,
- electrical power system,
- oil and gas,
- road transportation,
- railway transportation,
- air transportation
- banking and financial services,
- public safety services,
- healthcare system,
- administration and public services.

This list is not arbitrary – it's based on a US Congressional Report [4]. My own research interests include IoT systems in healthcare, for example, to assist patients with Alzheimer's disease.

Designers and builders need to take special precautions to ensure the safety of critical systems. But what about non-critical systems that might inadvertently interact with a critical system in an IoT and cause a catastrophic failure? Even if a certain IoT is not intended for a critical application the potential for unforeseen interaction with a life critical system is possible. In fact the IEEE definition of IoT concedes the notion that all system in an IoT might be connected: "The Internet of Things (IoT) is a computing concept where all things, including every physical object, can be connected - making those objects intelligent, programmable and capable of interacting with humans."

This scenario is very likely. In one demonstration, "hackers" where able to take control of the braking apparatus of a Jeep car, causing it to crash. But the attack vector was through software written to control the audio system [5]. And the Online Trust Alliance (OTA), whose members include Microsoft, Symantec, and Verisign, warn about this kind of problem and have published guidelines in this regard [6]. I would go further and suggest that every IoT system needs to be engineered as if it is a life critical system because of the potential for interaction (either planned or inadvertent) with a critical one.

III. Failure / Fault Database

To help systems designers building critical applications in the IoT we need a national database for IoT incident and vulnerabilities, similar to the National Vulnerability Database hosted by NIST [7]. We could use the NIST primitive set to key the database items. Such a database could:

1. help requirements engineers to anticipate hazards, write misuse and abuse cases, and create antipatterns,
2. assist designers in the IoT create more fault tolerant systems
3. facilitate creation of sensor fusion algorithms and techniques for dealing with uncertain data,
4. enable prognostic health management in sensor networks,
5. guide test engineers in developing test cases,
6. help users of the IoT understand the limitations of real systems.

And there would many other uses.

As a step in this direction and as part of my work with NIST, my son and I have created an experimental database of reported sensor failures in deployed IoT applications. Some of the reports are fascinating, for example, sensors being submerged by 100 year flood levels in an environmental IoT. The site is in the alpha testing phase, enters beta testing soon and will be made public by the end of the year.

In the meantime, if you have direct involvement with a deployed IoT and it has experienced some kind of sensor failure – whether due to device physics, environmental factors, mishap, or whatever -- I'd like to include that information in the sensor failure database. If you are interested, please email me and I will send you details. If requested, certain information can be kept confidential.

References

[1] IEEE Internet of Things Community, http://iot.ieee.org/, 2015.

[2] P. Laplante, "When does software affect the health, safety and welfare of the public?," Reliability Society Newsletter, http://rs.ieee.org/images/files/newsletters/2012/1_2012/Phil%20Laplante htm, February, 2012.

[3] J. Voas, "Foundations of the Internet of Things," http://dslsrv.gmu.edu/The%20Foundations%20of%20IoT_v2.5.pdf, 2015.

[4] J. D. Moteff and P. Parfomak, "Critical Infrastructure and Key Assets: Definition and Identification," Congressional Research Service, Library of Congress, 2004.

[5] A. Greenberg, "Hackers Remotely Kill a Jeep on the Highway -- with Me in It," Wired, http://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/, July 21, 2015.

[6] R. Chirgwin, "IoT Security is Rubbish Says IoT Vendor Collective," The A Register, August 12, 2015 http://www.theregister.co.uk/2015/08/12/iot_security_is_rubbish_says_iot_vendor_collective/, 2015

[7] National Vulnerability Database, https://nvd nist.gov/, 2015.

Phillip A. Laplante is Professor of Software Engineering at The Pennsylvania State University. He received his B.S., M.Eng., and Ph.D. from Stevens Institute of Technology and an MBA from the University of Colorado. He is a Fellow of the IEEE and SPIE and has won international awards for his teaching, research and service. Since 2010 he has led the effort to develop a national licensing exam for software engineers.

He has worked in avionics, CAD, and software testing systems and he has published 27 books and more than 200 scholarly papers. He is a licensed professional engineer in the Commonwealth of Pennsylvania and a Certified Software Development Professional. He is also a frequent technology advisor to senior executives, investors, entrepreneurs and attorneys.

His research interests are in software testing, requirements engineering and software quality and management.

Prior to his appointment at Penn State he was a software development professional, technology executive, college president and entrepreneur.

More information can be found at www.personal.psu.edu/pal11