

Research Review 2018

Leveraging Emerging Changes in Computing

Grace Lewis

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Copyright 2018 Carnegie Mellon University. All Rights Reserved.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

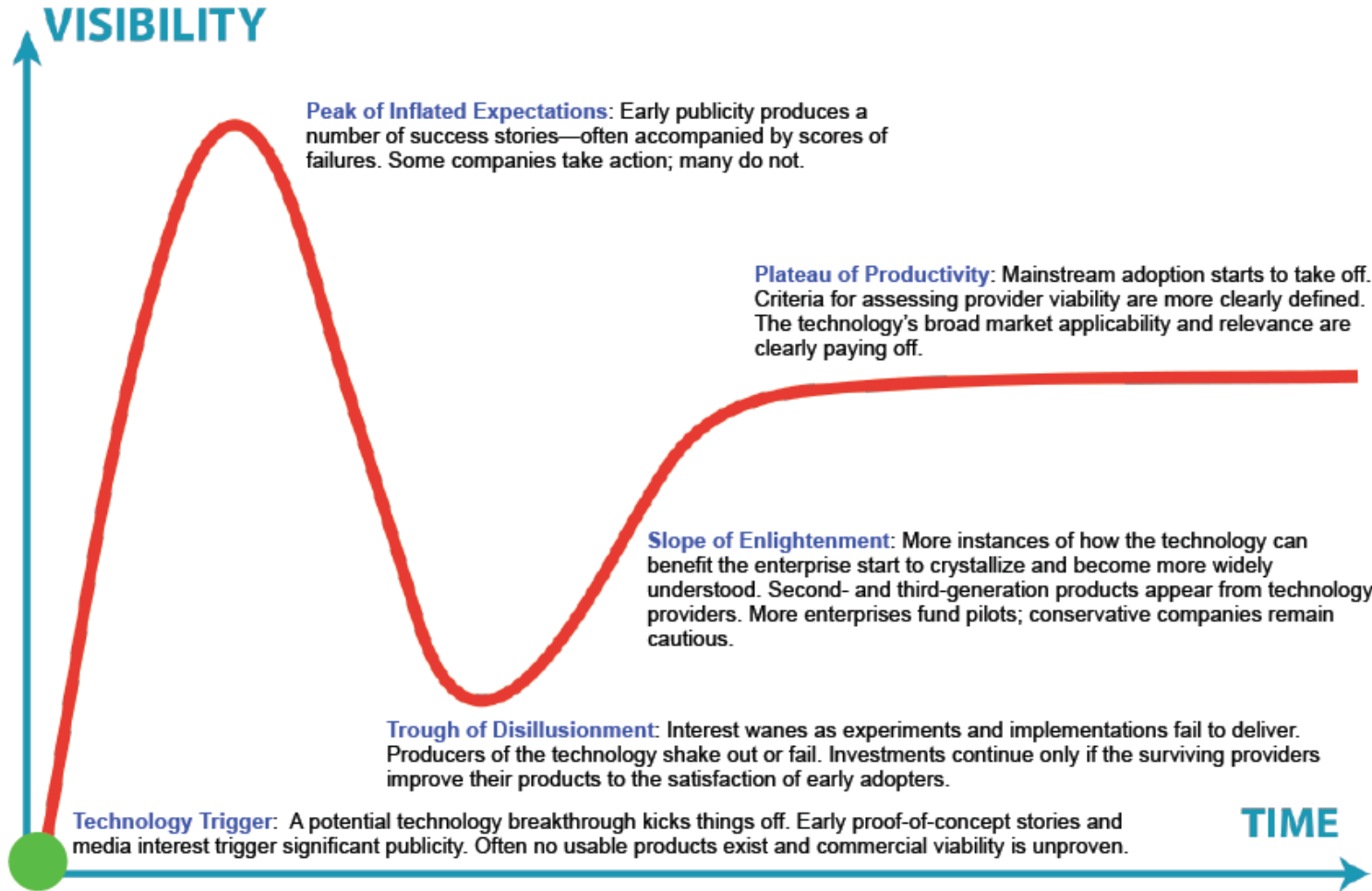
[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

Carnegie Mellon® is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

DM18-1196

Technologies Follow Hype Cycles



Emerging and evolving technologies affect the way that systems are developed, deployed, and acquired

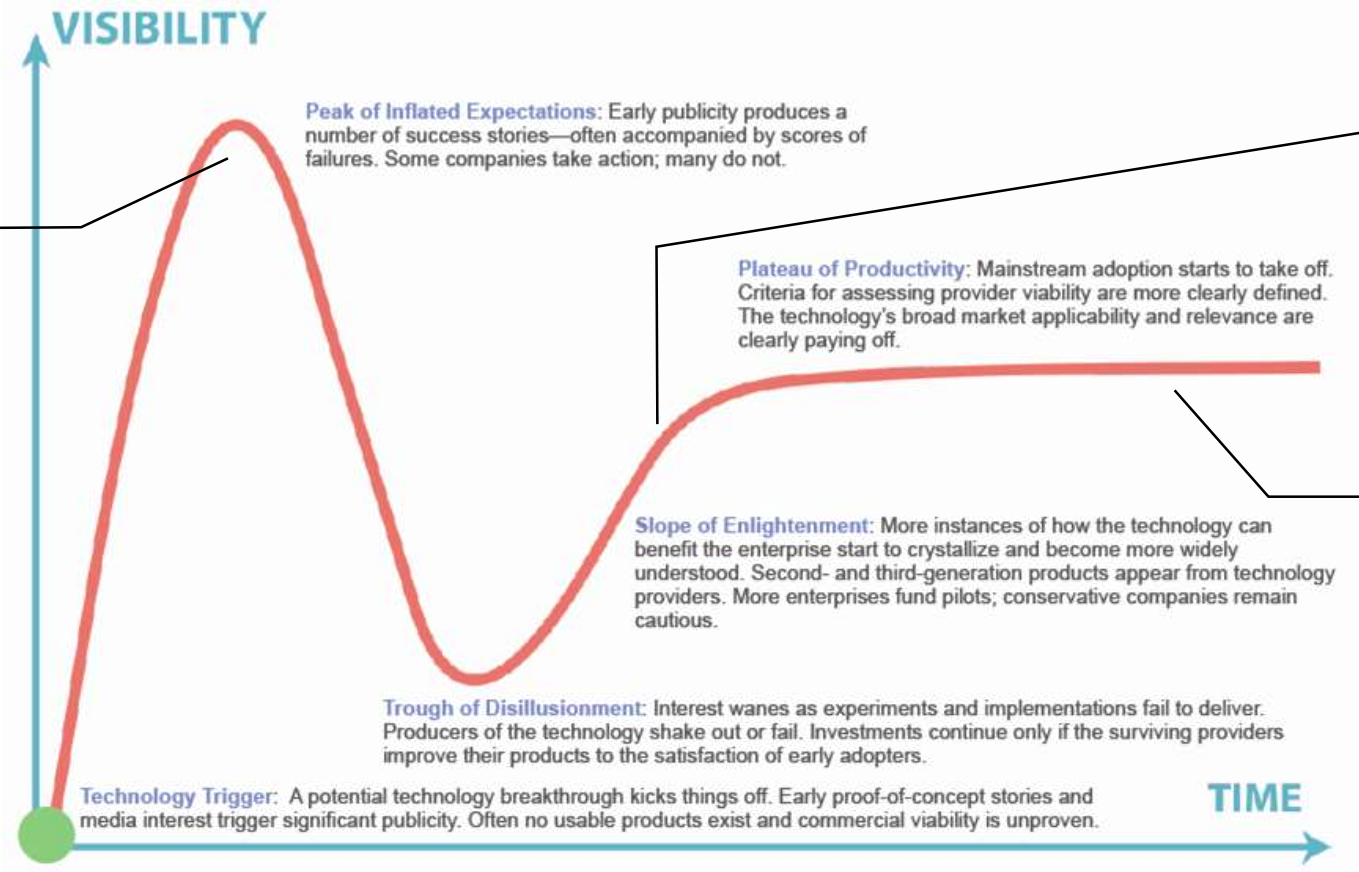
Timely identification, understanding, and adaptation of technologies leads to realizing computational and algorithmic advantage in DoD systems

Source: Gartner, Hype Cycle for Emerging Technologies. Credit: © 2018 Gartner, Inc. and/or its Affiliates. All Rights Reserved.

Exploiting and not Being Exploited by Emerging Changes in Computing—1

1

Target 2-10-year-out promising technologies for DoD systems that are near the *Peak of Inflated Expectations* or going down the *Trough of Disillusionment* ...



2

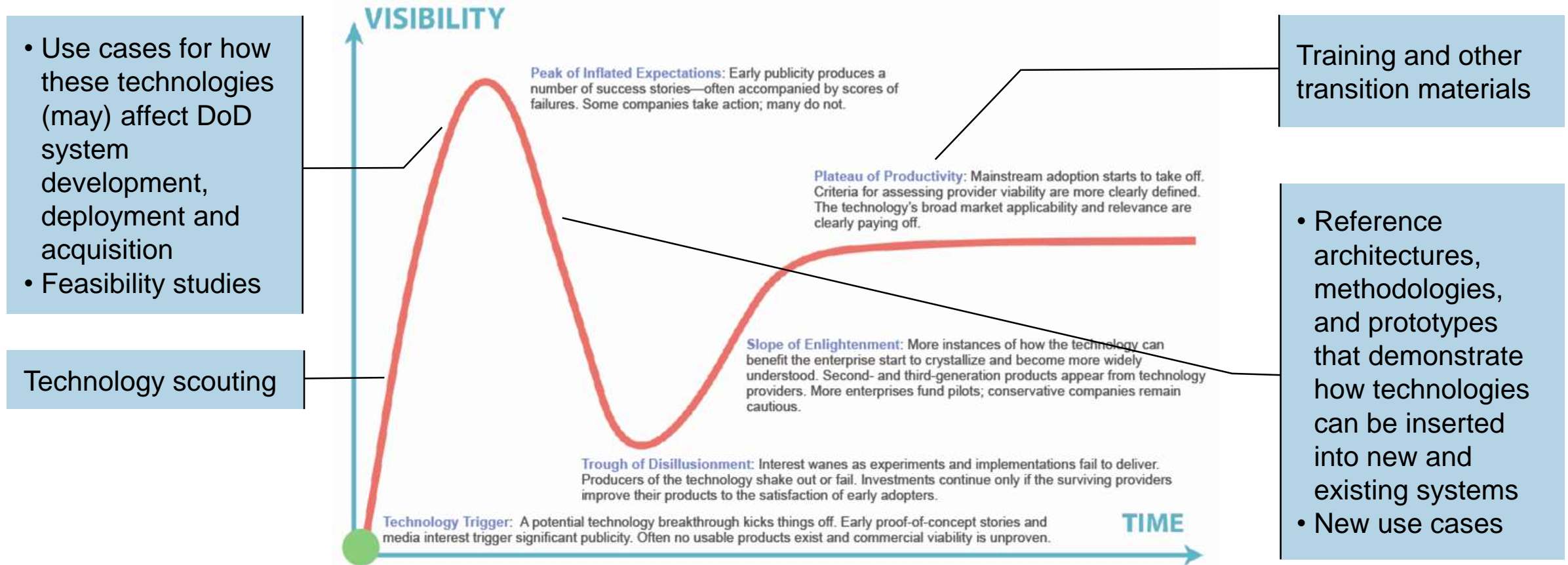
... to accelerate the *Slope of Enlightenment* ...

3

... and get them to the *Plateau of Productivity* faster.

Exploiting and not Being Exploited by Emerging Changes in Computing—2

The focus of our work is different depending on where technologies are in the hype cycle

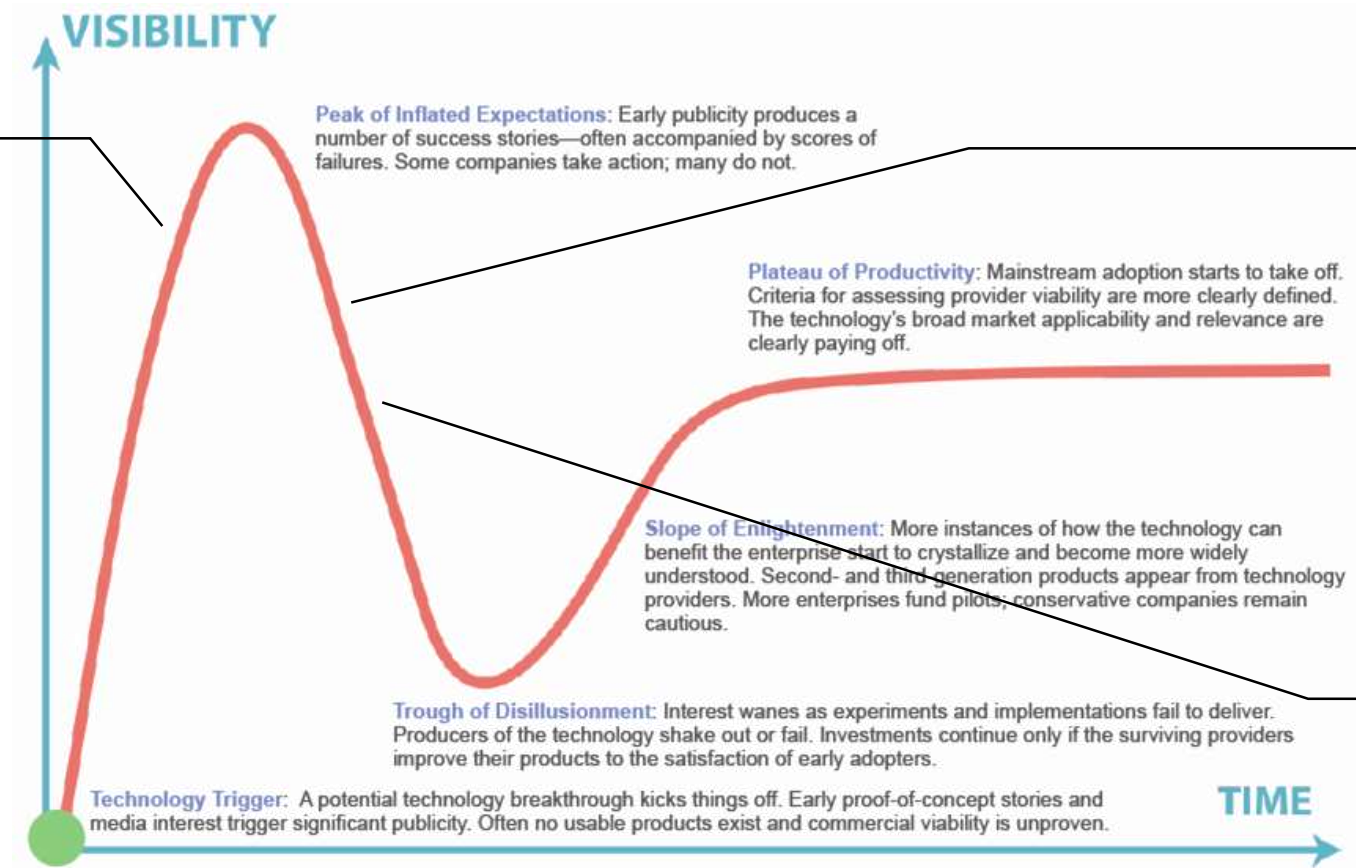


Examples of SEI Work

SOTA

LSI – Identifying Unlikely Events

Inverse Reinforcement Learning



SOTA

LSI – High-Assurance Software-Defined IoT Security

Secure IoT Platforms

SOTP

SOTA

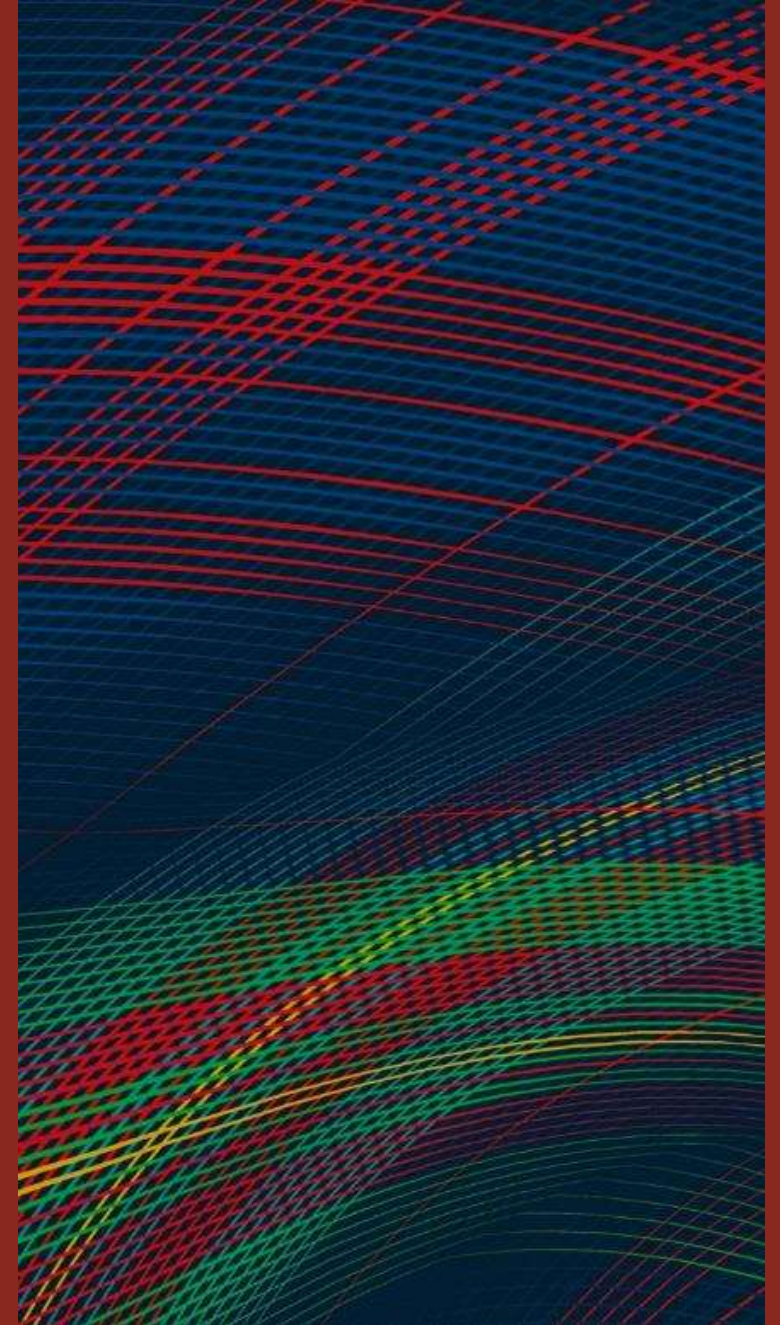
LENS – Timing Verification of Undocumented Multi-Core

Multicore in Real-Time Systems

Research Review 2018

Identifying Unlikely Events

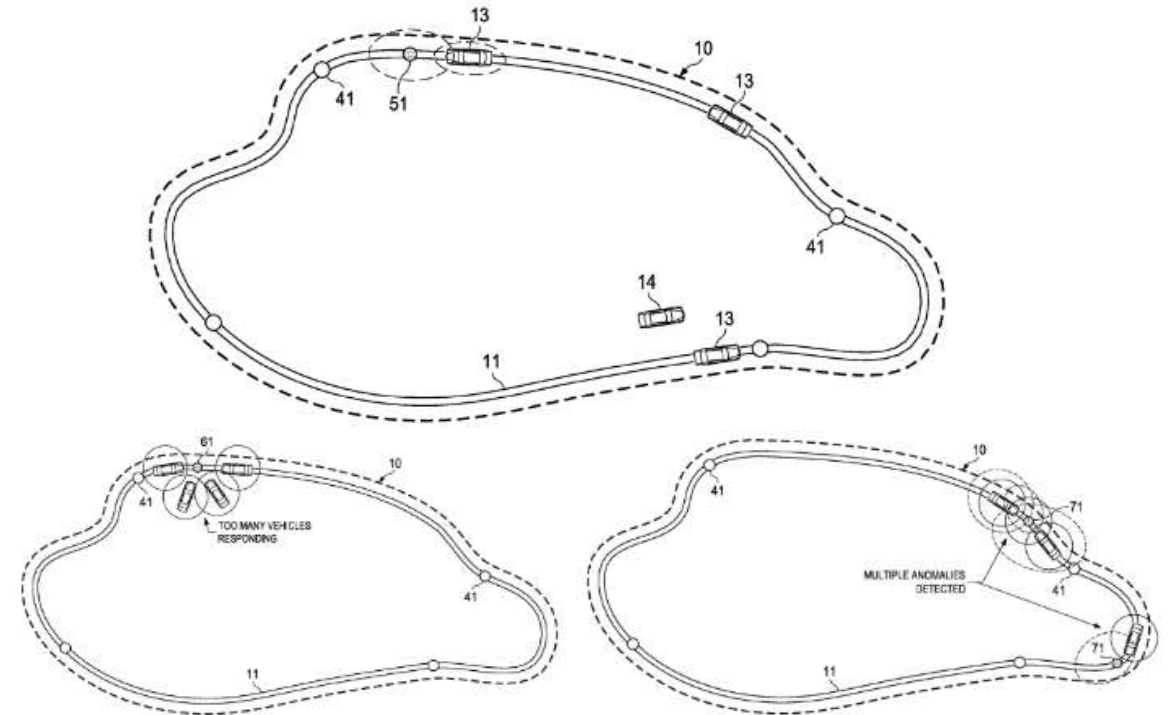
PI: Dr. Drew Gifford



Identifying Unlikely Events — Example

Perimeter Patrol

Goal: visit every meter of fence frequently, stop and handle issues as they come up



Southwest Research Institute Patent US 2015/0293535
Cooperative Perimeter Patrol System and Method 10/15

Identifying Unlikely Events — Current Algorithms



- are typically hand-crafted for particular applications
- require labeled anomalous data
- have high false-positive rates that require humans to verify predictions

Reinforcement Learning

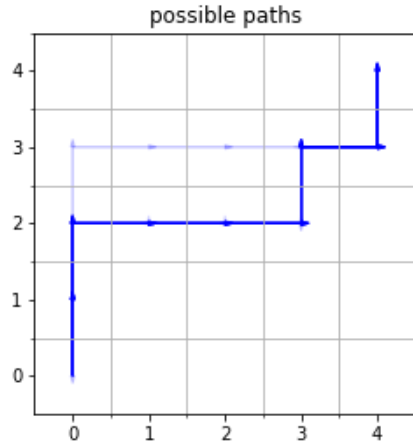
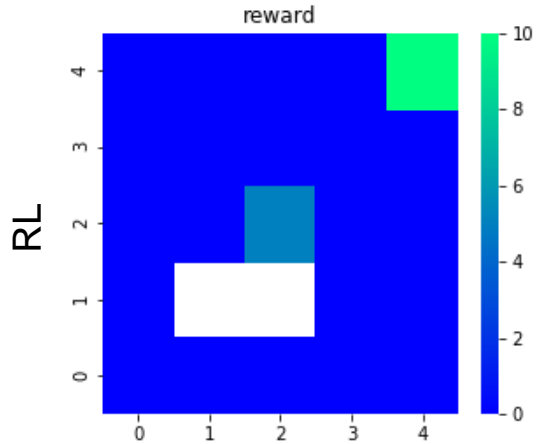


Reinforcement Learning (RL) involves a known reward function.

Given rewards for actions, what behaviors maximize the total reward?

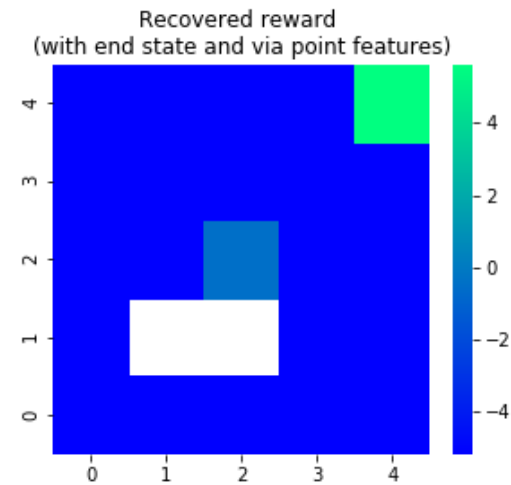
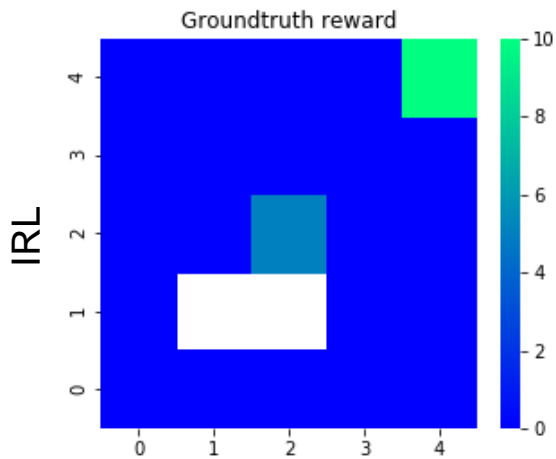


Inverse Reinforcement Learning



Inverse reinforcement learning (IRL) **recovers the reward function** that the demonstrated behaviors represent.

IRL learns a statistical model of likely (routine) and unlikely (anomalous) actions that are taken from each state.



Given observed behaviors, what reward structure would explain these behaviors?

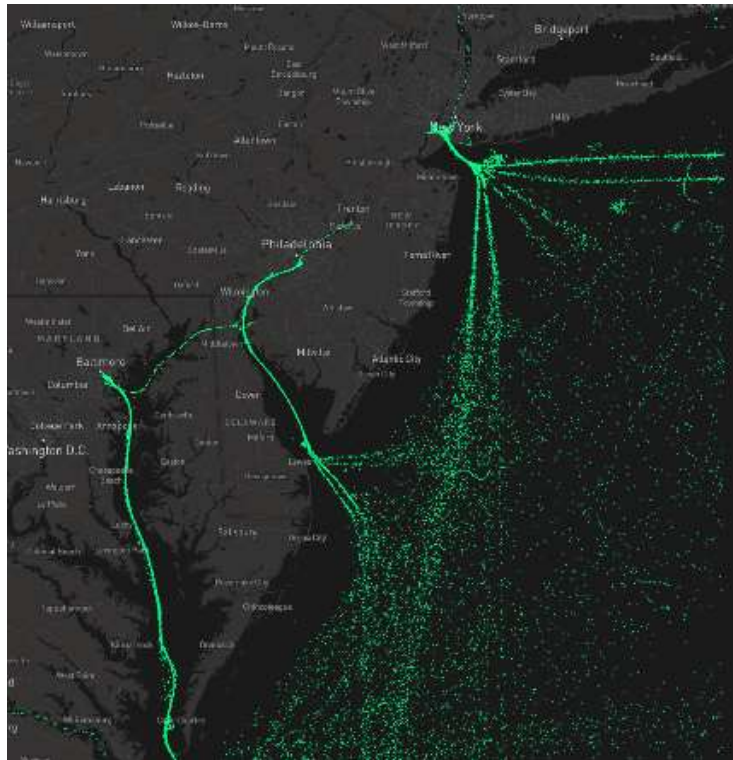
GridWorld Examples for RL and IRL

Project Goal: Extend Inverse Reinforcement Learning

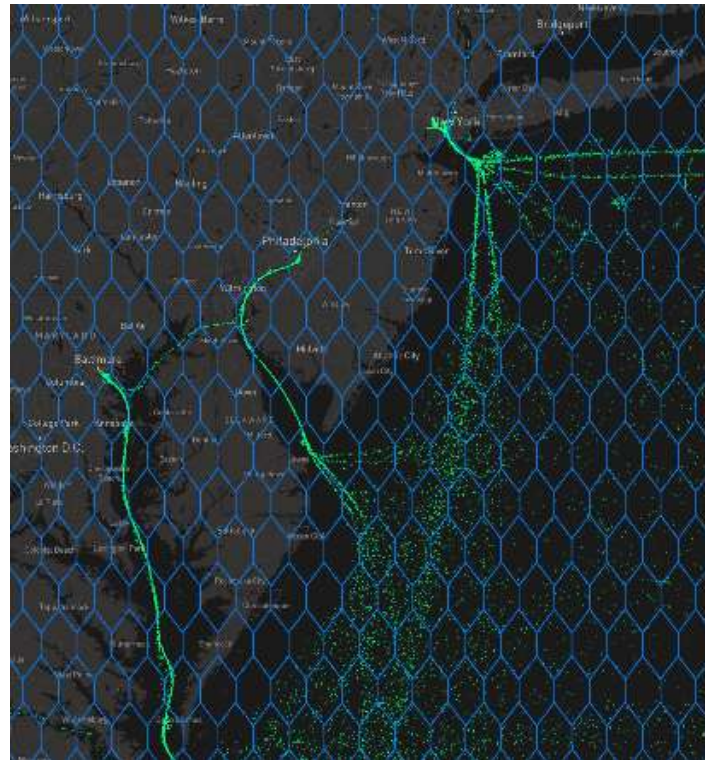
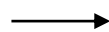


- Apply Inverse Reinforcement Learning (IRL) to a variety of DoD problems
- Determine a method for handling multiple possible routine actions
- Improve the training time for IRL using sparse linear algebra and parameter servers
- Create interfaces to allow analysts to observe unlikely actions and prioritize them
- Develop an end-to-end demonstration of routine/unlikely event detection

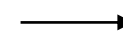
Results: IRL Applied to Marine Vessel Data



Raw Geographic
Cargo Ship Data

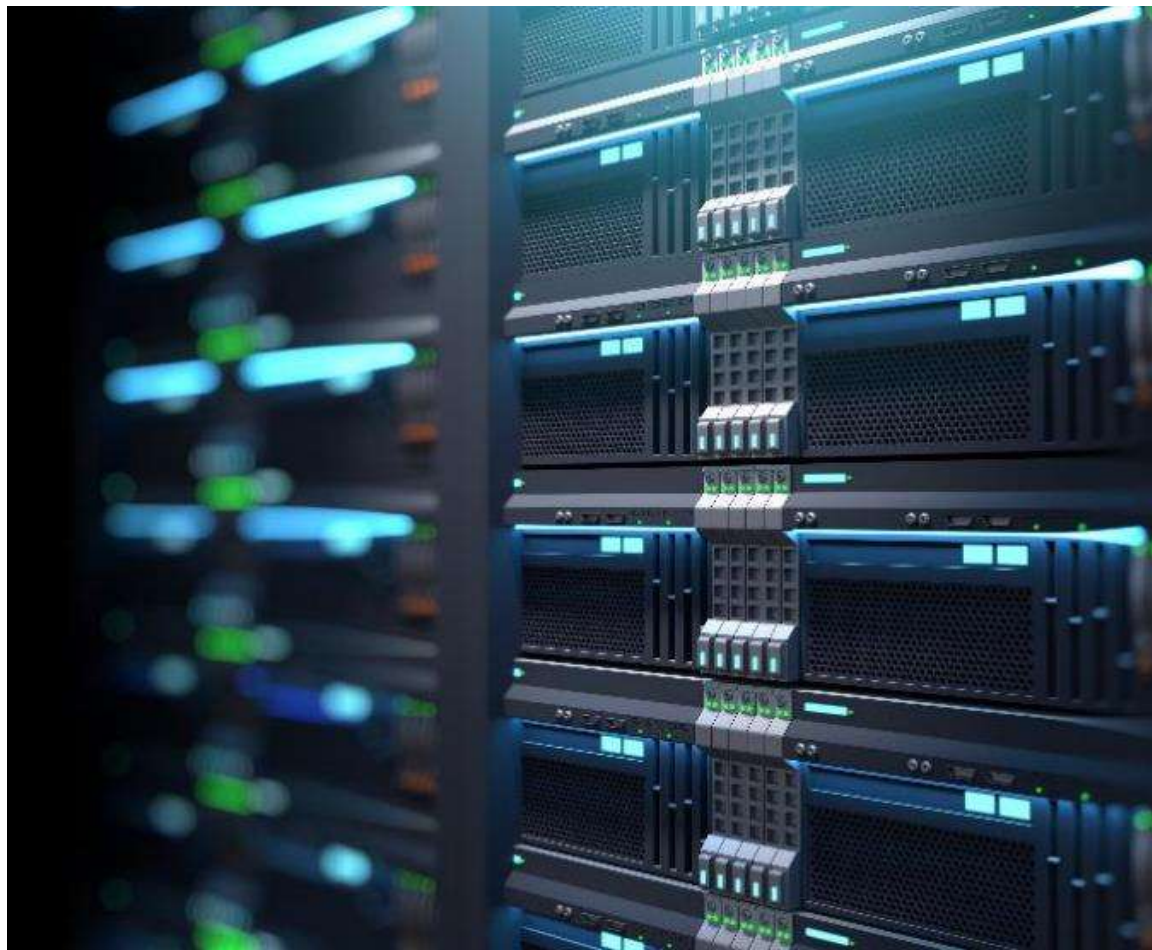


Convert into
HexWorld



Predicted Path into
New York Harbor

What's Next



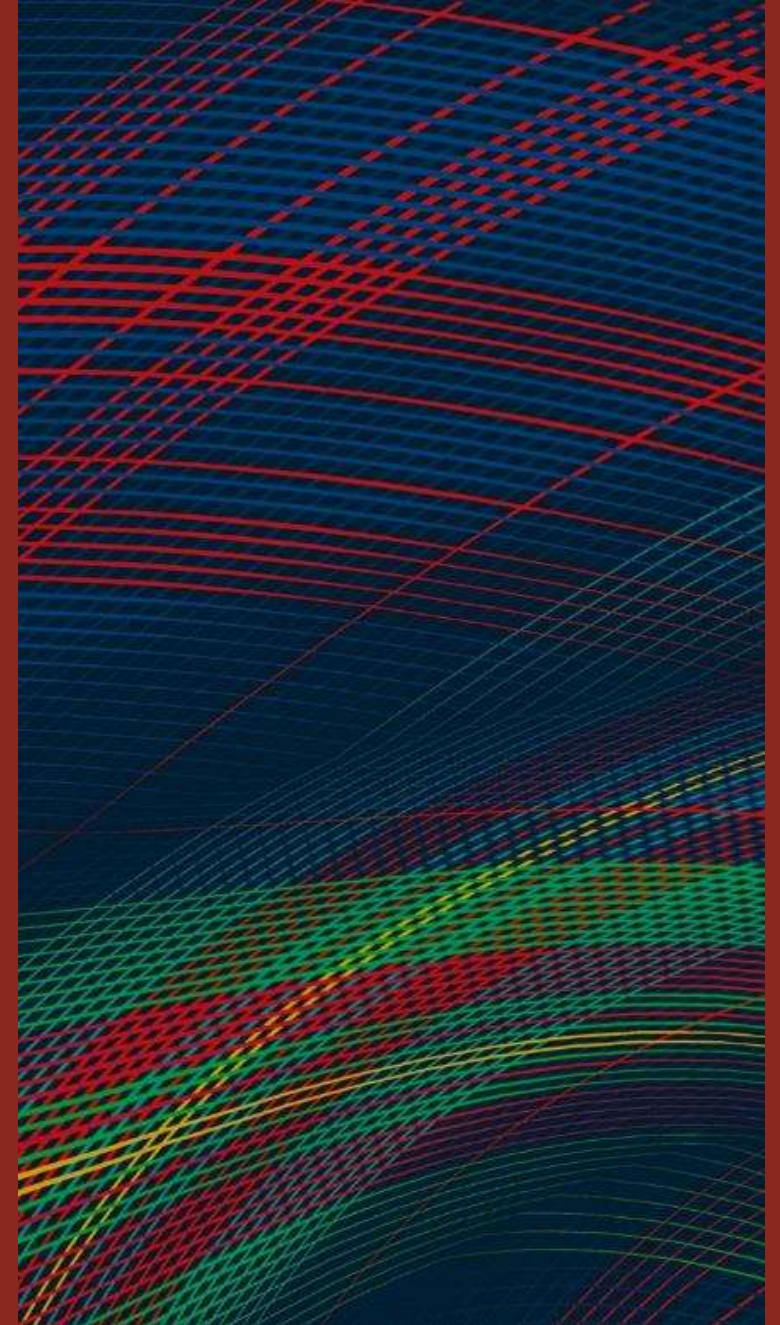
Extending results beyond movements in the physical world...

Working with CMU Parallel Data Lab to model behavior of supercomputer users.

Research Review 2018

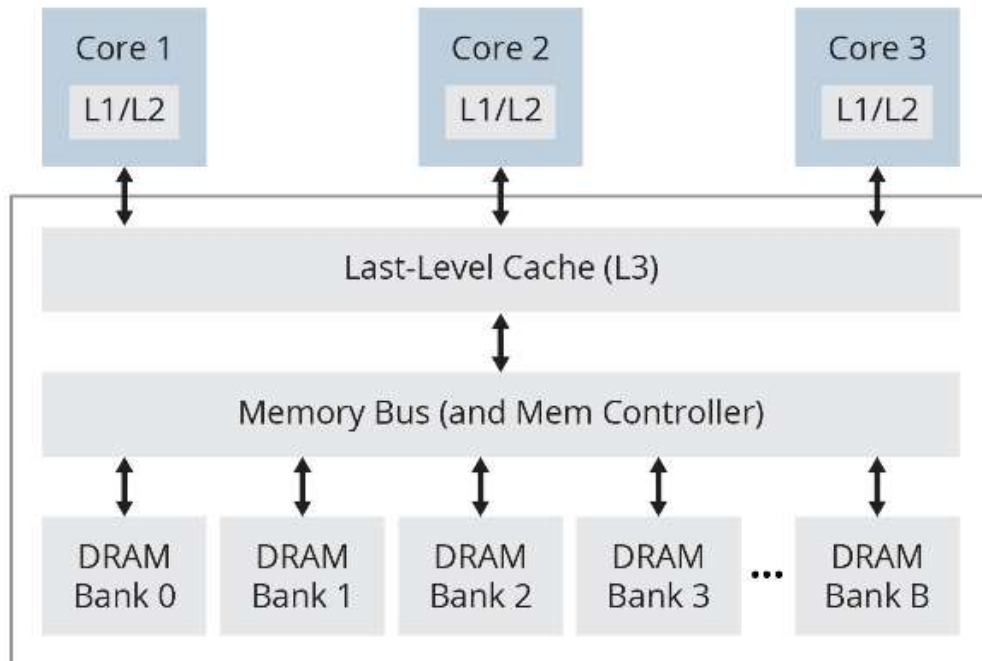
Timing Verification of Undocumented Multi-Core

PI: Dr. Bjorn Andersson

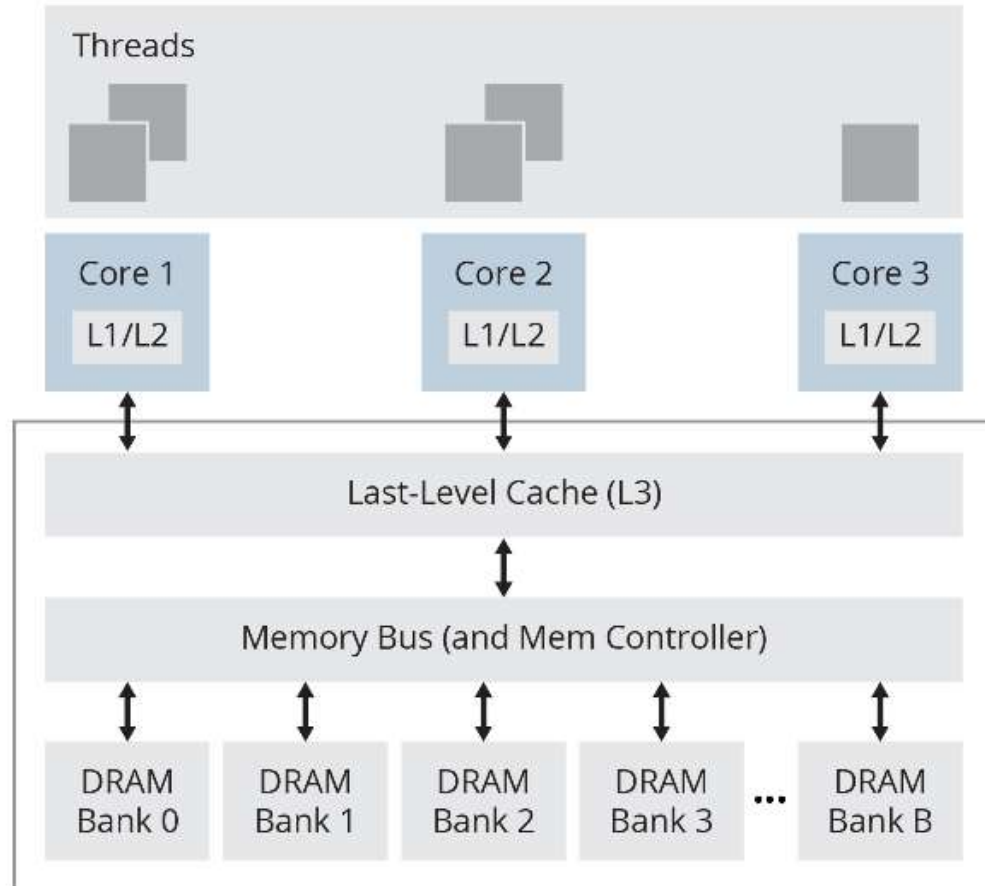


Hardware Trends

Hardware trends: Multicore processors are the norm



Hardware Trends



Hardware trends: Multicore processors are the norm

Observations

- The execution speed of a thread depends on other threads on other cores
- Many DoD systems have real-time requirements

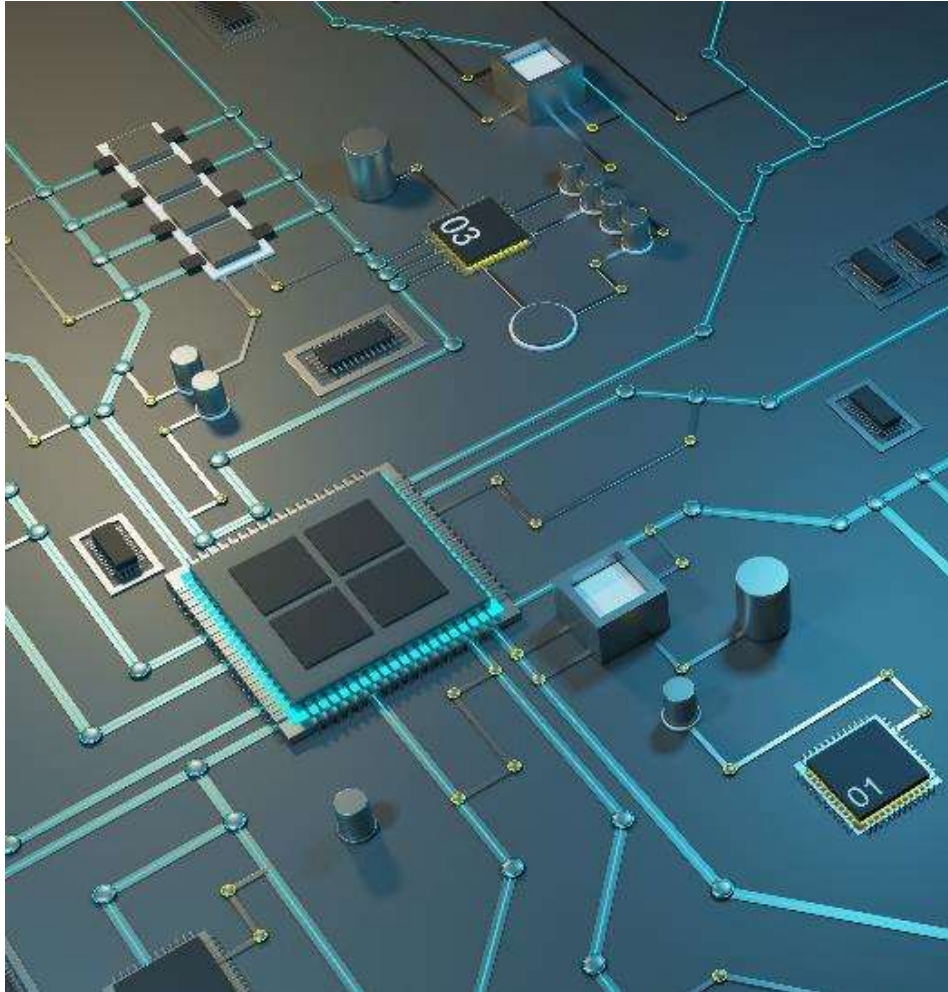
Problem

- Real-time threads fail to satisfy real-time requirements, leading to mission failure

SEI solution

- Increased safety and faster development and deployment

SEI Work in Multicore



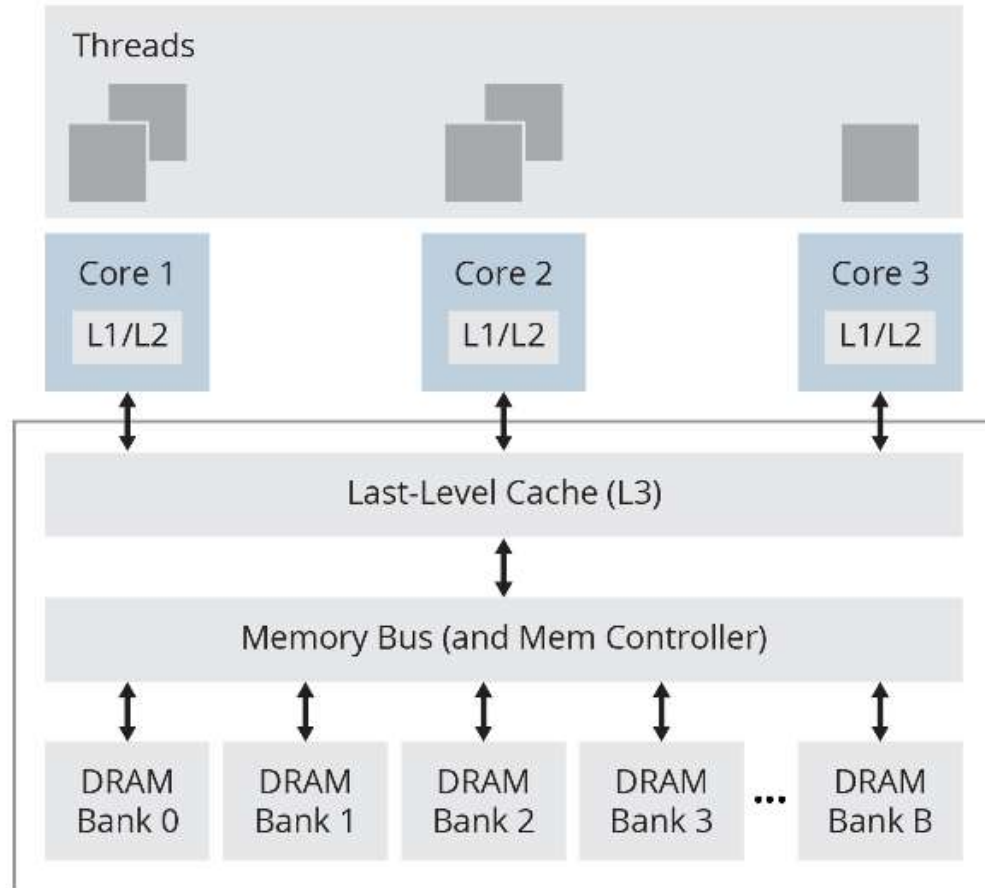
High-Confidence Cyber-Physical Systems (LINE), 2011-2013

Use of Virtual Machines in Avionics Systems (FAA), 2015-2017

Timing Verification of Undocumented Multicore (LENS), 2017

Forthcoming research in multicore, 2018 and beyond

Hardware Trends



Issues

- Shared hardware resources impact timing
- 103 times slowdown observed
- Current methods cannot deal with undocumented hardware
- The problem is getting worse: Slowdown increasing, more undocumented hardware

The SEI has developed a new method...

Schedulability Analysis of Tasks with Corunner-Dependent Execution Times

BJÖRN ANDERSSON, HYOSEUNG KIM, DIONISIO DE NIZ, MARK KLEIN, RAGUNATHAN (RAJ) RAJKUMAR, and JOHN LEHOCZKY, Carnegie Mellon University and UC Riverside

Consider fixed-priority preemptive partitioned scheduling of constrained-deadline sporadic tasks on a multiprocessor. A task generates a sequence of jobs and each job has a deadline that must be met. Assume tasks have Corunner-dependent execution times; i.e., the execution time of a job J depends on the set of jobs that happen to execute (on other processors) at instants when J executes. We present a model that describes Corunner-dependent execution times. For this model, we show that exact schedulability testing is co-NP-hard in the strong sense. Facing this complexity, we present a sufficient schedulability test, which has pseudopolynomial-time complexity if the number of processors is fixed. We ran experiments with synthetic software benchmarks on a quad-core Intel multicore processor with the Linux/RK operating system and found that for each task, its maximum measured response time was bounded by the upper bound computed by our theory.

Categories and Subject Descriptors: D.4.7 [Operating Systems]: Organization and Design—Real-time systems and embedded systems; G.4 [Mathematical Software]: Algorithm design and analysis

General Terms: Algorithms, Performance, Theory

Additional Key Words and Phrases: Multiprocessor, multicore processor, real-time scheduling, memory contention

ACM Reference format:

Björn Andersson, Hyoseung Kim, Dionisio de Niz, Mark Klein, Ragunathan (Raj) Rajkumar, and John Lehoczky. 2018. Schedulability Analysis of Tasks with Corunner-Dependent Execution Times. *ACM Trans. Embed. Comput. Syst.* 17, 3, Article 71 (May 2018), 29 pages. <https://doi.org/10.1145/3203407>

1 INTRODUCTION

Guaranteeing hard real-time requirements of concurrent software executing on a COTS multicore processor is a significant challenge because the execution speed (hence the execution time) of a program may depend on whether another program executes on another processor. This dependence has three causes: (1) sharing of resources in the memory system (e.g., cache, memory bus, memory banks, hardware prefetching units, TLBs), (2) sharing of resources within a processor core (e.g., multiple programs sharing ALUs as part of simultaneous multithreading),

Authors' addresses: B. Andersson, D. de Niz, and M. Klein, Software Engineering Institute, Carnegie Mellon University, 4500 Fifth Avenue, Pittsburgh, PA 15213, USA; emails: {baanderson, dionisio, mk}@sei.cmu.edu; H. Kim, Department of Electrical and Computer Engineering, University of California, 900 University Avenue, Riverside, CA 92521, USA; email: hyoseung@ucr.edu; R. (Raj) Rajkumar, Department of Electrical and Computer Engineering, Carnegie Mellon University, 4720 Forbes Avenue, Pittsburgh, PA 15213, USA; email: rajkumar@cmu.edu; J. Lehoczky, Department of Statistics and Data Science, Carnegie Mellon University, 4720 Forbes Avenue, Pittsburgh, PA 15213, USA; email: j16@andrew.cmu.edu.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2018 ACM 1539-0807/2018/05-ART71 \$15.00
<https://doi.org/10.1145/3203407>

ACM Transactions on Embedded Computing Systems, Vol. 17, No. 3, Article 71. Publication date: May 2018.

71

“...we present a sufficient schedulability test... This new schedulability test can be seen as a generalization of the classic exact schedulability test for fixed-priority preemptive single-processor scheduling...”

B. Andersson et al., "Schedulability Analysis of Tasks with Co-Runner-Dependent Execution Times," *ACM Transactions on Embedded Computing Systems*, 2018.

Looking Ahead: Forthcoming SEI Research in Multicore

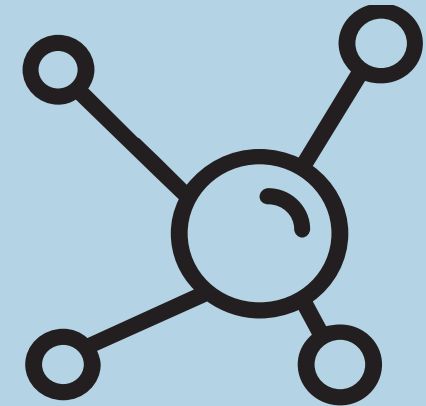
New work in multicore will address challenges in these areas:



Verification



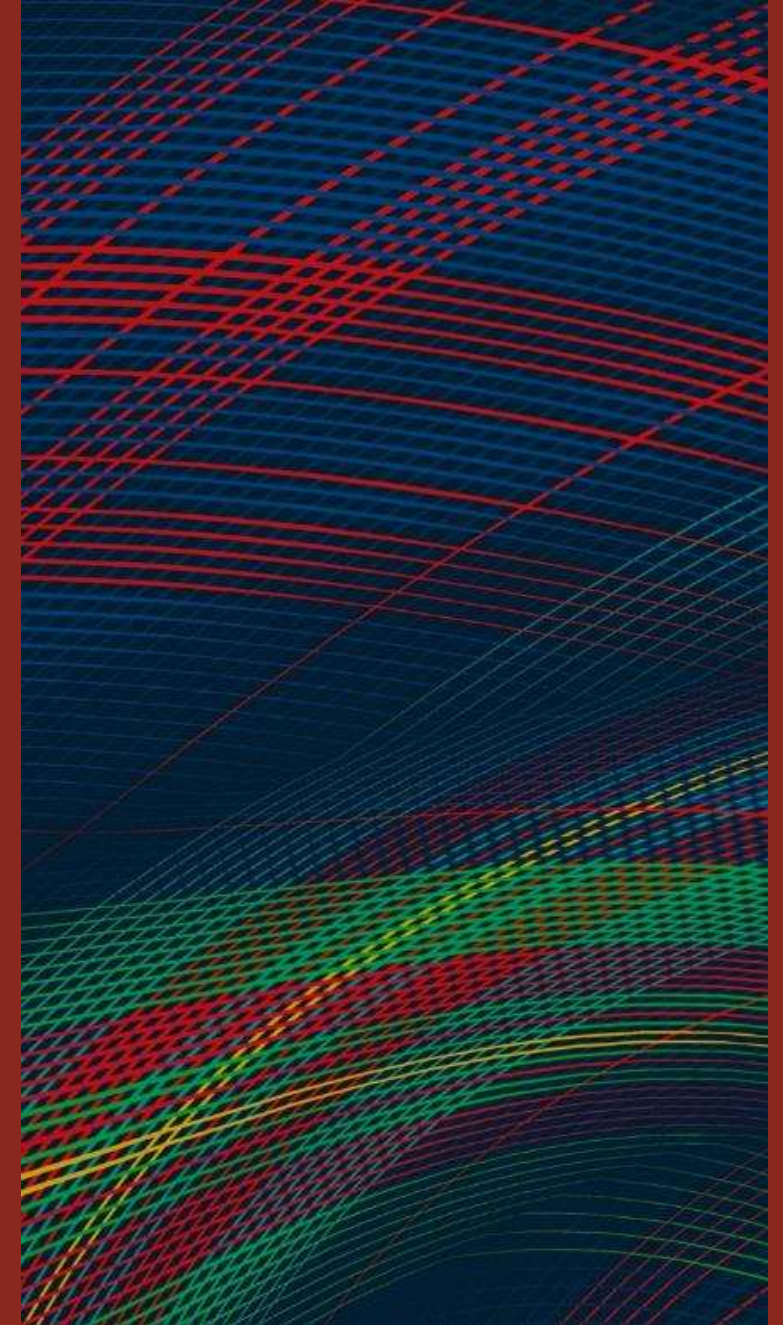
Obtaining Abstractions



Configuration

Kalki: High-Assurance Software-Defined IoT Security

The term "Kalki" is of Sanskrit origin and derived from the Sanskrit word "Kala," which means destroyer of filth or malice and bringer of purity, truth and trust.



Motivation

Problem

Despite the DoD's current use of Internet of Things (IoT) devices in Supervisory Control and Data Acquisition (SCADA) systems, and its interest in using such devices in tactical systems, adoption of IoT by has been slow mainly due to security concerns (e.g., reported vulnerabilities, untrusted supply chains)

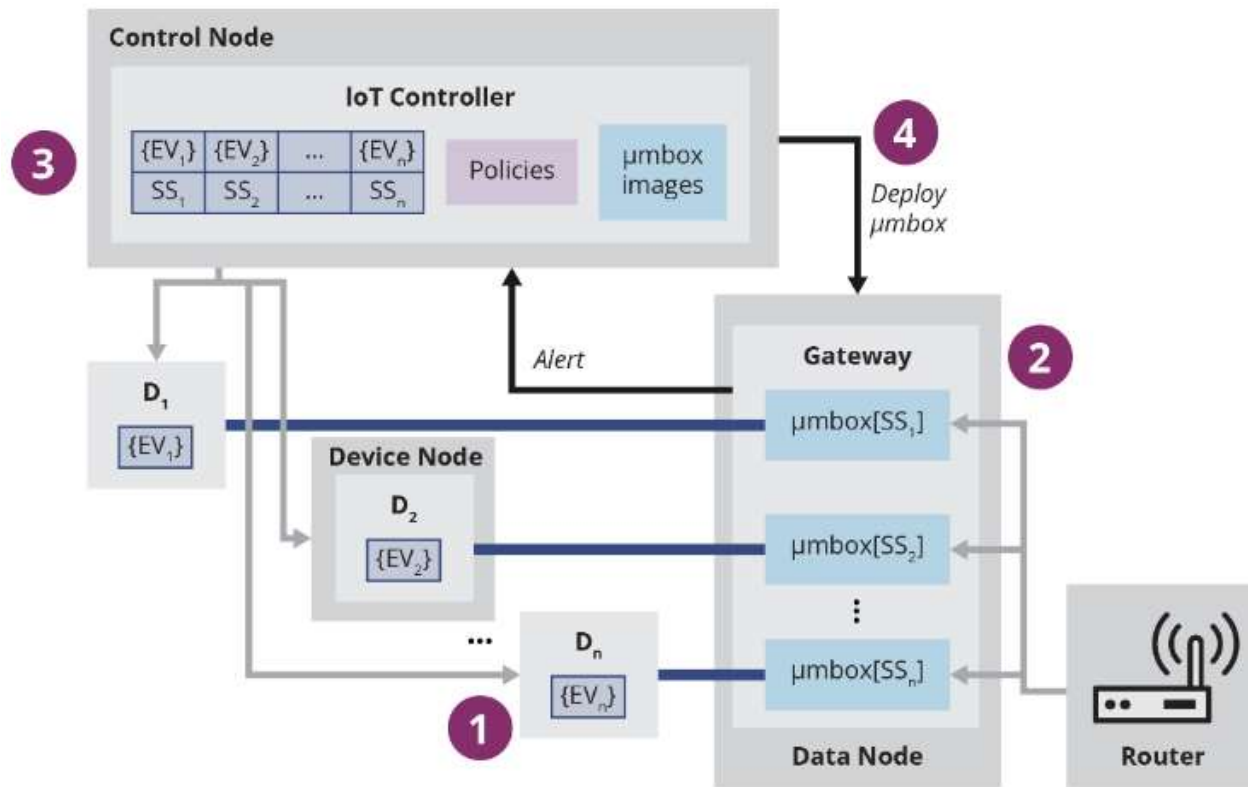
At the same time, DoD recognizes the rapid pace at which the IoT commercial marketplace is evolving, and its urgency to embrace commodity technologies to match its adversaries

Solution

Move part of security enforcement to the network to enable the integration of IoT devices into DoD systems, even if the IoT devices are not fully trusted or configurable, by creating an IoT security infrastructure that is provably resilient to a collection of prescribed threats

The “Software-Defined” Aspect

Use software-defined networking (SDN) and network function virtualization (NFV) to create a highly dynamic IoT security framework



- 1 Each IoT device, D , senses/controls a set of environment variables, EV
- 2 Network traffic to/from each device is tunneled through μ boxes that implement the desired network defense for the device's current security state
 - $\mu\text{box}[SS_1] = \text{Firewall}$
 - $\mu\text{box}[SS_2] = \text{IPS}, \dots$
- 3 IoT controller maintains a shared statespace composed of $\{EV\}$ and security state (SS) for each device
 - $SS = \{\text{Normal}, \text{Suspicious}, \text{Attack}\}$
- 4 Changes in the shared statespace are evaluated by policies and may result in the deployment of new μ boxes

The “High Assurance” Aspect

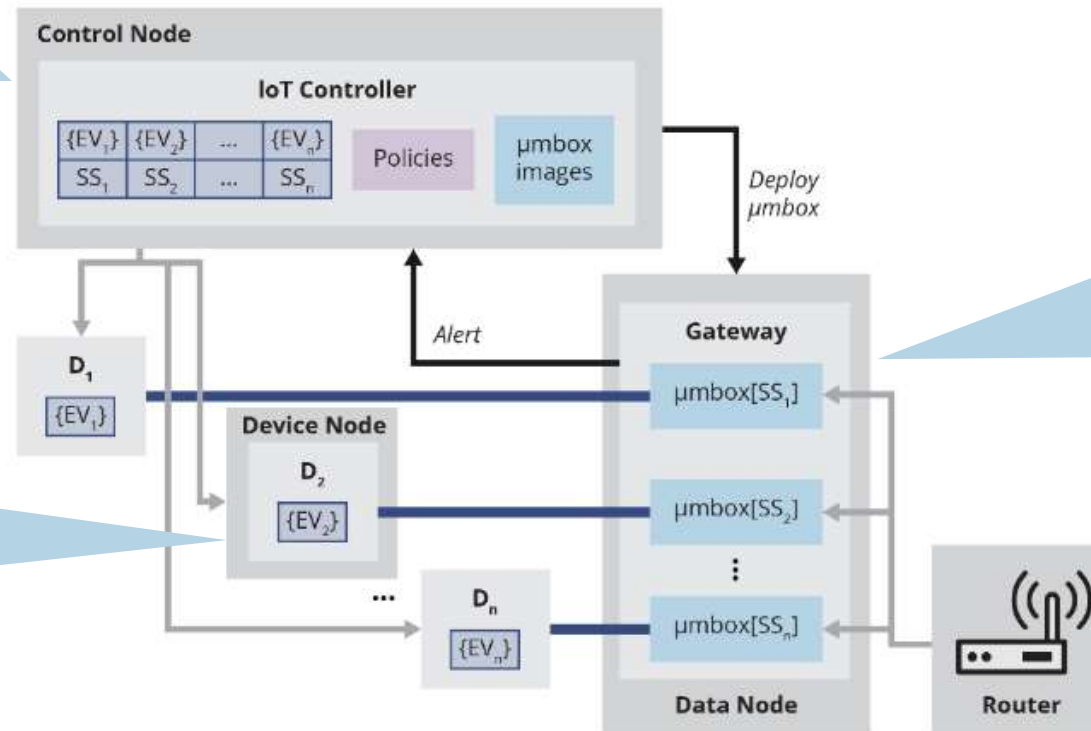
Use überSpark (a framework for building secure software stacks) to incrementally develop and verify security properties of elements of the software-defined IoT security infrastructure

Control Node Properties

- Policy data integrity
- μ box image storage integrity

Device Node Properties

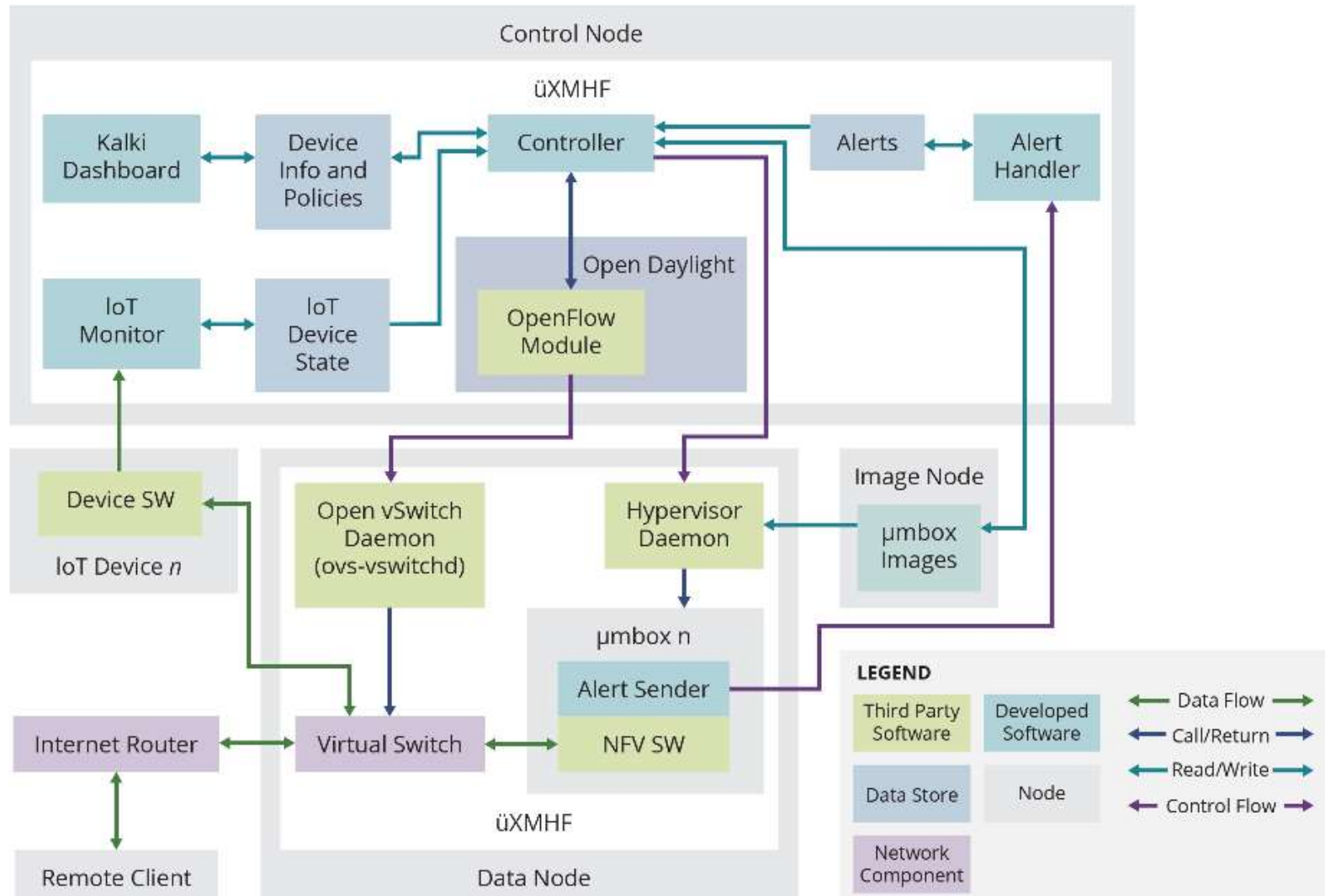
- Attestation
- Authenticated channel of communication with the IoT Controller



Data Node Properties

- Isolation between μ boxes of different trust levels: trusted, untrusted, and verified
- μ box deploy-time integrity

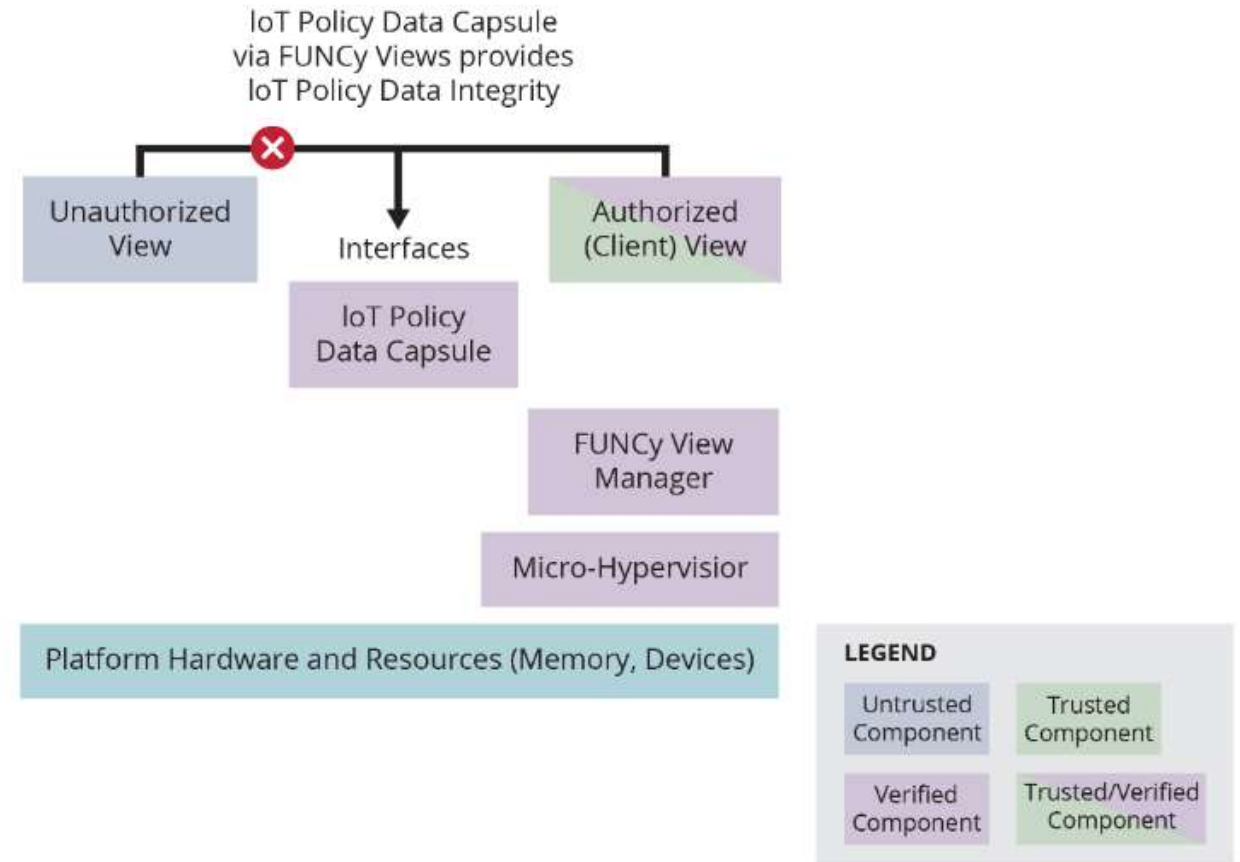
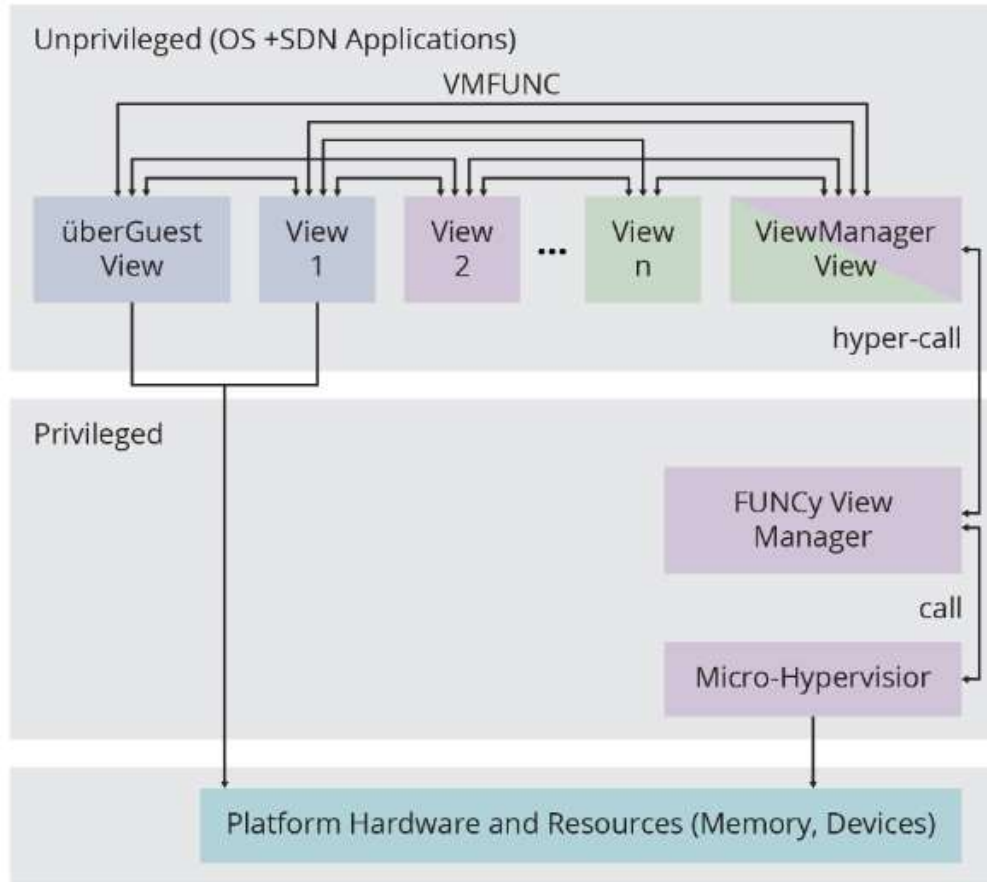
Year 1 Accomplishments—1



Initial Architecture and Prototype of the IoT Security Framework (focus on Control Node)

Year 1 Accomplishments—2

FUNCy Views (Secure) System Architecture: Hardware-assisted, Low-latency, Low-TCB, Legacy Code Compartmentalization on x86 platforms

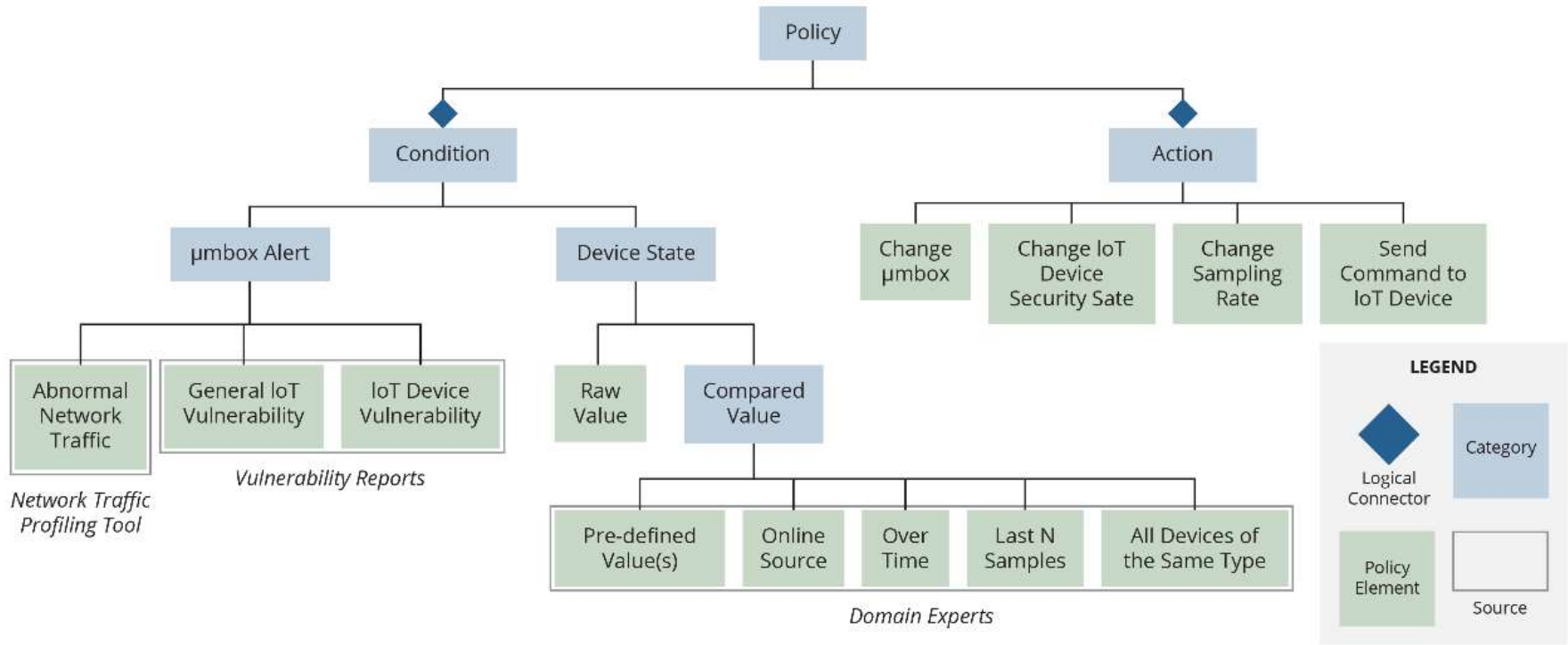


LEGEND

Untrusted Component	Trusted Component
Verified Component	Trusted/Verified Component

Year 1 Accomplishments—3

Policy Model

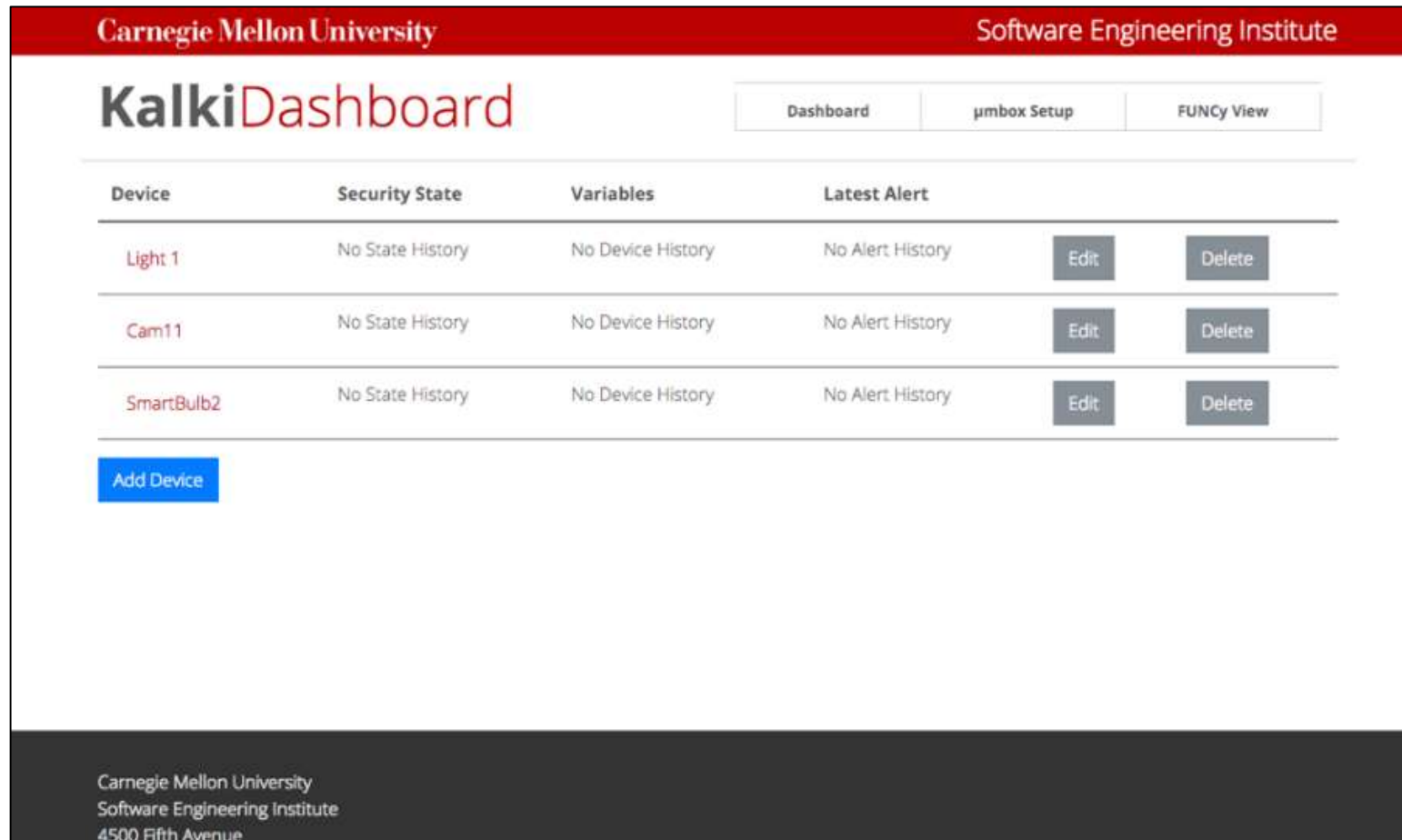


A policy is

- the set of conditions that indicate a change in the security state of an IoT device, and
- the set of actions taken when the conditions are met

Year 1 Accomplishments—4

Dashboard



The screenshot displays the KalkiDashboard interface. At the top, there is a red header bar with "Carnegie Mellon University" on the left and "Software Engineering Institute" on the right. Below the header, the title "KalkiDashboard" is prominently displayed in a large, dark font. To the right of the title, there are three navigation tabs: "Dashboard" (which is active), "µmbox Setup", and "FUNCy View".

The main content area features a table with the following columns: "Device", "Security State", "Variables", and "Latest Alert". Each row represents a device and includes "Edit" and "Delete" buttons. Below the table, there is a blue button labeled "Add Device".

Device	Security State	Variables	Latest Alert		
Light 1	No State History	No Device History	No Alert History	Edit	Delete
Cam11	No State History	No Device History	No Alert History	Edit	Delete
SmartBulb2	No State History	No Device History	No Alert History	Edit	Delete

At the bottom of the dashboard, there is a dark grey footer bar containing the text: "Carnegie Mellon University", "Software Engineering Institute", and "4500 Fifth Avenue".

Year 1 Accomplishments—5

Initial Threat Model

#	Threat	In Scope?
1	Attacker finds a way to deploy the wrong μ mbx for an IoT device given its security posture at a point in time	Y
2	Attacker loads malicious firmware/software on the IoT device	Y
3	Attacker finds a way to circumvent μ mbx	Y
4	Attack from a μ mbx to another	Y
5	Attacker compromises software running inside a μ mbx	Y
6	Attacker identifies combination of inputs that can cause the FSM (internal policy representation) to lead to undesirable results	N
7	Attacker compromises communication between μ mbx and IoT device	Y
8	Attacker compromises communication between μ mbx and IoT controller	Y

What's Next



Extend architecture and prototype of the IoT Security Framework



Implement additional security properties



Add features for easier μ box creation and IoT device integration

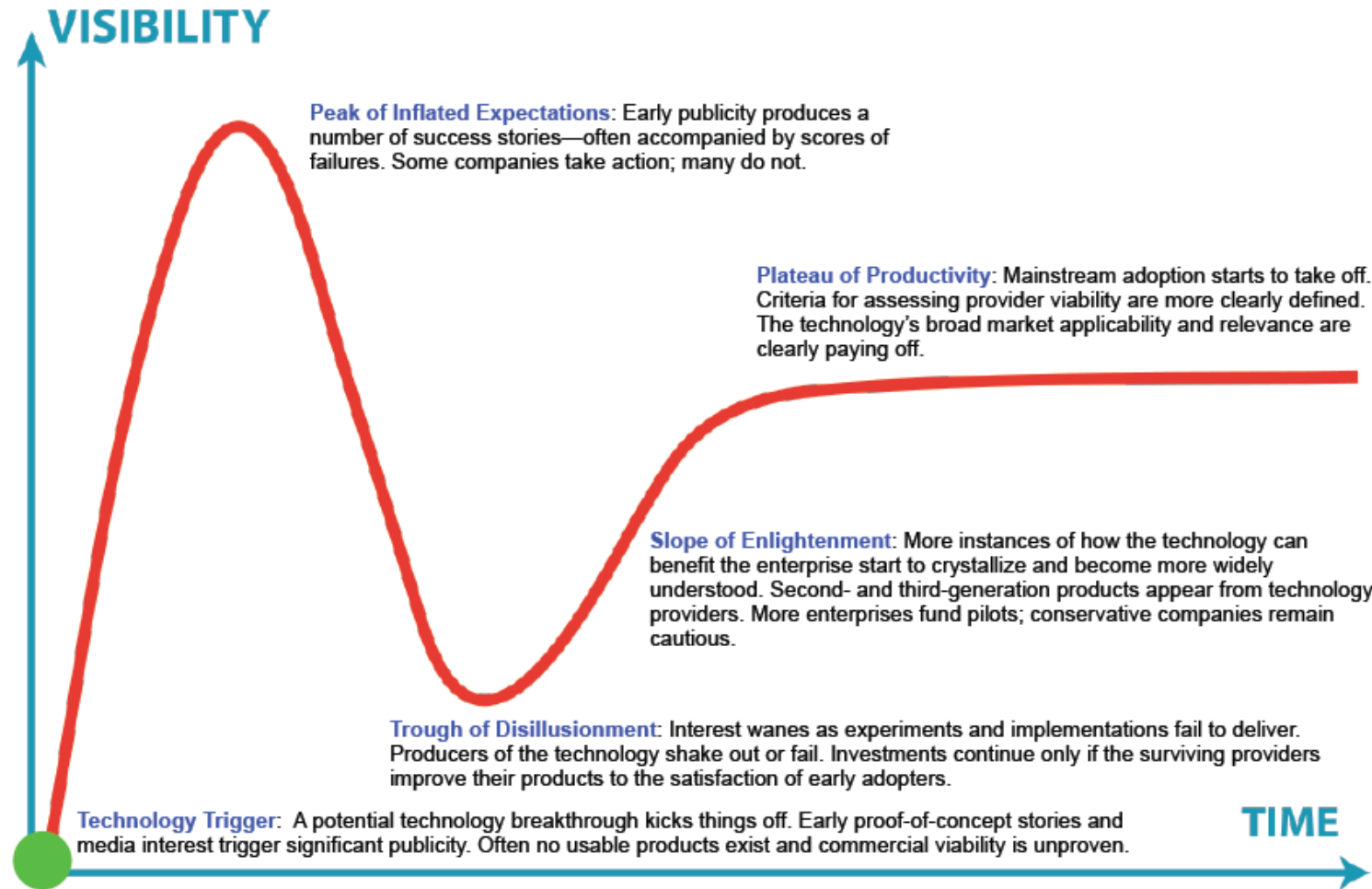


Participate in exercises



Transition IoT Security Framework and lessons learned

Understanding Technology Hype Cycles is Important for Leveraging Emerging Changes in Computing



Emerging and evolving technologies affect the way that systems are developed, deployed, and acquired

Timely identification, understanding, and adaptation of technologies leads to realizing computational and algorithmic advantage in DoD systems

Source: Gartner, Hype Cycle for Emerging Technologies. Credit: © 2018 Gartner, Inc. and/or its Affiliates. All Rights Reserved.