Carnegie Mellon University Software Engineering Institute

Research Review 2021

Rapid Certifiable Trust

October 2021

Dionisio (Dio) de Niz

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

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.

DM21-0814

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution.

Rapid Capability Fielding

Fast

- **DoD Rapid Capability Offices (Air Force, Army, Strategic Capability Office)**
- **Maximize reuse**
  - Open source
  - Ever increasing complexity

Multiply Human Capabilities

- **Learning Autonomy**
  - Continuously adapting behavior

BUT Trustworthy

- **Fast validation**
- **Safety-critical interactions with the physical world (Cyber-Physical System)**
  - Physics
  - Timing
  - Logic

Rapid Certifiable Trust

Fast Trustworthy Validation

- Automation with formal verification

Complexity

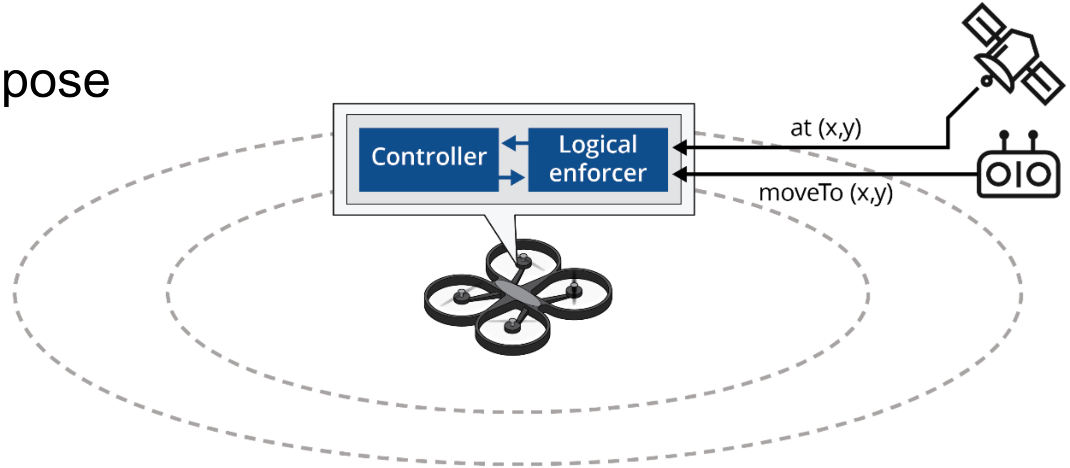
- Traditional Verification Does Not Scale

Adapting Behavior

- Cannot verify at design time

Scalable Enforcement-Based Verification

- Leave Most Code **Unverified**
- Add **simpler (verifiable)** runtime enforcer to make algorithms predictable
- Formally: specify, verify, and compose multiple enforcers
  - Logic: replaces unsafe values
  - Timing: at right time
  - Physics: verified physical effects
- Enforcer protection against failures/attacks
- Resilient to failures / dynamic environment

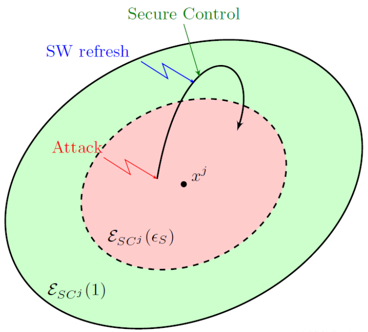


Verifying Physics (Control Theory)

Carnegie Mellon University Software Engineering Institute

Recoverable Set: $\varepsilon_{SC^j}(1)$

Safety Set: $\varepsilon_{SC^j}(\epsilon_s) \triangleq \epsilon_s\, \varepsilon_{SC^j}(1)$

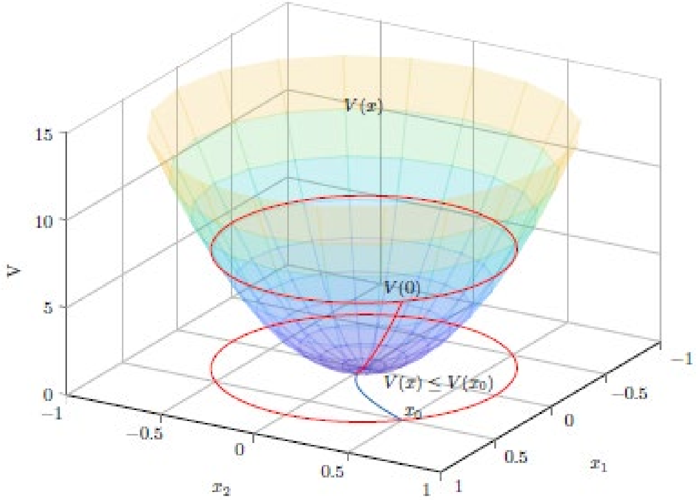


Controlled System: $\dot{x} = f_\varphi(x) \triangleq f(x, \varphi(x))$
Lyapunov Function: $V_\varphi : \mathbb{R}^n \to \mathbb{R}$, $\mathcal{N}_{V_\varphi}(x_{eq}) \subseteq \mathcal{N}_\varphi(x_{eq})$,
$V_\phi(x_{eq}) = 0$ and $\forall x \in \mathcal{N}_{V_\varphi}(x_{eq}) - \{x_{eq}\}$: (i) $V_\varphi(x) > 0$,

$$\dot{V}_\varphi(x) = \frac{\partial V}{\partial x} \cdot f_\varphi(x) < 0$$

Lyapunov level set: For $\epsilon > 0$,

$$\mathcal{E}_\varphi(\epsilon) = \{x \in \mathcal{N}_{V_\varphi}(x_{eq}) | V_\varphi(x) \leq \epsilon\}. \qquad \epsilon \leq 1$$

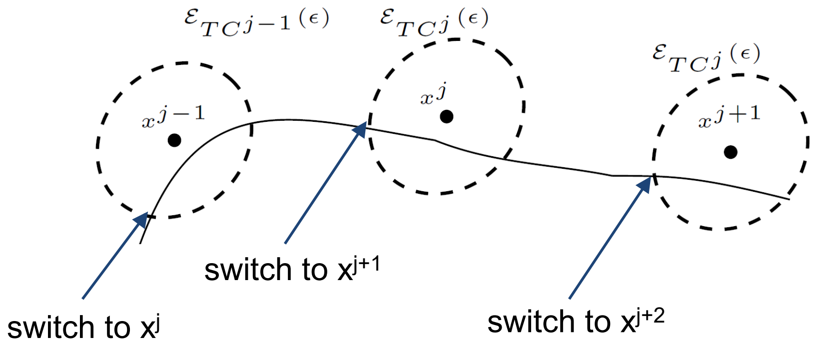


R. Romagnoli, B. H. Krogh, B. Sinopoli. Design of Software Rejuvenation for CPS Security Using Invariant Sets. ACC 2019

Analysis of Mission Progress

Idea:

Provide a sequence of waypoints that represent a sequence of equilibrium points around which we define the Safe Set.



Goal:

- Safely transition from one waypoint to the next
- Liveness (in the case of no errors)

R. Romagnoli, B. H. Krogh, B. Sinopoli. Safety and Liveness of Software Rejuvenation for Secure Tracking Control. ECC 2019.

Analysis of Mission Progress Enforcing Unsafe Behavior

6 DOF $\Rightarrow$ 12 state variables

$$\ddot{p}_x = -\cos\phi\sin\theta\frac{F}{m}$$

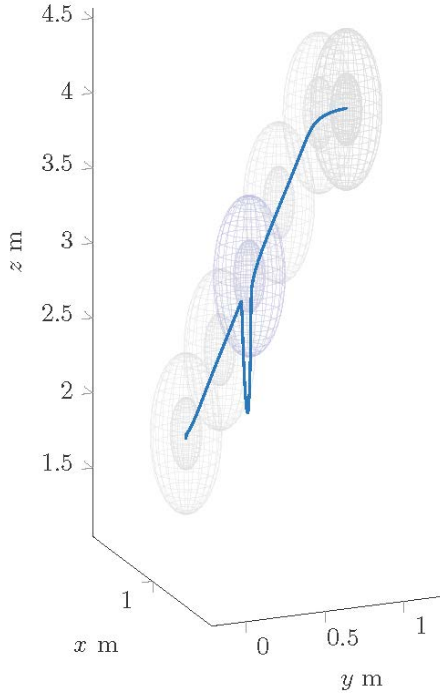
$$\ddot{p}_y = \sin\phi\frac{F}{m}$$

$$\ddot{p}_z = g - \cos\phi\cos\theta\frac{F}{m}$$

$$\ddot{\phi} = \frac{1}{J_x}\tau_\phi$$

$$\ddot{\theta} = \frac{1}{J_y}\tau_\theta$$

$$\ddot{\psi} = \frac{1}{J_z}\tau_\psi.$$

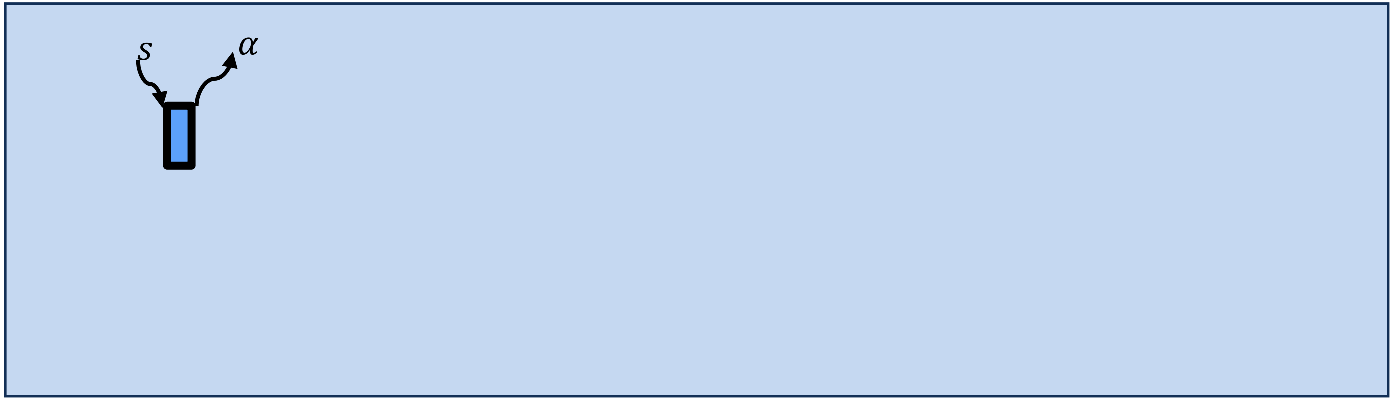


Linear design:

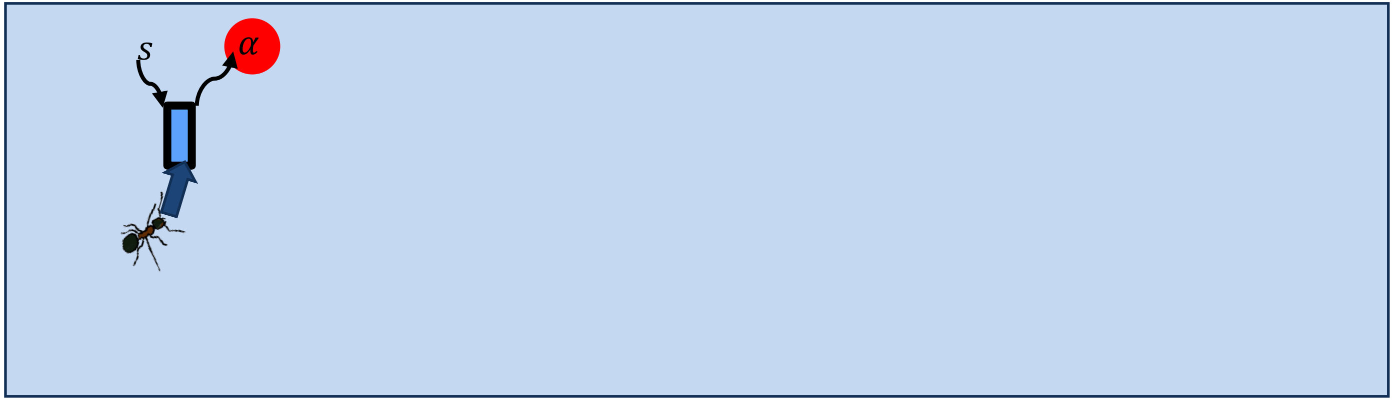
- linearize at equilibrium
- assume full state available
- LQ state feedback design
- reference points = equilibrium states

Drone Experiment

Enforcing Unverified Components



Enforcing Unverified Components



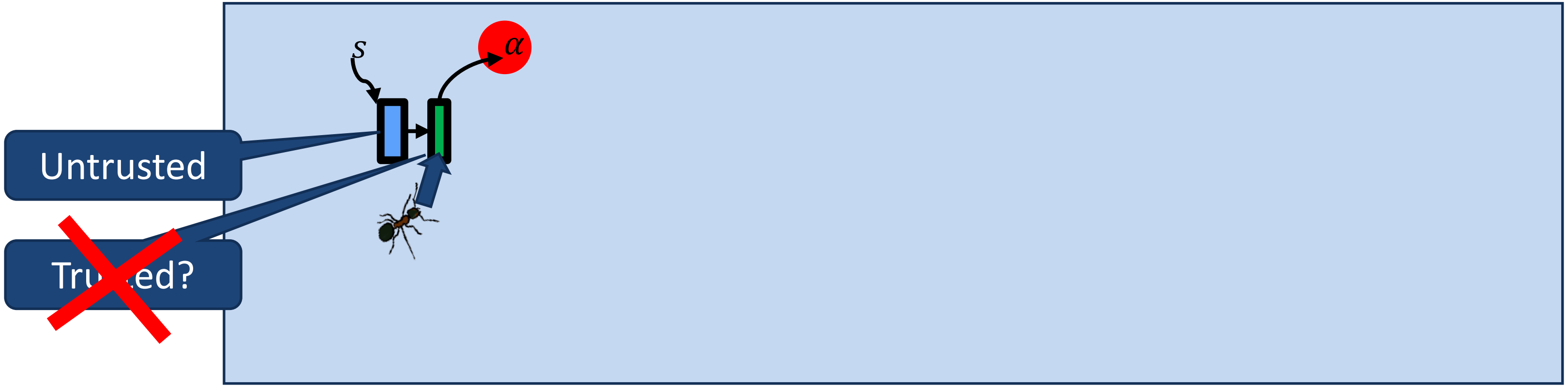
Ant illustration by Jan Gillbank, license by Creative Commons Attribution 3.0 Unported

Enforcing Unverified Components

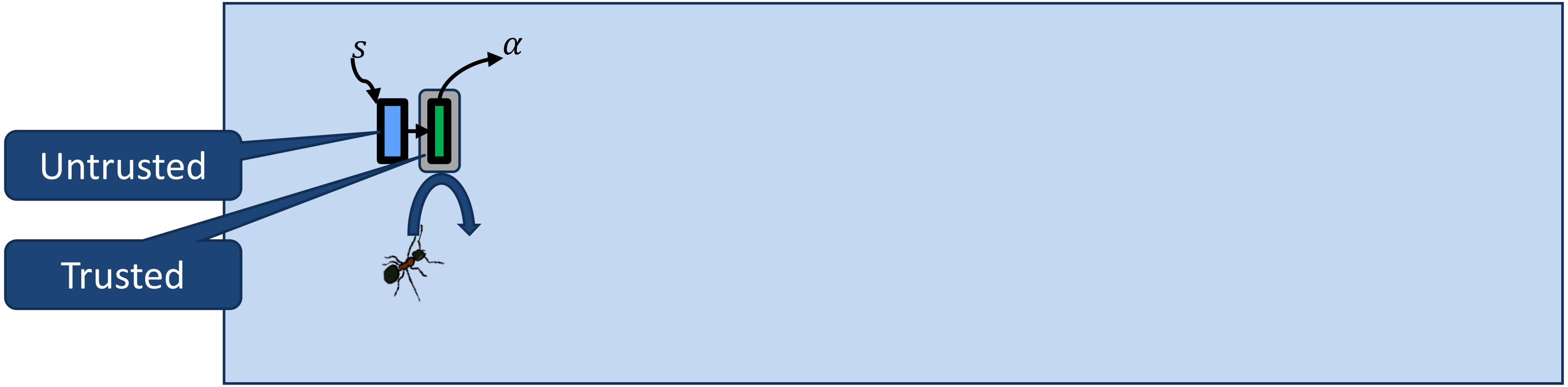
Carnegie Mellon University Software Engineering

Values

But enforcer can be corrupted (bug or cyber attack)

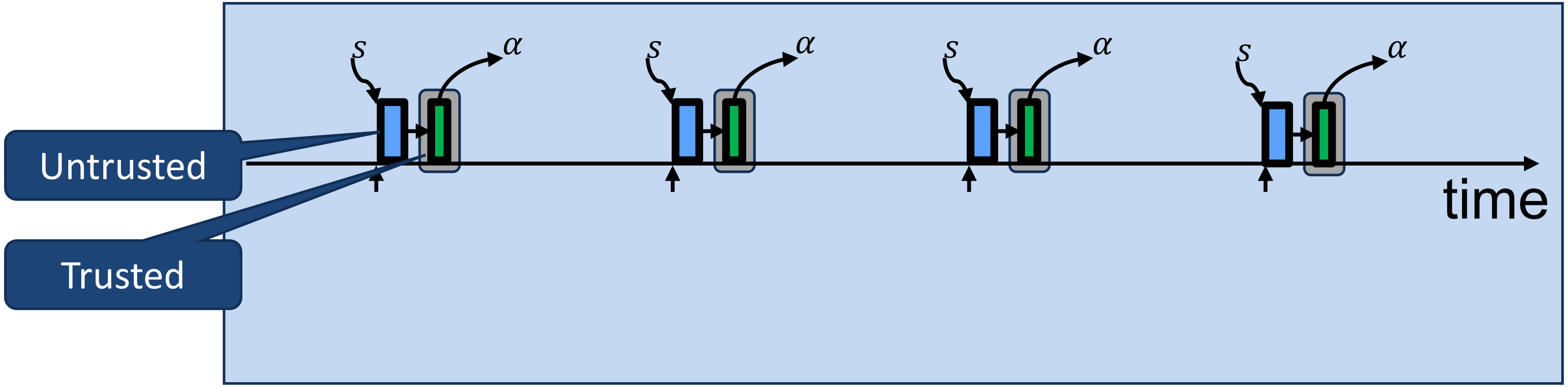


Add Memory Protection

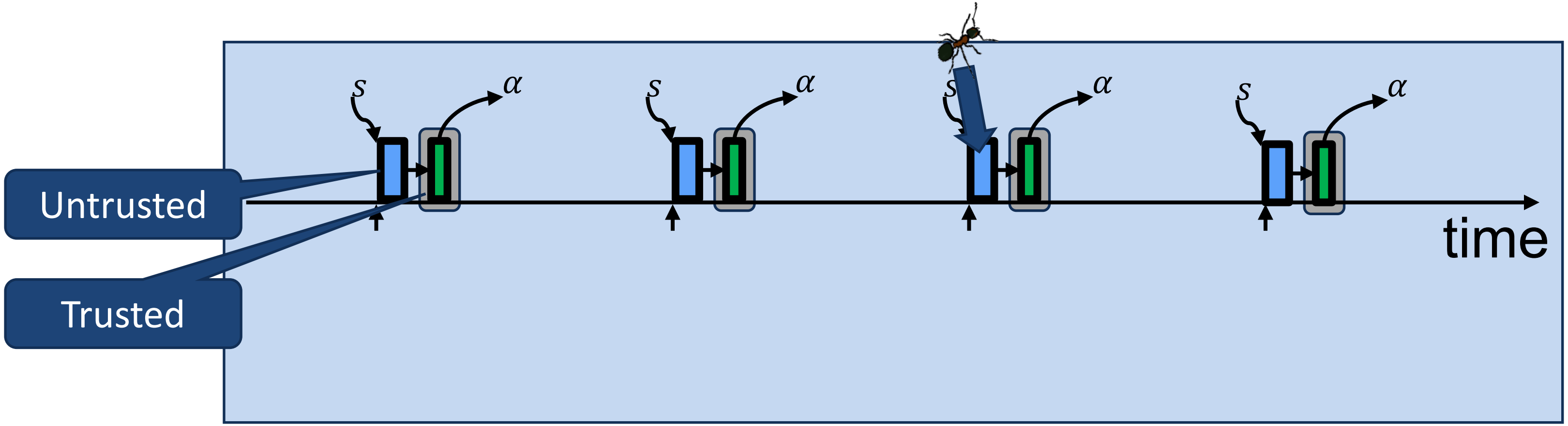


Trusted = Verified & Protected

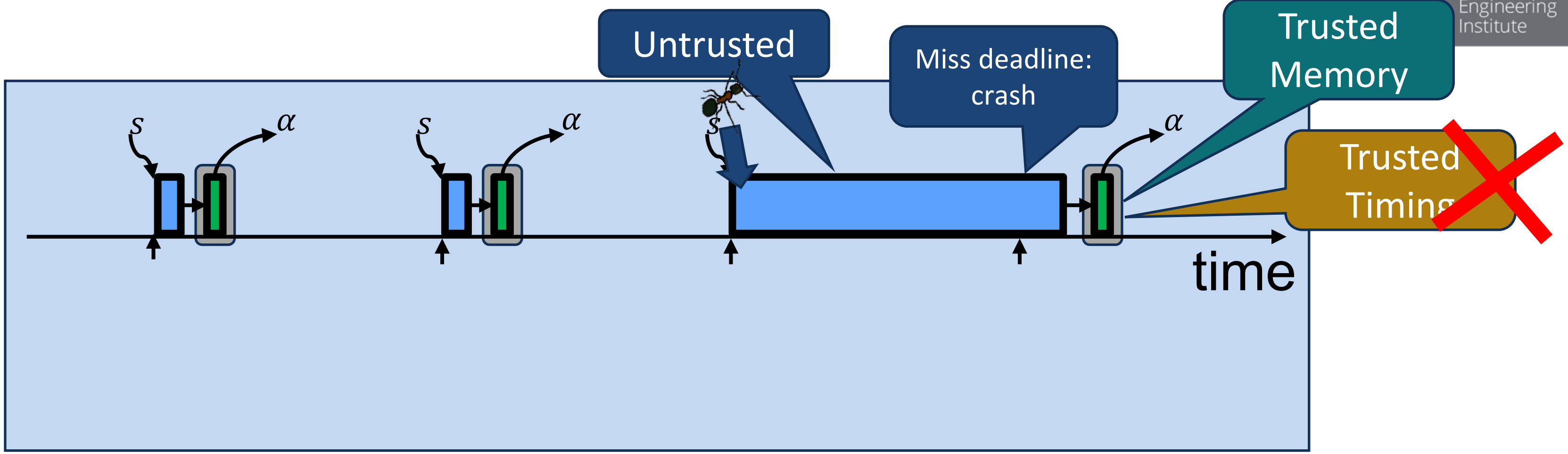
Periodic Execution Must Finish by Deadline



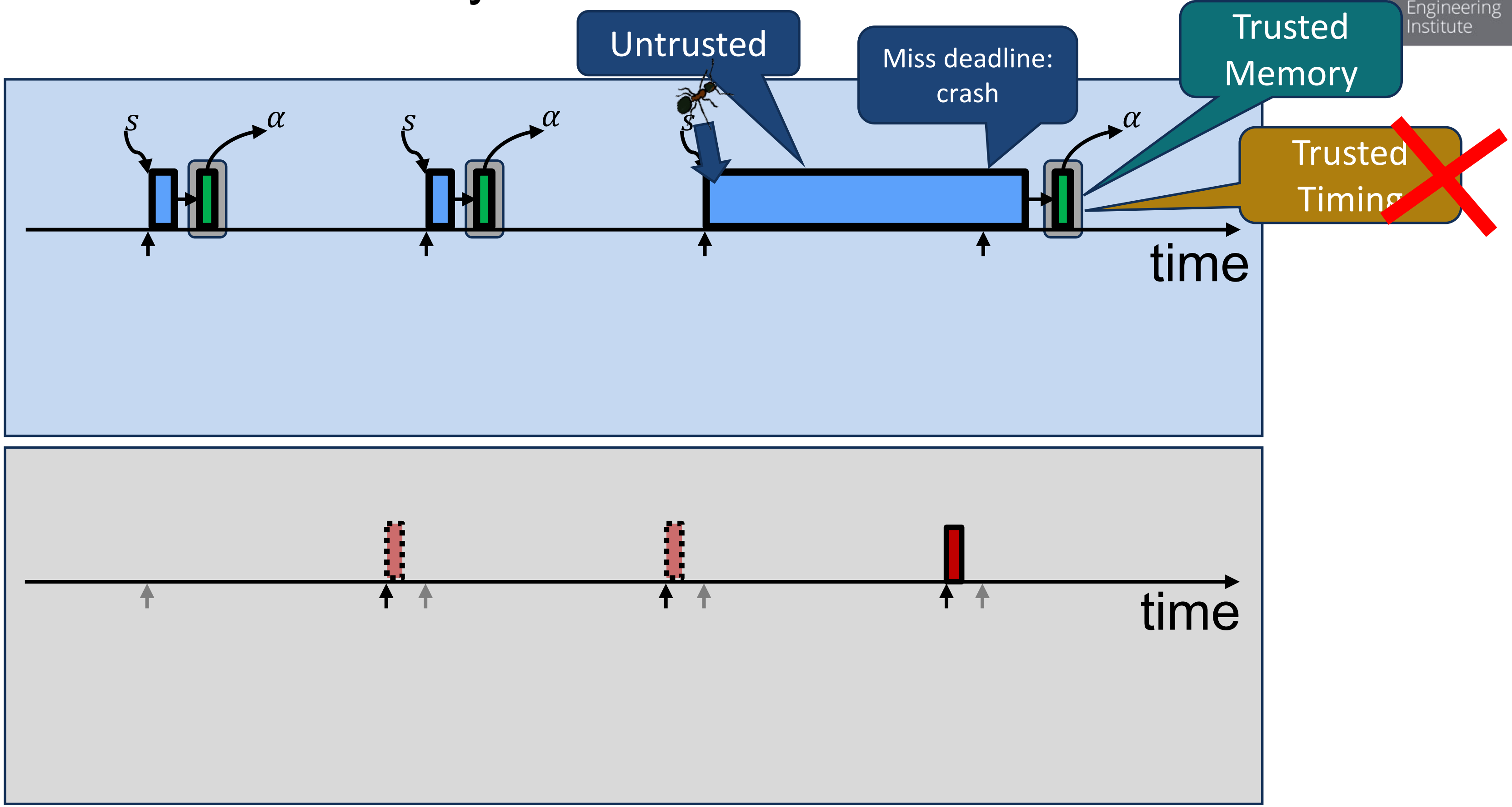
Periodic Execution Must Finish by Deadline



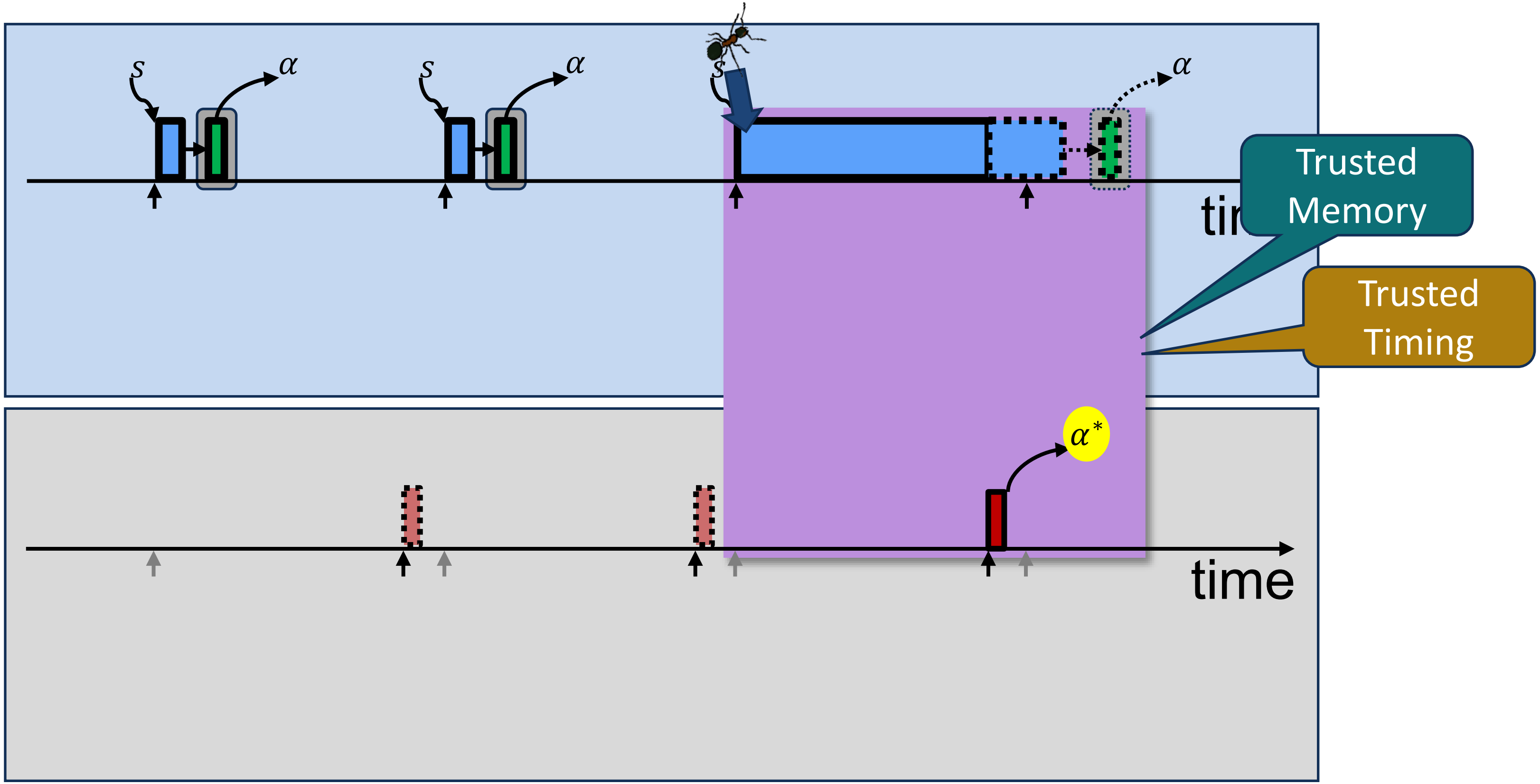
Periodic Execution Finish by Deadline



Periodic Execution Finish by Deadline



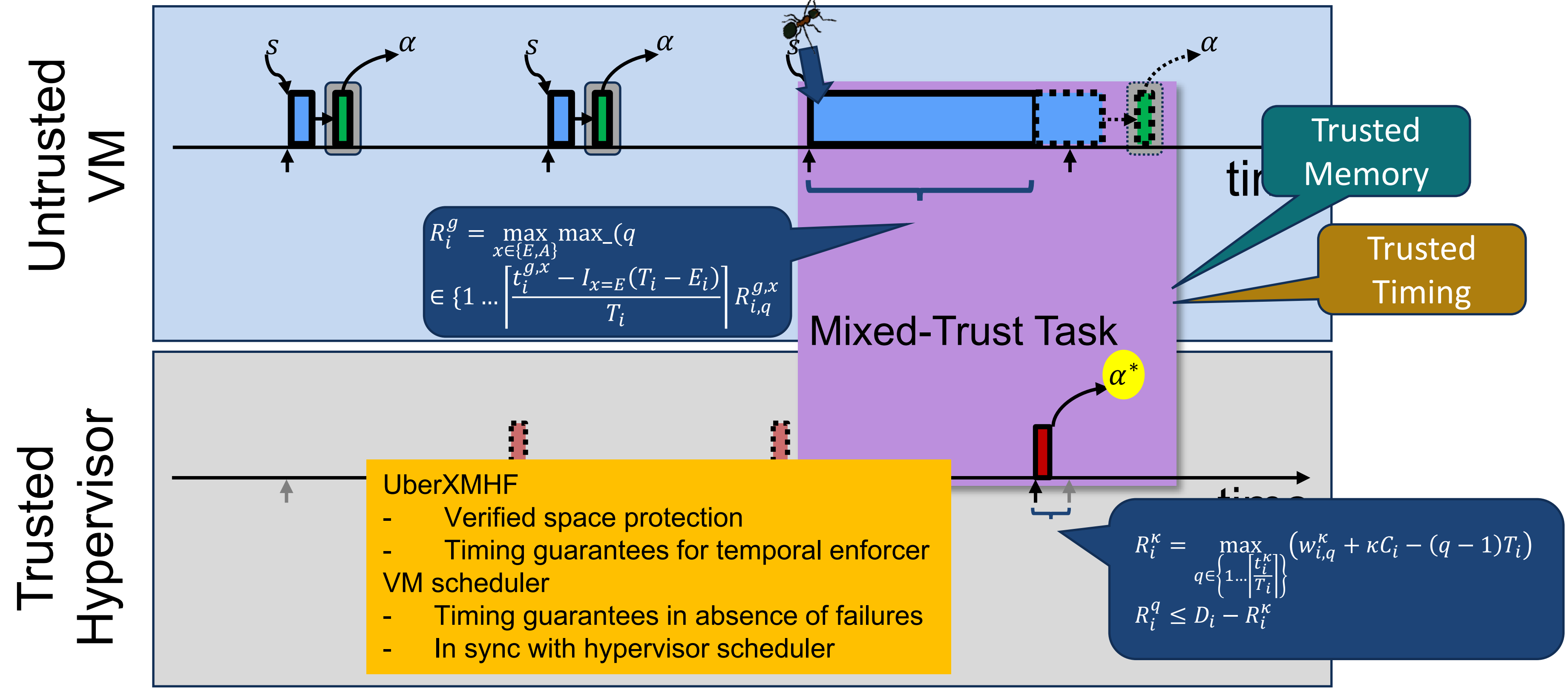
Periodic Execution Finish by Deadline



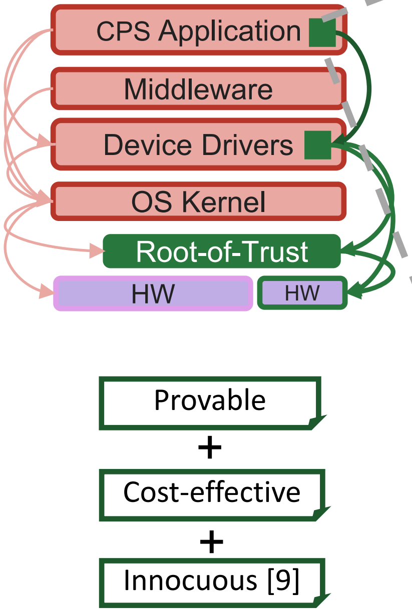
Real-Time Mixed-Trust Computation

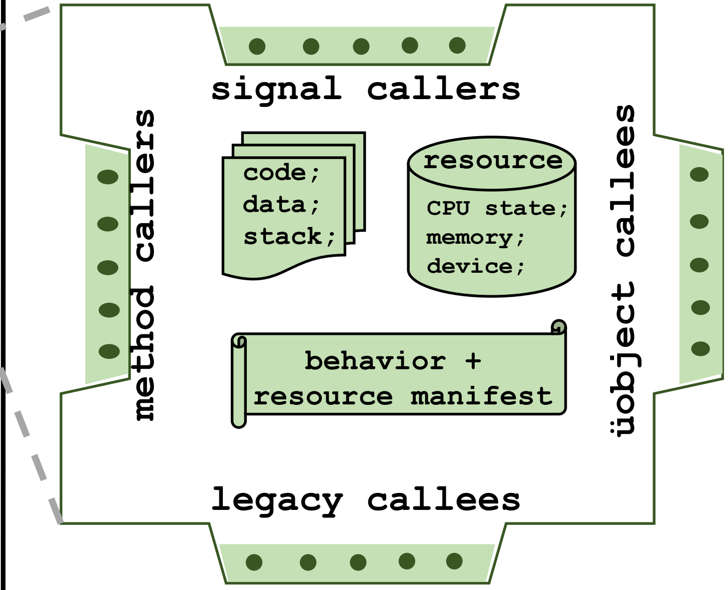
D. de Niz, B. Andersson, M. Klein, J. Lehoczky, A. Vasudevan, H. Kim, & G. Moreno. Mixed-Trust Computing for Real-Time Systems. IEEE RTCSA, 2019.

R.Martins, M. McCall, D. de Niz, A. Vasudevan, B. Andersson, M. Klein, J. Lehoczky, and H. Kim. Formal Verification of a Mixed-Trust Synchronization Protocol. RTNS, 2021.



Verified Protection at Hypervisor, Kernel, Application



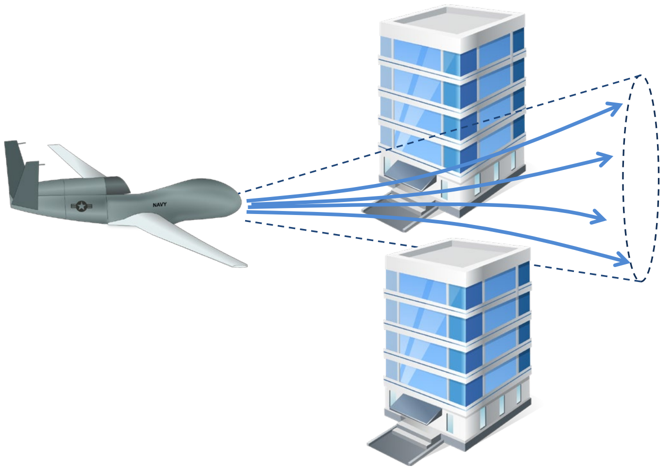


- Singleton object guarding exclusive indivisible system resource
- Principled entry, interruption, legacy code invocations and üobject invocations
  - execution trace respecting program control-flow enables use of state-of-the-art program verification tools
  - facilitate AG reasoning and composition
- Call-return Interfacing
  - Handle various CHIC programming idioms
- Resource Interface Confinement
  - Resource protection and access control
  - Support Shared memory concurrency -> multi-threaded execution and reasoning

A. Vasudevan, P. Maniatis, R.Martins, S. Chaki. Practical, Provable, End-to-End Guarantees at the Edge. USENIX Workshop on Hot Topics in Edge Computing 2020

M. McCormack, A. Vasudevan, G. Liu, V. Sekar
Formalizing an Architectural Model of a Trustworthy Edge IoT Security Gateway
RTCSA 2021

Predictive Mixed-Trust



Enforcement Lookahead

- Non-Holonomic Vehicles
  - Cannot switch direction instantaneously
  - Require computing safe trajectory set (safety cone)

- Enforce safety cone

Balancing safety cone calculation and enforcement

- How far into the future to calculate
- Leave enough CPU computation to for safety enforcement

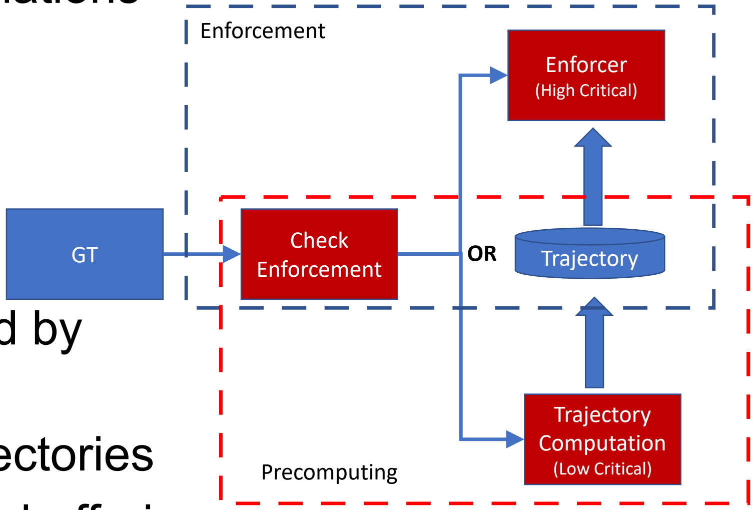
Images by Jan Helebran and OpenClipart-Vectors t from Pixabay

Predictive Mixed-Trust Architecture

Carnegie Mellon University Software Engineering Institute

Safety Enforcement only needed in hazardous deviations

- If enforcement is needed then it can pause trajectory generation
- Considered High-Criticality task



Trajectory generation can use CPU cycles not used by safety enforcement

- But ensure that we have enough lookahead trajectories
- Precomputed trajectories work as a computation buffering

Predictive Mixed-Trust Scheduling

Balance Trajectory Production and Consumption

- *Production rate:* G_i^d, *Consumption rate:* S_i^e, *Enforcement interarrival:* I_i
- $G_i^d(I_i - 1) - S_i^e \geq 0$

Verify that the enforcement worst-case response time meet the deadline

- $R_i^p = \kappa C_i^p + \sum \left\lceil \frac{R_i^p}{T_j} \right\rceil \kappa C_j^p - \left\lfloor \frac{R_i^p}{I_j T_j} \right\rfloor \left(\kappa C_j^p - \kappa C_j^e \right) \leq D_i$

D. de Niz, B. Andersson, H. Kim, M. Klein, and J. Lehoczky.
Toward Precomputing in Real-Time Mixed-Trust Scheduling.
Work-in-Progress. IEEE RTSS 2020

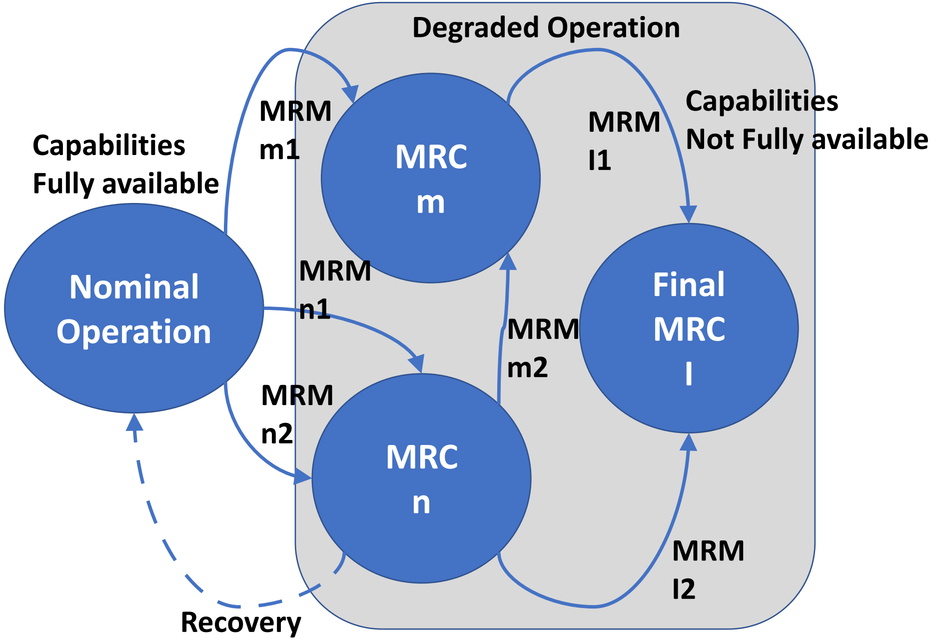
Resilient Real-Time Mixed-Trust

CPS need to adapt to failures/environment

- Daimler: Safety First for Automate Driving[1]
  - Built to preserve safety across failures
    - Minimum Risk Maneuver (MRM) to transition to degraded Mode or Minimal Risk Condition (MRC)

Resilient Real-Time Mixed Trust

- Add enforcer to preserve safety
- Use enforcer to execute MRM



[1]Daimler et al. Safety First for Automated Driving. https://www.daimler.com/documents/innovation/other/safety-first-for-automated-driving.pdf, 2019

Collision Avoidance Enforcer Example

LIDAR

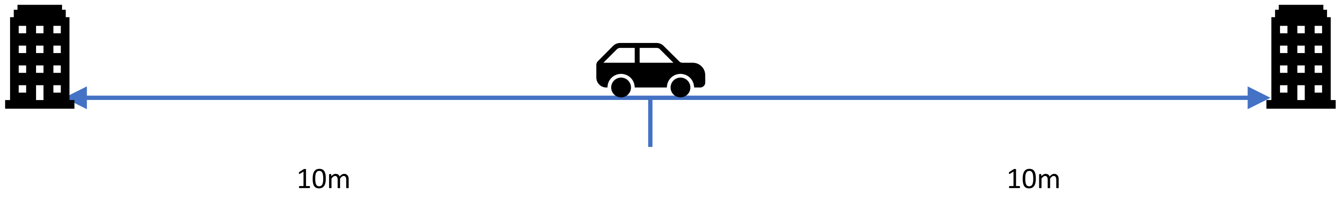
- Detection distance $20\ m$
- Max braking: $-10\frac{m}{s^2}$
- Max speed: $20\frac{m}{s}$

SONAR

- Detection distance $5\ m$
- Max braking: $-10\ m/s^2$
- Max speed: $10\frac{m}{s}$

LIDAR Failure Transitioning Enforcer

- Upon failure: start braking at $-10\frac{m}{s^2}$
- Once speed < $10\frac{m}{s}$ Transition to SONAR enforcer



Resilient Real-Time Mixed-Trust Digraph Model / Scheduling

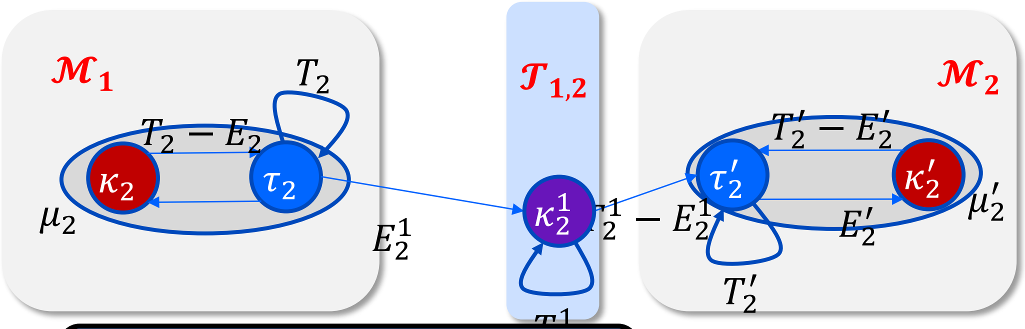
Task: graph $G_i = (V_i, E_i)$

$V_i = \{v_{i,1}, v_{i,2}, \dots\}$

$E_i = \{e_{i,1}, e_{i,2}, \dots\}$

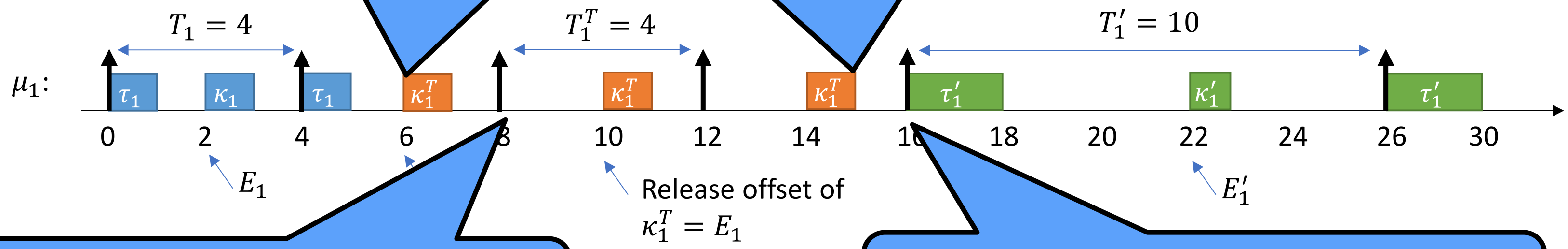
$v_{i,1} = (\tau_{i,1}, \kappa_{i,1}), \dots$

$e_{i,1} = (v_{i,1}, v_{i,2}), \dots$



Hypertask decides to switch to mode κ_1^T

Hypertask decides to switch to mode κ_1'



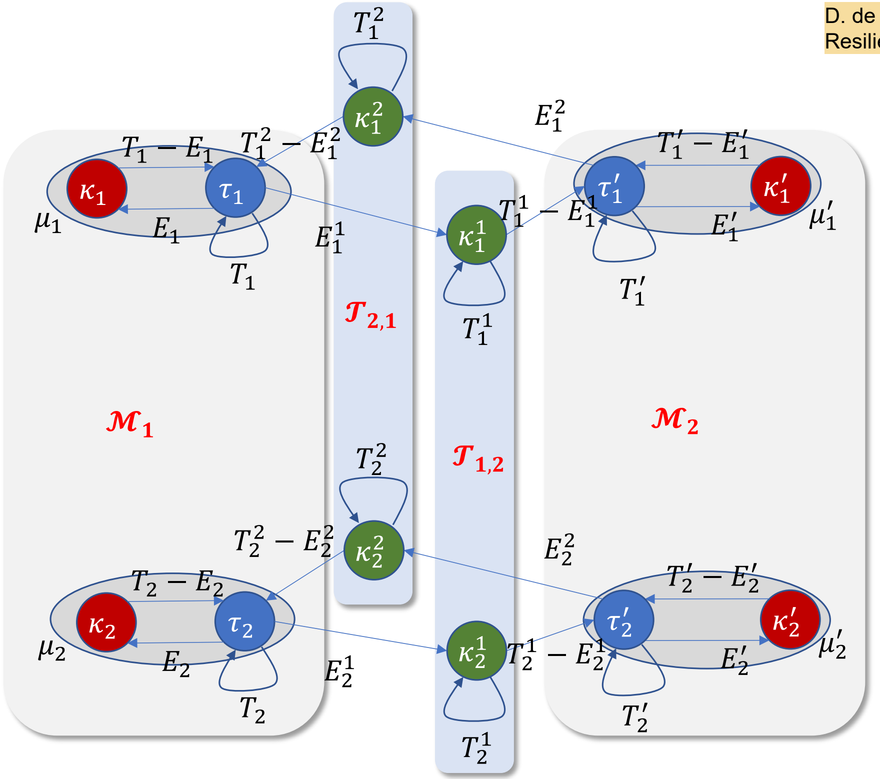
When VM informs HV of new GT job, HV informs GT of new mode (no GT)

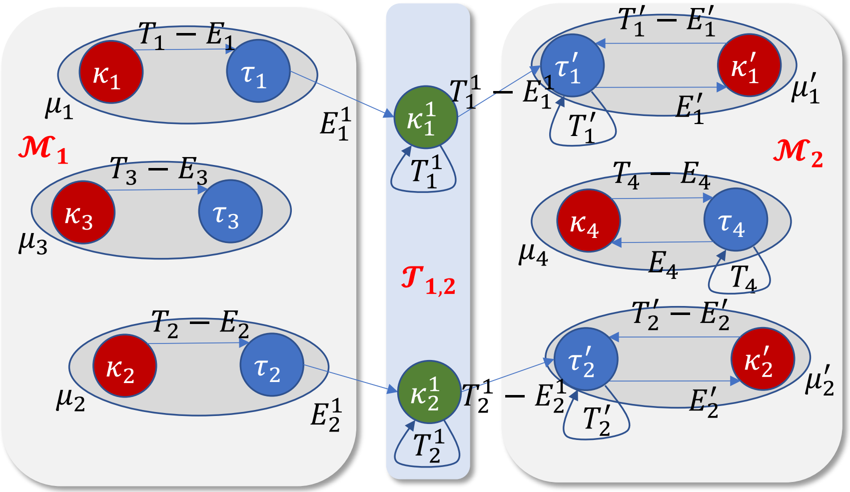
When VM informs HV of new GT job, HV informs GT of new mode (τ_1')

Digraph Transformation for Schedulability

Carnegie Mellon University Software Engineering Institute

D. de Niz, B. Andersson, H. Kim, M. Klein, and J. Lehoczky.
Resilient Mixed-Trust Scheduling. IEEE RTSS 2021.





$$R(v_{i,k}) = MI(v_{i,k}) + C_{i,k},$$

$$MI(v_{i,k}) = P(v_{i,k}) + \sum_{g_j \in hp(i)} rf_{\pi(g_j)}^{v_{i,k}}(MI(v_{i,k}) + J(g_j)),$$

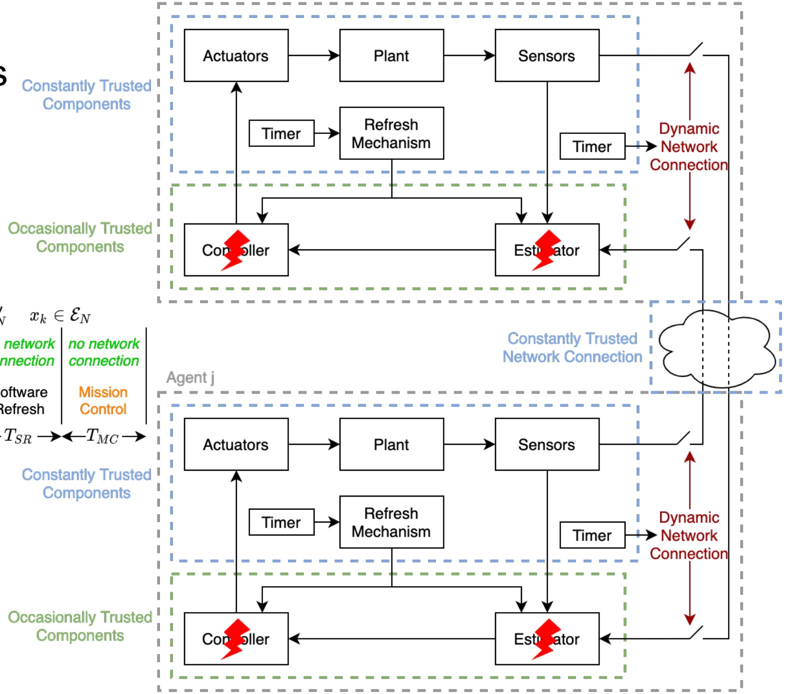
Decentralized Control Enforcement Approach

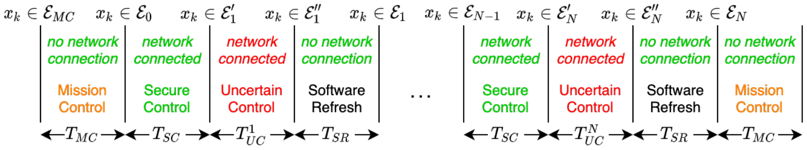
Carnegie Mellon University Software Engineering Institute

Each agent is normally disconnected from the network to prevent misbehaviors

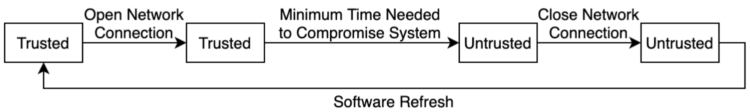
Decentralized systems require occasional communication between agents to ensure overall system safety

Root of trust: secure onboard hardware module





Trust Timeline



Resilience

Decentralized Control Enforcement Verification

Network Connected

$$\dot{x}(t) = Ax(t) + Bu(t)$$

$$x(t) \in \mathcal{C}(x_j) \quad u(t) \in \mathcal{U}$$

equilibrium point: $x_j \in \mathbb{R}^n$

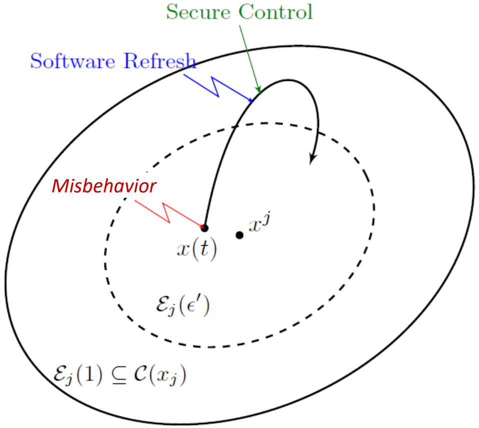
Network Disconnected

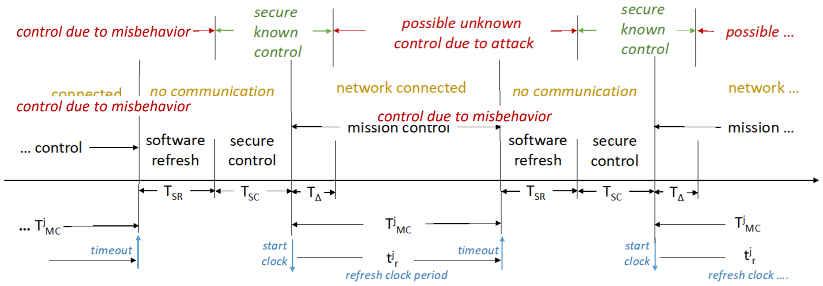
$$\dot{x}(t) = A_f\,(x(t) - x_j)$$

$$u(t) = -Kx(t) \qquad A_f \triangleq A - BK$$

$$\mathcal{E}_j(1) \triangleq \{x | (x - x_j)^T P (x - x_j) \leq 1,\ P \succ 0\} \text{ (safe states)}$$

$$\mathcal{E}_j(\epsilon') \triangleq \{x | (x - x_j)^T P (x - x_j) \leq \epsilon',\ P \succ 0,\ \epsilon' \in [0, 1]\}$$

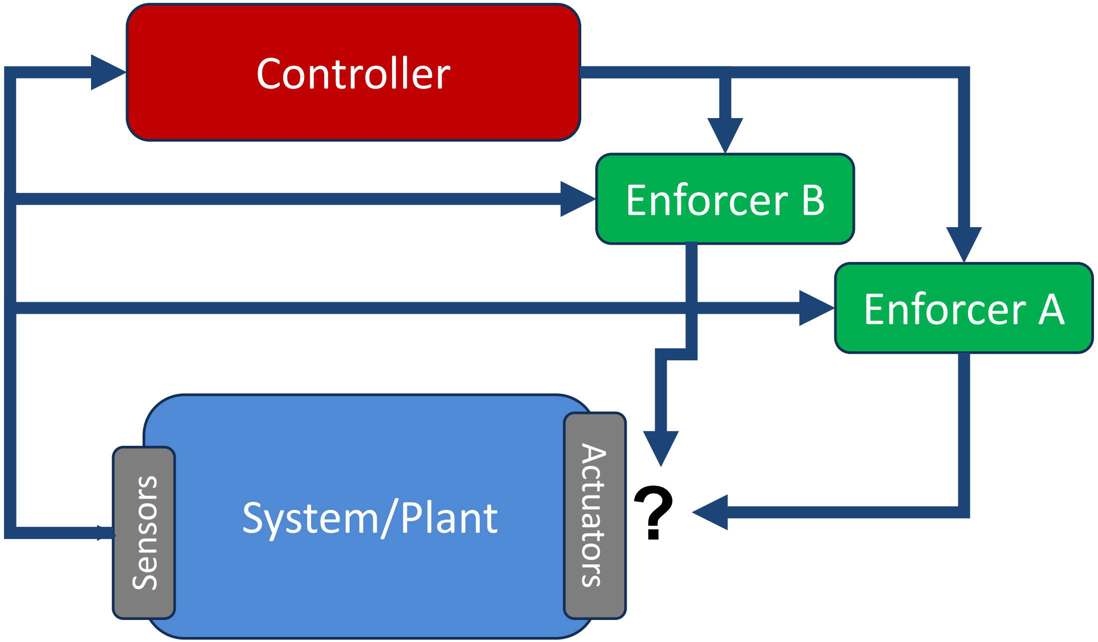


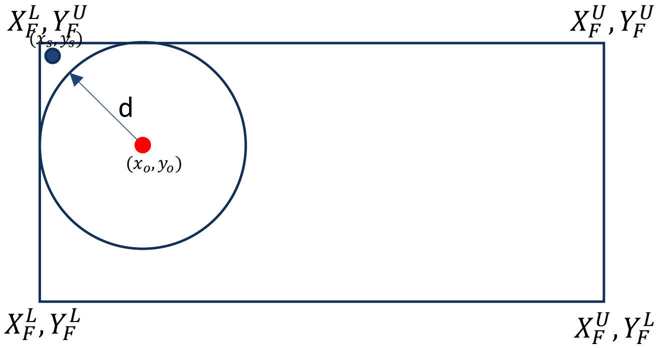


P. Griffioen, R. Romagnoli, B. H. Krogh, and B. Sinopoli, "Resilient Control in the Presence of Man-in-the-Middle Attacks," IEEE ACC, 2021,.
P. Griffioen, R. Romagnoli, B. H. Krogh, and B. Sinopoli, "Decentralized Event-Triggered Control in the Presence of Adversaries," IEEE CDC 2020

R. Romagnoli, P. Griffioen, B. H. Krogh, and B. Sinopoli, "Software Rejuvenation Under Persistent Attacks in Constrained Environments," IFAC 2020.
P. Griffioen, R. Romagnoli, B. H. Krogh, and B. Sinopoli, "Secure Networked Control for Decentralized Systems via Software Rejuvenation," IEEE ACC 2020.

Conflict Resolution of CPS Enforcers





Signal Temporal Logic to Detect and Resolve Conflicts

Signals

- Over D over time T : $s: T \to D$
  - $T \subseteq \mathbb{R}_{\geq 0}$ represent points in time

Signal Temporal Logic

- Extension to Linear Temporal Logic
- Formulas φ
  - u: predicate of the form $f\big(s(t)\big) \geq 0$
  - **Until**: $\varphi_1 U_{[a,b]} \varphi_2$
  - **Eventually**: $F_{[a,b]}\varphi = true\ U_{[a,b]}\varphi$
  - **Always**: $G_{[a,b]}\varphi = \neg F_{[a,b]} \neg \phi$

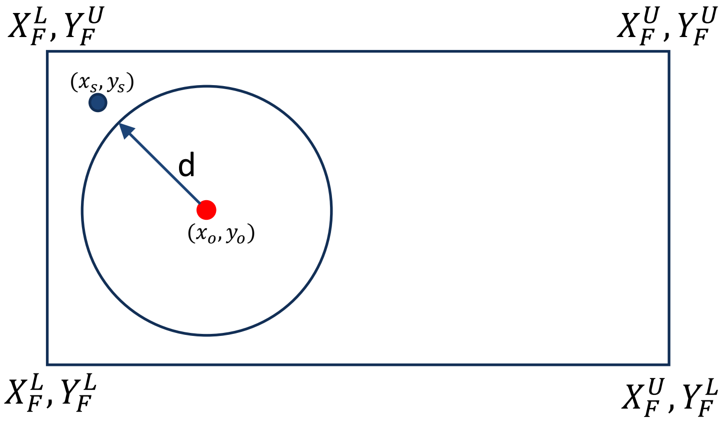
Robustness

- Quantify distance from property violation
- $\rho(\varphi, s, t)$

Always $|x| > 0.5 \Rightarrow$ *after 1 s,* $|x|$ *settles under 0.5 for 1.5 s*

$$\varphi := \mathbf{G}(|x[t]| > 0.5 \to \mathbf{F}_{[0,1]}(\mathbf{G}_{[0,1.5]}|x[t]| < 0.5))$$

Two enforcers that can conflict



Position of drone under control (self): (x_s, y_s)
Position of pursuing drone (other): (x_o, y_o)

Signal $s(t) = (x_s, y_s, x_o, y_o)$

Enforcer 1

- Guarantee:

$$\varphi_1\colon \sqrt{(x_s(t) - x_o(t))^2 + (y_s(t) - y_o(t))^2} - d \geq 0$$

- Robustness:

$$\rho(\varphi_1, t, s) = \sqrt{(x_s(t) - x_o(t))^2 + (y_s(t) - y_o(t))^2} - d$$

Enforcer 2

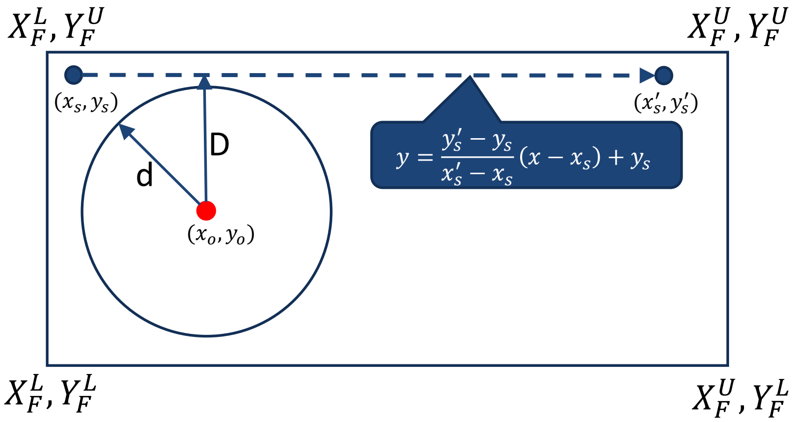
- Guarantee

$$\varphi_2\colon X_F^L \leq x_s(t) \leq X_F^U \wedge Y_F^L \leq y_s(t) \leq Y_F^U$$

- Robustness

$$\rho(\varphi_2, t, s) = \min\left(x_s(t) - X_F^L, X_F^U - x_s(t), y_s(t) - Y_F^L, Y_F^U - y_s(t)\right)$$

Invariants to avoid “corners”



B. Gafford, T. Dürschmid, G. A. Moreno, E. Kang (2020). Synthesis-Based Resolution of Feature Interactions in Cyber-Physical Systems.
IEEE/ACM International Conference on Automated Software Engineering (ASE) 2020.

$$s(t) = (x_s, y_s, x_o, s_o, x_s', y_x')$$

$$\varphi_1: \sqrt{(x_s(t) - x_o(t))^2 + (y_s(t) - y_o(t))^2} - d \geq 0$$

$$\varphi_2: \min\left(x_s(t) - X_F^L, X_F^U - x_s(t), y_s(t) - Y_F^L, Y_F^U - y_s(t)\right) \geq 0$$

$$\varphi_3: \sqrt{(x_s'(t) - x_o(t))^2 + (y_s'(t) - y_o(t))^2} - \sqrt{(x_s(t) - x_o(t))^2 + (y_s(t) - y_o(t))^2} \geq 0$$

$$\varphi_4: D - d \geq 0, D = \sqrt{(x_s - x_o)^2 + (y_x - y_o)^2} \cos\alpha, \alpha = \beta - \gamma - 90^o, \beta = \text{atan}\left(\frac{y_s - y_o}{x_s - x_o}\right), \gamma = \text{atan}\left(\frac{y_s' - y_s}{x_s' - x_s}\right)$$

Concluding Remarks

Carnegie Mellon University Software Engineering Institute

Verification for Rapid Fielding: Minimize Verification+ Verified Enforcement

Verify Cyber-Physical Properties

- Correct Value Right Time Correct Physical Behavior
- Resolve Conflicts Between Enforcers

Enforcement Protection (Open Source Mixed-Trusted Runtime Environment)

- Enforced **unverified** code + Prevent **verified** enforcer corruption

Resilient

- Resilience to environment changes, sensor failures, networked environments

Transition

- ONR Yolo: to fielded Navy system in progress!
- AFRL TEAMS: Enforcement, Adaptation in Manned and Unmanned Teams (MUMT)

Community

- 17 Academic/Industrial Publications: RTSS, RTCSA, USENIX, CDC,ACC, ECC, IFAC, ASE, AUVSI
- Open-source : Real-time mixed-trust runtime, Uberspark verified hypervisor

Team

Carnegie Mellon University Software Engineering Institute

SEI

Dr. Gabriel Moreno

Dr. Amit Vasudevan

Dr. Bjorn Andersson

Prof. Bruce Krogh

Mark Klein

Anton Hristozov

Michael McCall

Dr. Aaron Greenhouse

Dr. Dionisio de Niz

Collaborators

Prof. John Lehoczky (CMU/Stats)

Prof. Hyoseung Kim (UC Riverside)

Dr. Raffaele Romagnoli (CMU/ECE)

Prof. Bruno Sinopoli (WUSL)

Paul Griffioen (CMU/ECE)

Dr. Ruben Martins (CMU/CSD)