

RHIMES PI Meeting

YOLO Transition to Navy Systems

March 6-7, 2018

Presenter: Dionisio de Niz

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213



Copyright 2019 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.

References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute.

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.

DM19-0252

Team Members

Carnegie Mellon University

SEI

- Dr. Dionisio de Niz
- Dr. Bjorn Andersson
- Dr. Bruce Krogh (Prof. Emeritus)
- Dr. Gabriel Moreno
- Mr. Anton Hristozov
- Dr. Amit Vasudevan

ECE

- Raffaele Romagnoli
- Prof. Bruno Sinopoli

Columbia University

- Prof. Simha Sethumadhavan
- Miguel Arroyo
- Mohamed Tarek

Cyber-Physical Systems (CPS)

Composed of

- Software Controlling (cyber)
- Physical Phenomena (physical)

Examples:

- Car Airbag
 - Software detects crash and triggers airbag inflation
- Drone
 - Software detects and corrects drone attitude
- Inverted Pendulum
 - Software detects and corrects pendulum inclination

Inertia

- Physical phenomena evolves even w/o software interaction
- Stable control requires continuous cyber-physical interaction at correct timing

Cyber-Physical Systems Security

CPS

- Denial of Service can lead to damage
- Security mechanisms preserve physical process properties (e.g., control stability)
- If software stops physical process continues
- Software state recoverable from sensors

Good News:

- Physical process continues working during software blackouts

Bad News:

- Long software blackout leads to physical damage

IT

- Denial of Service: only temporary interruption
- Security mechanisms focused on information protection
- If software stops service stops
- Software state must be backed up

Cyber-Attacks Resilience in CPS: Software Rejuvenation (YOLO)

YOLO Model: Reset Constantly, and Change Program

Reset processor to safe state (checkpoint)

- Current state is lost

Recover state from:

- Physical process
- Protected storage

BUT

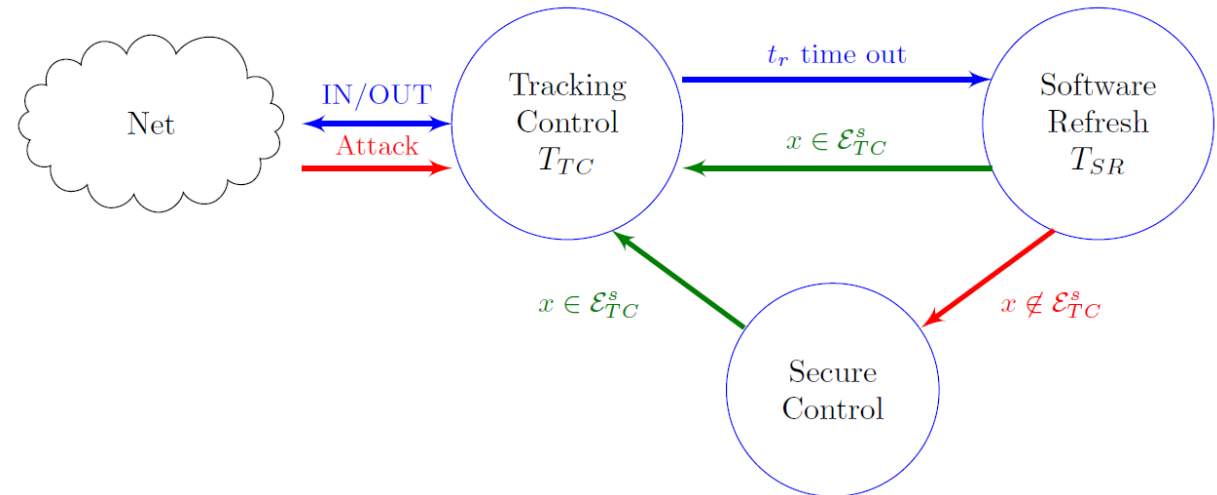
- Ensure: cure not worst than illness

Verify:

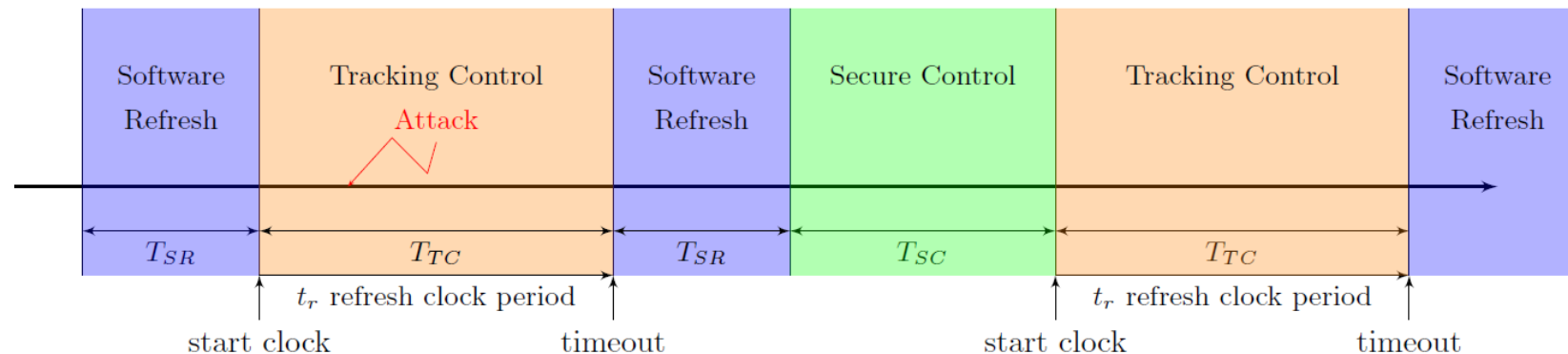
- Reboot not too frequent
- Controller blackout not too long
- State is recoverable or protected (including checkpoint)

Software Rejuvenation Operating Modes

1. Tracking Control (TC)
2. Software Refresh (SR)
3. Secure Control (SC)



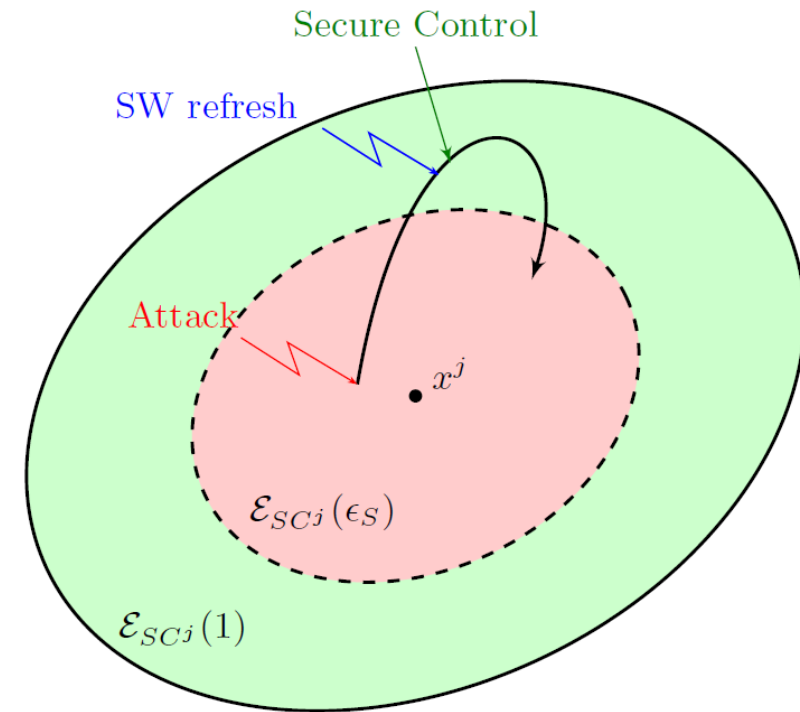
- The switch from TC to SR is triggered by a timer (unsecure information)
- From SR to TC or SC there is a condition to be satisfied (secure information)



Software Rejuvenation Secure Control

Defining

- Safety Set $\mathcal{E}_{SC^j}(1)$ (green)
- $\mathcal{E}_{SC^j}(\epsilon_s) \triangleq \epsilon_s \mathcal{E}_{SC^j}(1)$ (light red)
- $T_{UC} = T_{TC} + T_{SR}$

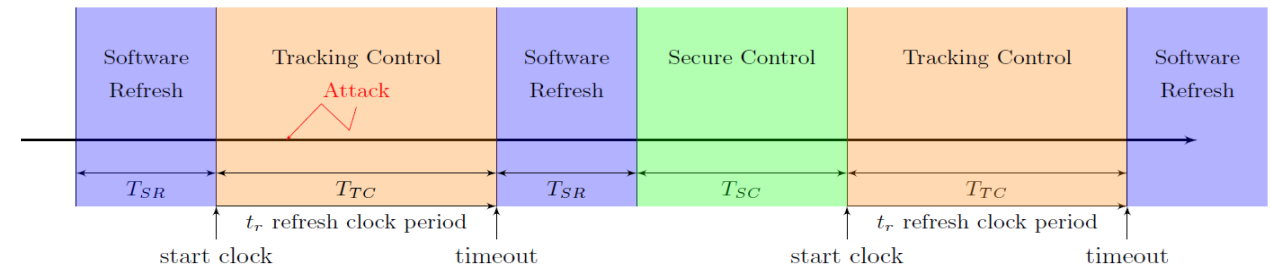


Goal

- Find ϵ_s and T_{UC} such that

$$\mathcal{R}(T_{UC}; \mathcal{E}_{SC^j}(\epsilon_s), U) \subseteq \mathcal{E}_{SC^j}(1)$$

$$T_{UC} > T_{SR}$$



Reachability set computed from the inner set under the action of uncertain control input for the time interval T_{UC}

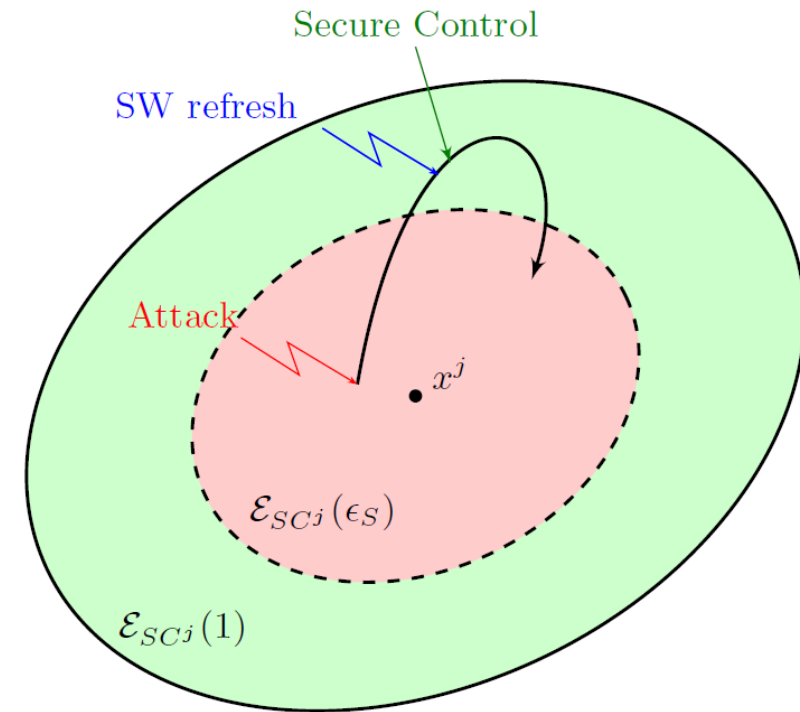
Software Rejuvenation Secure Control

Given ϵ_s and T_{UC} such that

$$\mathcal{R}(T_{UC}; \mathcal{E}_{SC^j}(\epsilon_s), U) \subseteq \mathcal{E}_{SC^j}(1)$$

How can we guarantee stability and recoverability?

$\mathcal{E}_{SC^j}(1)$ has to be a Positively Invariant Set



Software Rejuvenation Secure Control

Controlled System: $\dot{x} = f_\phi(x) \triangleq f(x, \phi(x))$

Lyapunov Function: $V_\phi : \mathbb{R}^n \rightarrow \mathbb{R}$, $\mathcal{N}_{V_\phi}(x_{eq}) \subseteq \mathcal{N}_\phi(x_{eq})$,
 $V_\phi(x_{eq}) = 0$ and $\forall x \in \mathcal{N}_{V_\phi}(x_{eq}) - \{x_{eq}\} : (i) V_\phi(x) > 0, (ii)$

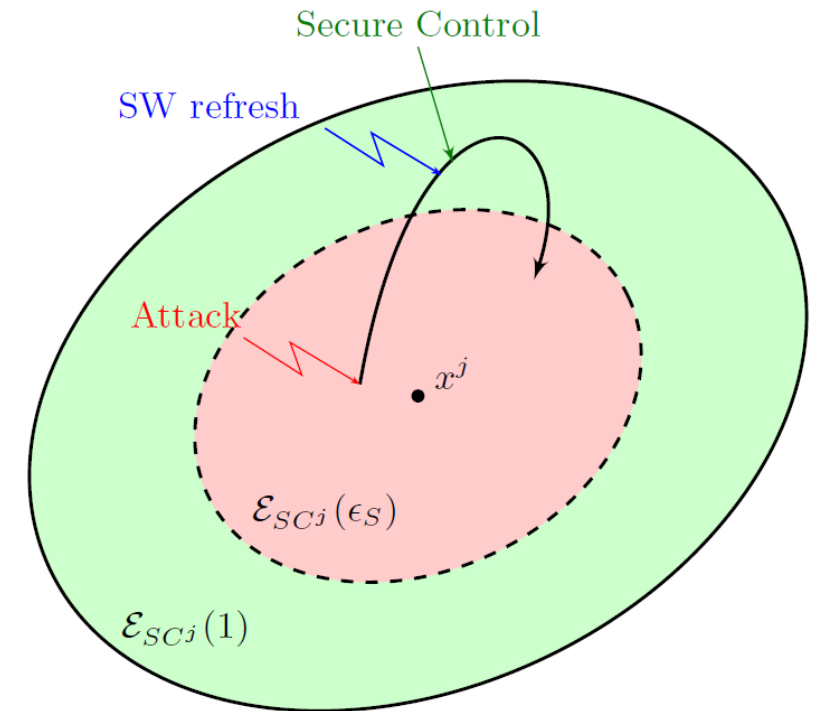
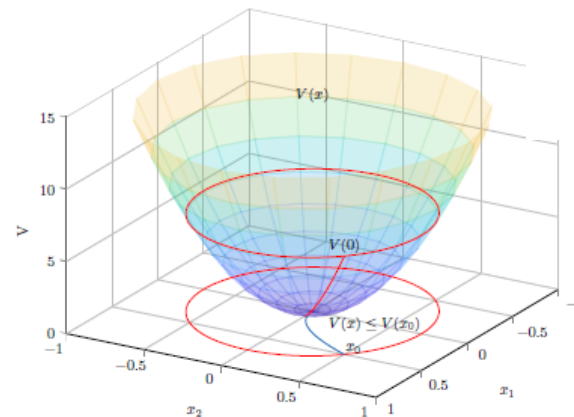
$$\dot{V}_\phi(x) = \frac{\partial V}{\partial x} \cdot f_\phi(x) < 0$$

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

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

Positively Invariant Set. For any $0 < \epsilon \leq 1$, $\mathcal{E}_\phi(\epsilon)$ is an *invariant set*.

$$\forall t > 0, \mathcal{R}(t; \mathcal{E}_\phi(\epsilon), \phi) \subseteq \mathcal{E}_\phi(\epsilon)$$

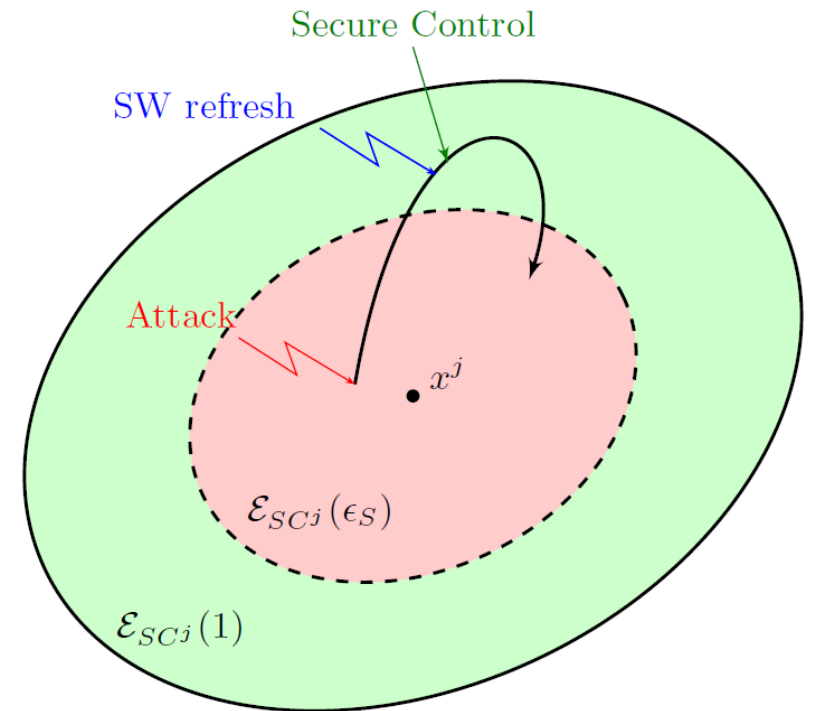


Software Rejuvenation Secure Control

Prop.1. Given $\dot{x} = f_\phi(x) \triangleq f(x, \phi(x))$ with stabilizing controller φ for equilibrium state $(x_{eq}, \varphi(x_{eq}))$ and Lyapunov function $V_\varphi(x)$ as defined above, given $\epsilon > 0$ for any $\epsilon < \epsilon' \leq 1 \exists \gamma > 0 \ni \forall t \geq (\epsilon' - \epsilon)\gamma^{-1}$,

$$\mathcal{R}(t; \mathcal{E}_\varphi(\epsilon'), \varphi) \subseteq \mathcal{E}_\varphi(\epsilon).$$

Prop.2. For any $U \subseteq \mathcal{U}$ and any $0 < \epsilon < \epsilon' \leq 1$, $\exists T_U > 0 \ni \mathcal{R}(t; \mathcal{E}_\varphi(\epsilon), U) \subseteq \mathcal{E}_\varphi(\epsilon') \forall t < T_U$.



- We can always recover in a finite time
- Given a reduced version of the Safety Set we can always find a period of time where is allowed uncertain control.

Software Rejuvenation

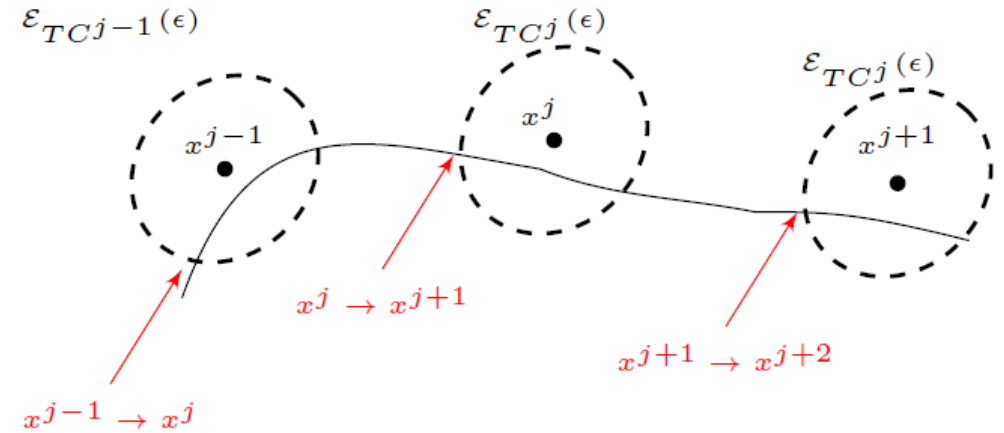
Analysis of mission progress

Idea:

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

Goal:

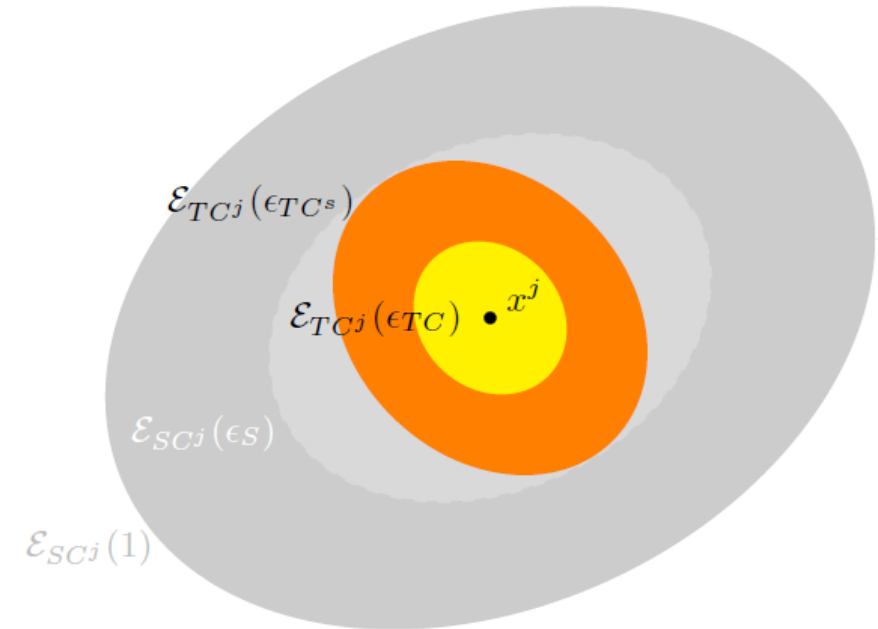
- Safety transition from one way point to the next one.
- Liveness (in the case of no attack)



Software Rejuvenation

Analysis of mission progress

- The new sets generalize the fact that a different controller may be used during TC.
- Those sets are contained in the Safety Set

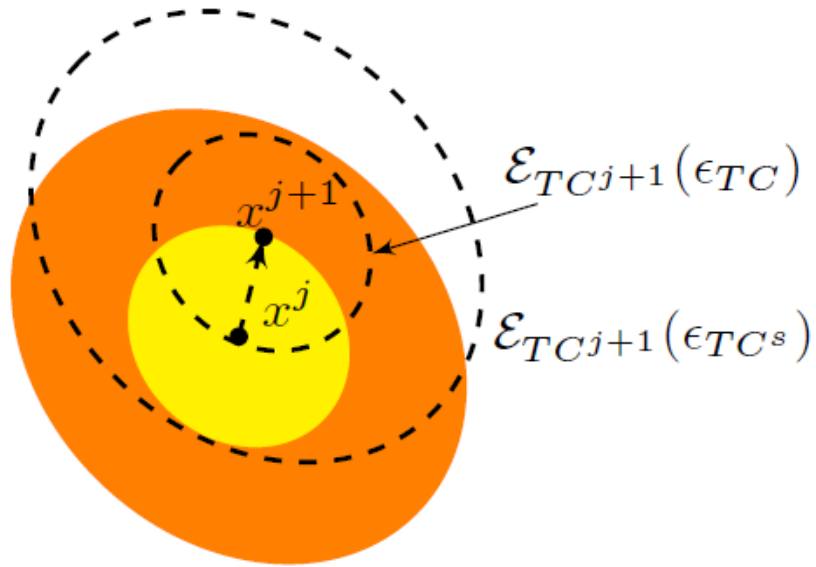


$$\mathcal{E}_{TCj}(\epsilon_{TC}^s) \subseteq \mathcal{E}_{SCj}(\epsilon_s)$$
$$\mathcal{E}_{TCj}(\epsilon_{TC}) \subseteq \mathcal{E}_{TCj}(\epsilon_{TC}^s)$$

Software Rejuvenation

Analysis of mission progress

Safety Transition



Theorem. Let \bar{t} be the time instant associated to the transition between two consecutive equilibrium points x^j and x^{j+1} , then $x(\bar{t}; x_0, U) \in \mathcal{E}_{TC^{j+1}}(\epsilon_{TC}^s)$ with $x_0 \in \mathcal{E}_{SC}(1)$ if and only if

$$\mathcal{E}_{TC^j}(\epsilon_{TC}) \subseteq \mathcal{E}_{TC^{j+1}}(\epsilon_{TC}^s)$$

Software Rejuvenation

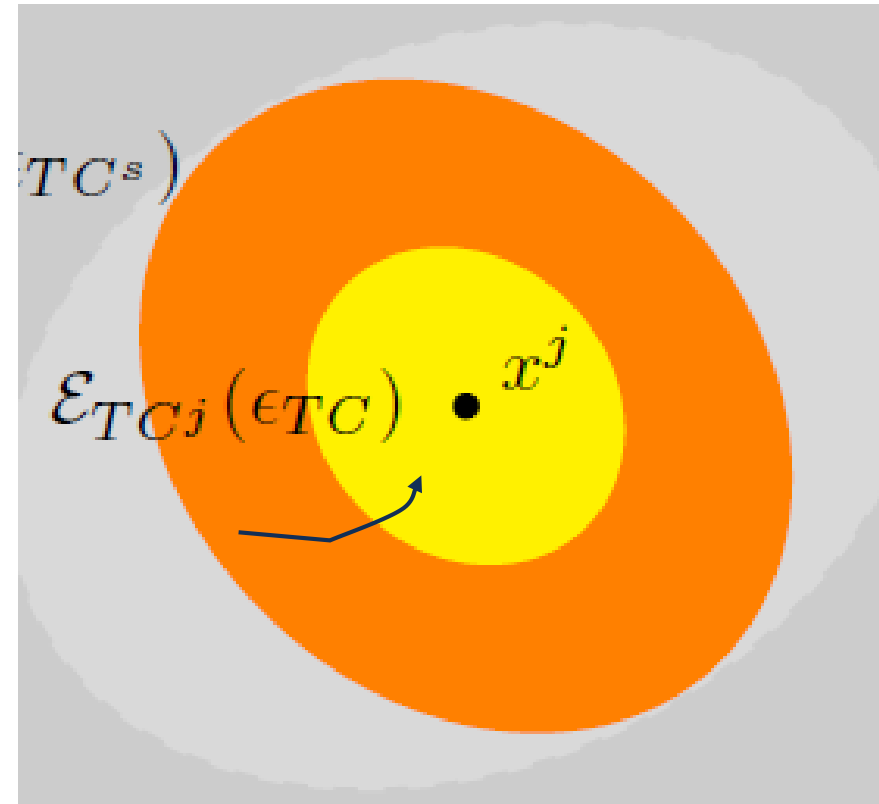
Analysis of mission progress

Liveness: in presence of software refresh the state of the system has to converge in to $\mathcal{E}_{TC^j}(\epsilon_{TC})$

Assumptions

- No attack
- During SR the control inputs are constant and equal to the last value provided by the controller before the reboot.

PS: If the attack is persistent the liveness cannot be guaranteed in this case. But the system is always safe.



Software Rejuvenation

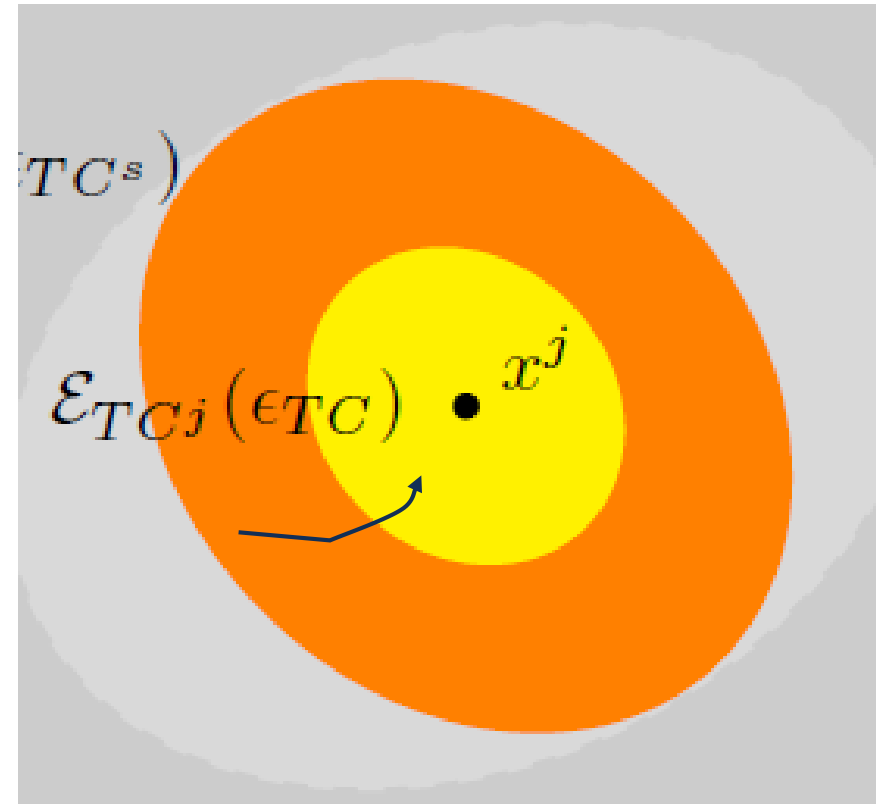
Analysis of mission progress

Liveness: in presence of software refresh the state of the system has to converge in to $\mathcal{E}_{TCj}(\epsilon_{TC})$

Assumptions

- No attack
- During SR the control inputs are constant and equal to the last value provided by the controller before the reboot.

PS: If the attack is persistent the liveness cannot be guaranteed in this case. But the system is always safe.



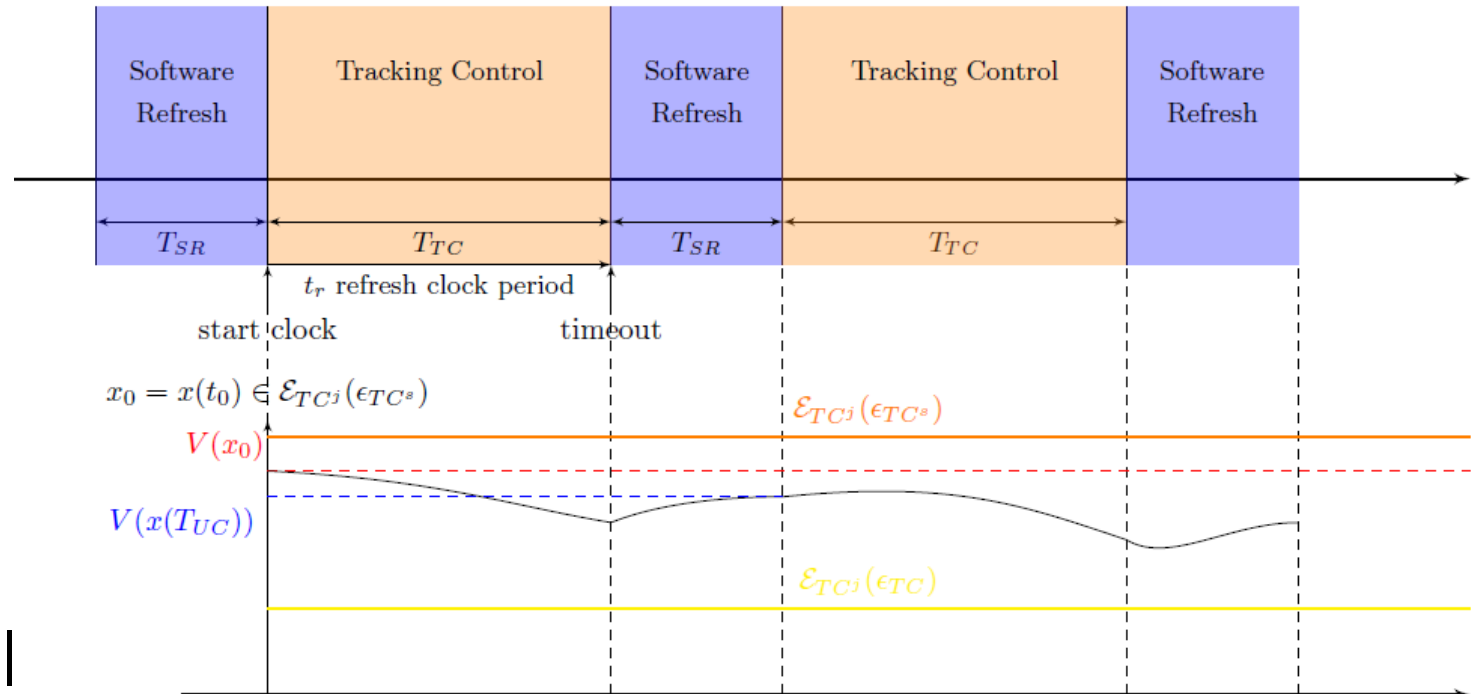
Software Rejuvenation

Analysis of mission progress

- From $V(x_0)$ the Lyapunov function decreases up to $V(x(T_{TC}))$
- At the end of SR, the function V may increase



$$V(x_0) - V(x(T_{TC})) > |V(x(T_{TC})) - V(x(T_{TC} + T_{SR}))|$$



Theorem 2. For all $x_0 \in [\mathcal{E}_{TCj}(\epsilon_{TC}^s) - \mathcal{E}_{TCj}(\epsilon_{TC})]$,
 $\exists \bar{T} > 0 \ni \forall t > \bar{T} \ x(t; x_0, u_{[0,t]}) \in \mathcal{E}_{TCj}(\epsilon_{TC})$, if

$$V_{TCj}(x_0) - V_{TCj}(x(T_{TC}; x_0, TC)) >$$

$$|V_{TCj}(x(T_{UC}; x_0, \bar{u}_{[0,T_{SR}]}) - V_{TCj}(x(T_{TC}; x_0, TC^j))|.$$

Software Rejuvenation

Analysis of mission progress

6 DOF \Rightarrow 12 state variables

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

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

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

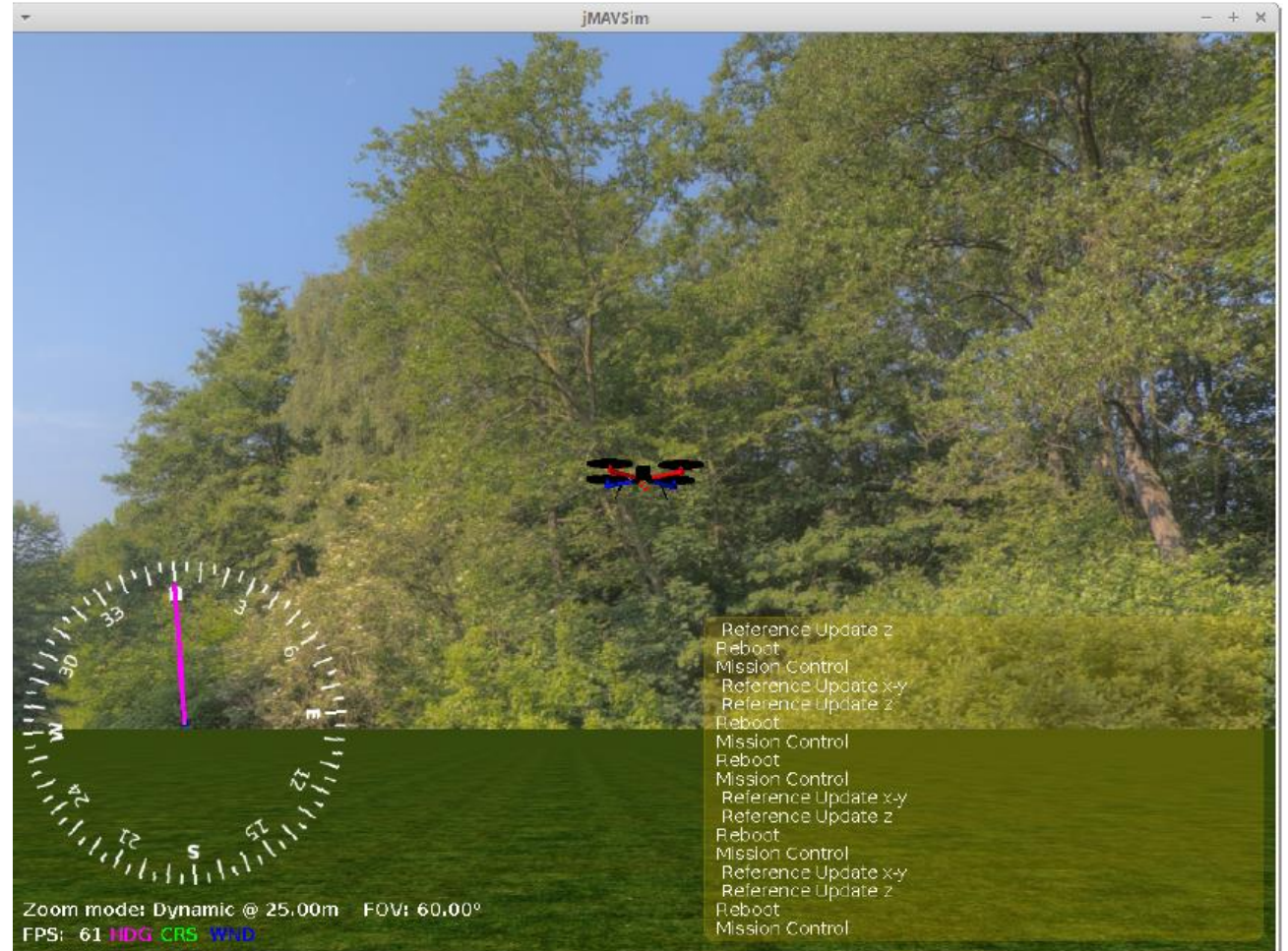
$$\ddot{\phi} = \frac{1}{J_x} \tau_\phi$$

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

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

Linear design:

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



Software Rejuvenation: Drone experiment



Software Rejuvenation

Analysis of mission progress

6 DOF \Rightarrow 12 state variables

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

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

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

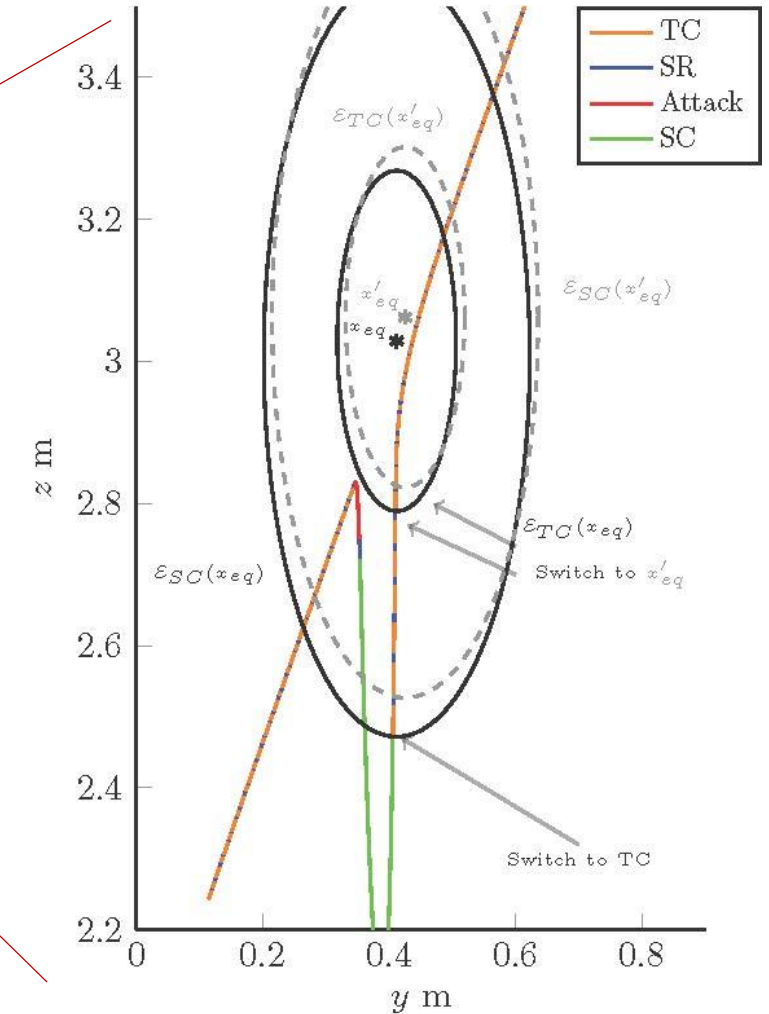
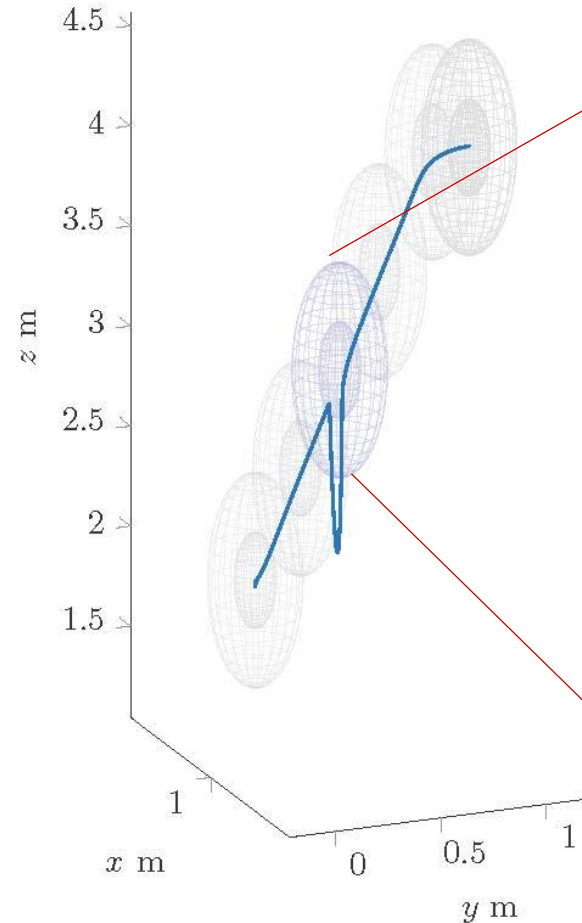
$$\ddot{\phi} = \frac{1}{J_x} \tau_\phi$$

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

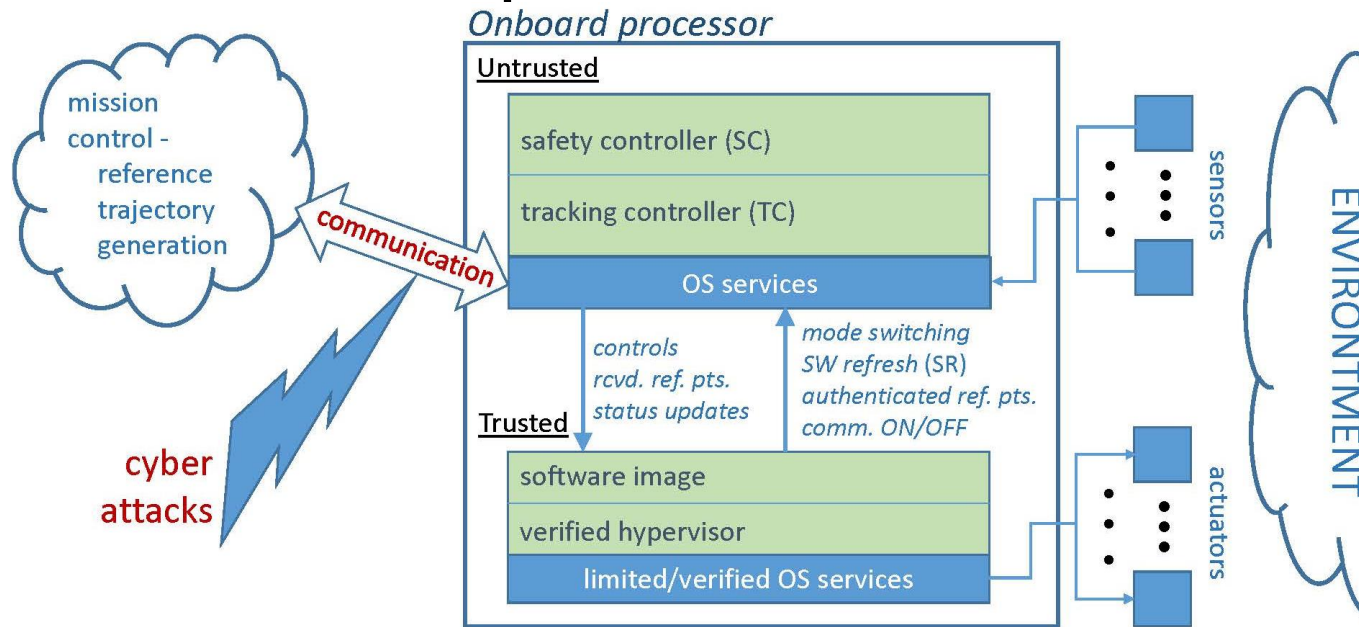
$$\ddot{\psi} = \frac{1}{J_z} \tau_\psi$$

Linear design:

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

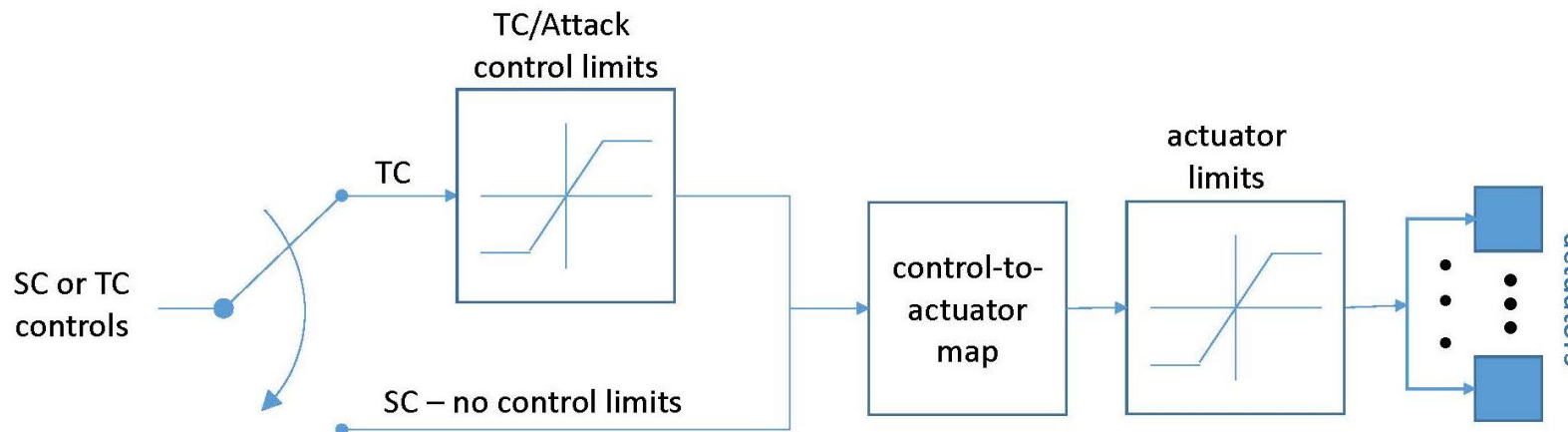


Software Rejuvenation Protection Requirements



Requirements

- Authenticate reference points
- Protect checkpoint integrity
- Protect refresh mechanism
- Set actuator limits



Software Rejuvenation Protections (1)

Space (Memory) Protection

- Cheap implementation (low/no overhead)

Drone Experiment

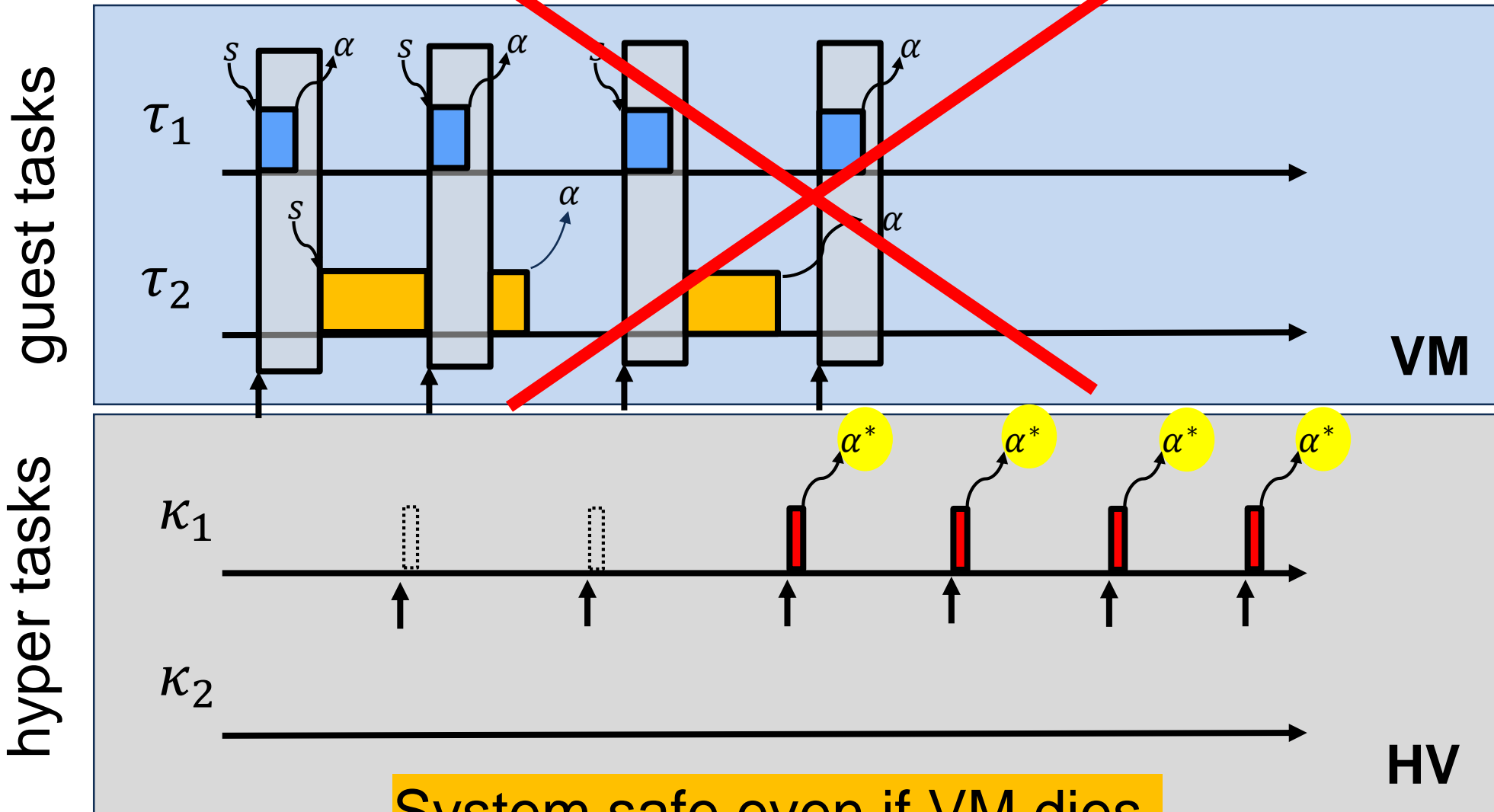
- Take advantage of Micro-Hypervisor :
 - Only memory protection: (nested page tables – hardware translation)
 - Data
 - Code
 - No other virtualization
 - Verified code implementing memory protection
 - Separate serial port for communication

Time Protection

- Separate timer for hypervisor (no virtualization)
- Execute protected code in hypervisor (hyperapp)

Software Rejuvenation Protection (2)

Two Execution Environments: Two schedulers: VM+HV



System safe even if VM dies

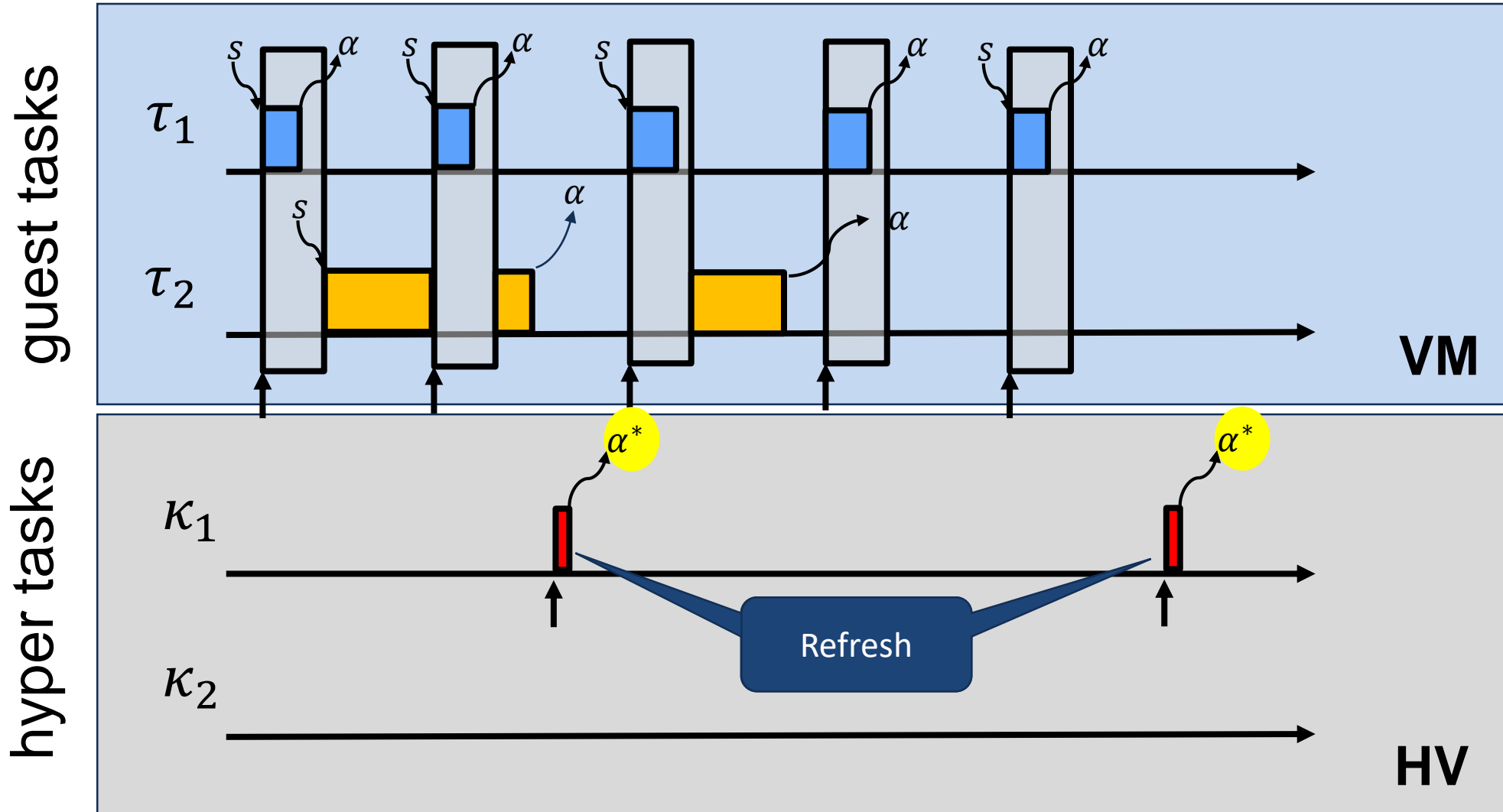
Current Experiments (1)

Drone Protection (VM Crash)

The screenshot displays a desktop environment with two main windows. On the left is the JMAVSim window, which shows a 3D simulation of a drone in a green field. A compass is visible at the bottom of the window, and status text at the bottom reads: "Zoom mode: Dynamic @ 25.00m FOV: 60.00° FPS: 60 HDG CRS WIND Init MAVLink". On the right is the QGroundControl v3.4.4 window, which shows a top-down aerial view of the drone's location in a park area. The view includes labels for "Irchelpark", "Milchbuckttunnel", and "Winterthurerstrasse". A red arrow labeled "H" points to a location in the park. A "WAITING FOR VIDEO" message is displayed in a black box. A data panel on the right shows flight statistics: "Altitude-rel (m) -0.0", "Ground Speed (m/s) 0.0", and "Flight Time 00:00:00". A context menu is open over the top-right corner of the QGroundControl window, with options: "Start recording", "Cancel recording", "Save recording", "Hide window", and "Quit".

Software Rejuvenation Protection (3)

Two Execution Environments: Two schedulers: VM+HV



Experiment to put it all together (ongoing)

Implement a YOLOized System

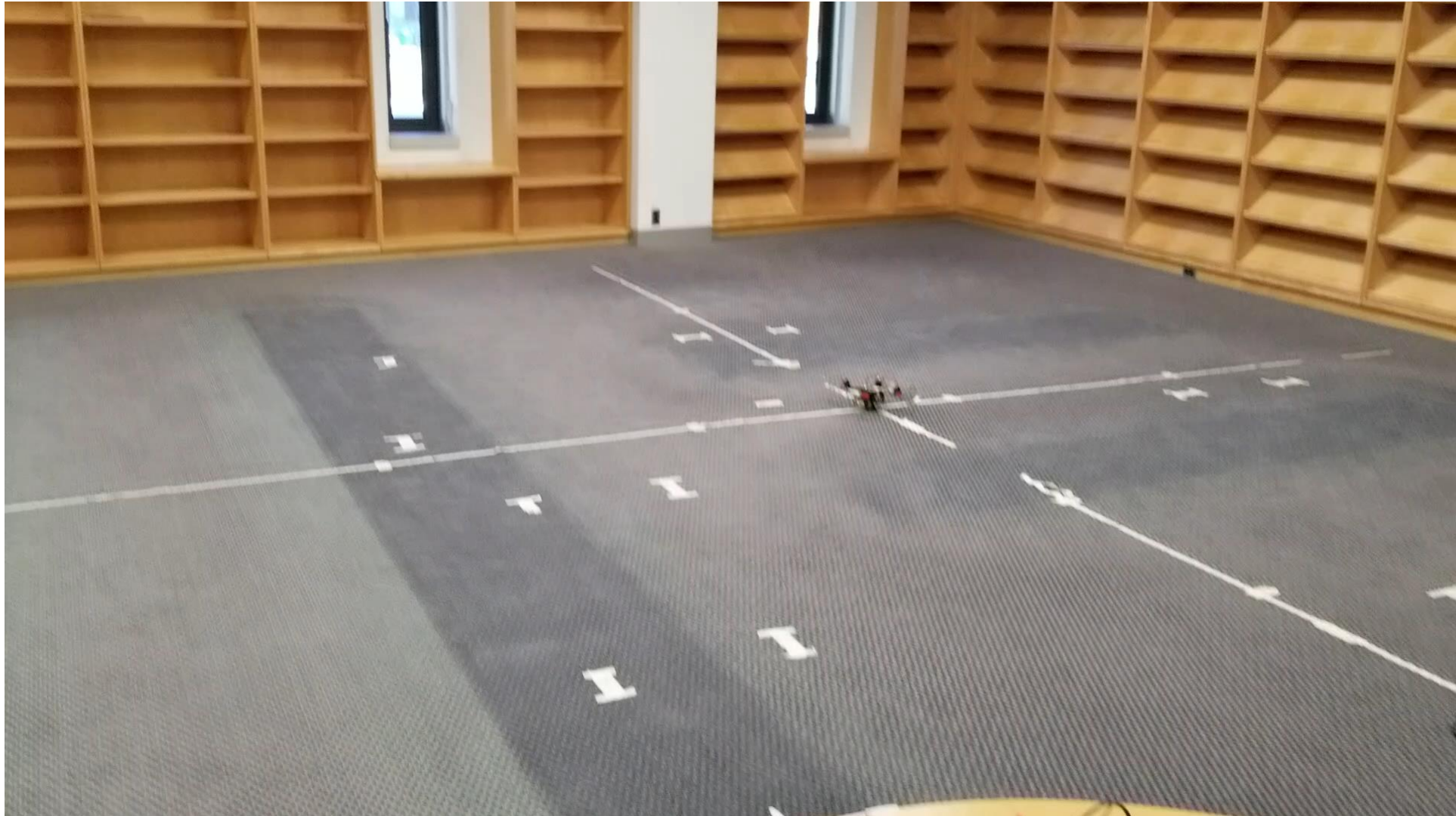
- Verified Refreshing Controller
- Verified Refreshing Trajectory Tracker
- Timing Verification
- Space and Timing Protection
- YVT Tool Verification Model (more in a moment)

Indoor Drone

- PX4 Controller Implementation
- Process-based refresh (micro-reboots)

Current Experiments (2)

Micro-reboot in indoor drone



Yoloized-system Verification Tool (YVT)

Integrate verification tools into architectural model

- AADL Model
- OSATE (AADL modeling tool)

Physics Verification

- Matlab Annex (sublanguage)
 - Model of Plant and Controller
- Java interface with Matlab
- Display safe ellipsoids
- Prove properties

Scheduling tool

- Generates SMT models (calling Z3)
- Generates counter-example timeline

AADL Model

Top Level Model

```
system implementation test1.i
  subcomponents
    p1: process controllers.i;
    quadrotor: system quadrotor.i;
    cpu: processor generic_processor;
  connections
    c1: port quadrotor.transfer->p1.transfer;
  connections
    Actual_Processor_Binding => (reference (cpu)) applies to p1;
end test1.i;
```

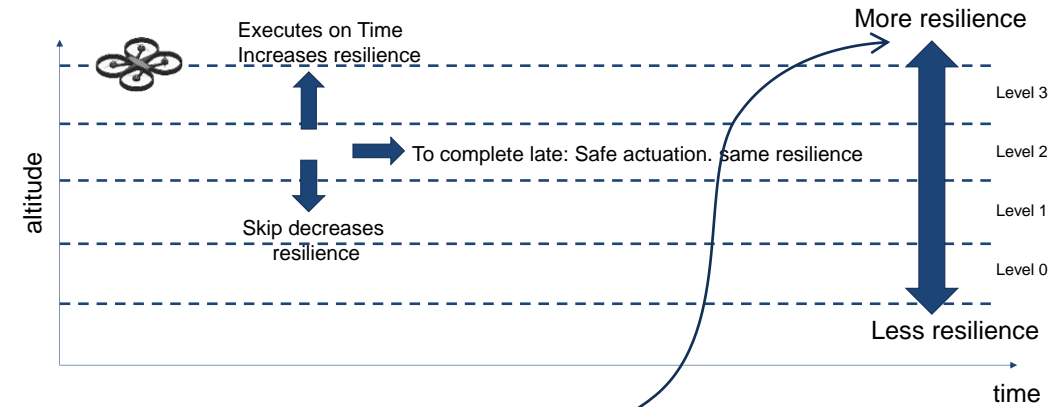
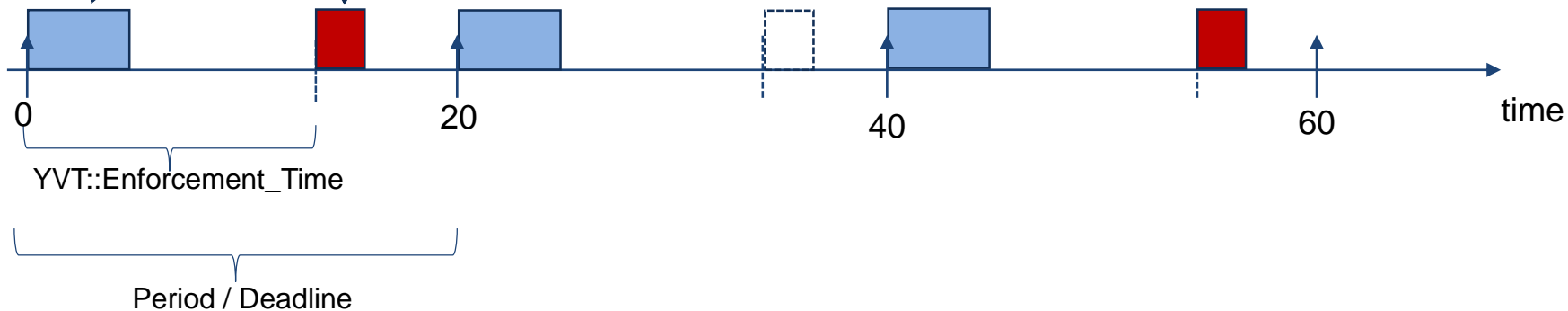
Controllers component

```
process implementation controllers.i
  subcomponents
    t1: thread controller_thread.quadrotor_controller2;
    t2: thread controller_thread.i;
  connections
    c1: port transfer->t1.transfer;
end controllers.i;
```

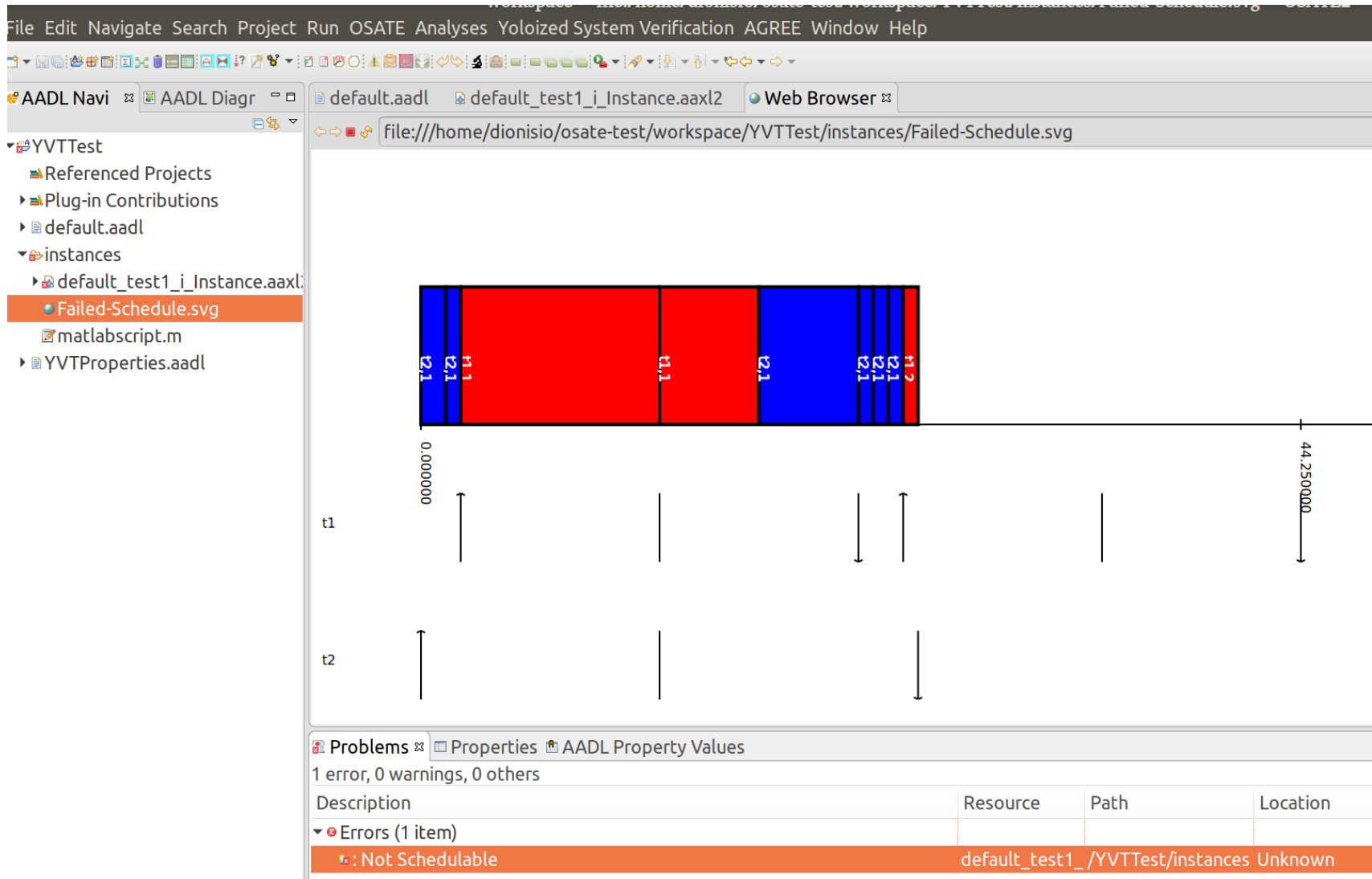
Scheduling Parameters

```

process implementation myproc.i
  subcomponents
    t1:thread controller_thread.quadrotor_controller2;
    t2:thread controller_thread.i;
  connections
    c1: port transfer->t1.transfer;
  properties
    -- Unschedulable parameters
    Priority => 2 applies to t1;
    Period => 20 ms applies to t1;
    Deadline => 20 ms applies to t1;
    YVTProperties::Enforcement_Time => 10 ms applies to t1;
    Compute_Execution_Time => 8 ms .. 8 ms applies to t1;
    YVTProperties::Enforcement_Execution_Time => 8 ms applies to t1;
    YVTProperties::Max_Resilience_Level => 2 applies to t1;
    YVTProperties::Max_Jobs_Per_Valid_Exec_Time => 2 applies to t1;
    ...
  
```



Schedulability feedback (counter-example timeline)



Physics Verification (1)

Plant Model

```
system implementation quadrotor.i
    annex matlab {**
        %% Quadrotor Example - Invariant Ellipsoids - version 1.1
        % Author: Raffaele Romagnoli   Date: 09/04/18
        %% Note:
        %%   xx_v: the equilibrium point due to the gravity force is considered

        %% Quadcopter: Linearized Dynamics + LQR controller

        %% Generic Quadrotor X PX4 Parameters
        % no integrator
        m = 0.8;           % mass (Kg)
        L = 0.33/2;       % arm length (m)
        Jx = 0.005;       % inertia seen at the rotation axis. (Kg.m^2)
        Jy = 0.005;       % inertia seen at the rotation axis. (Kg.m^2)
        Jz = 0.009;       % inertia seen at the rotation axis. (Kg.m^2)
        g = 9.81;         % acceleration due to gravity m/s^2
        ...
    **};
end quadrotor.i;
```

Physics Verification (2)

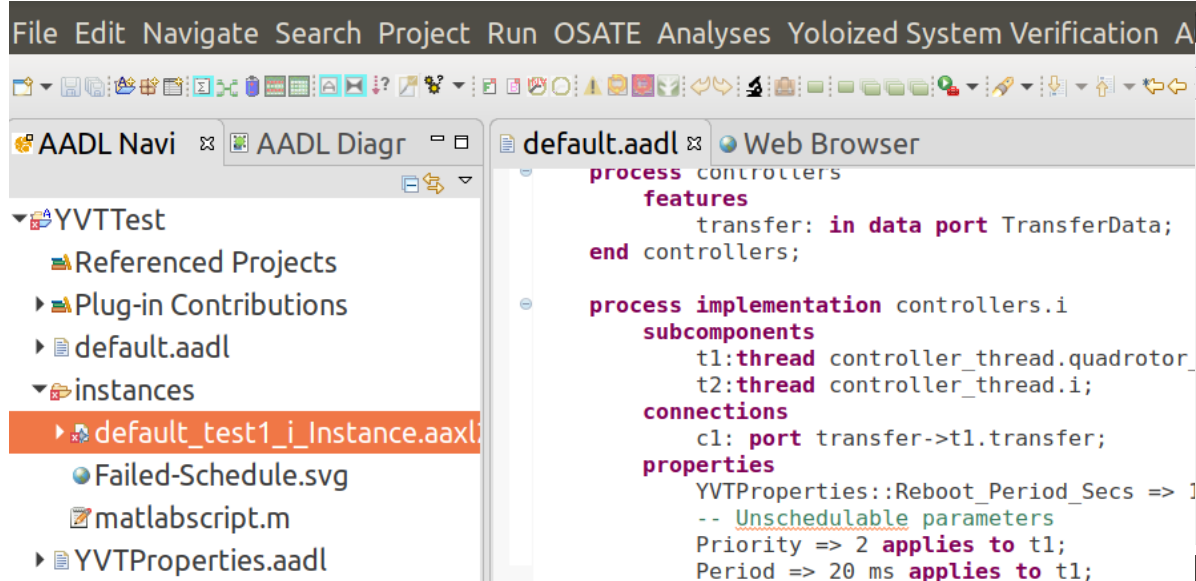
Controller Model

```
thread implementation controller_thread.quadrotor_controller2
  annex matlab {**
    %@output:positionok, orientationok, velocityok, angularok, livenessok
    %@input:reboot_period_secs
    %% LQR Controller
    Q=50*eye(n);
    R=100*eye(p);
    [K_hat,S,E] = lqr(hat_sys,Q,R);
    ...

  **};
end controller_thread.quadrotor_controller2;
process implementation controllers.i
  subcomponents
    t1:thread controller_thread.quadrotor_controller2;
    t2:thread controller_thread.i;
  connections
    c1: port transfer->t1.transfer;
  properties
    YVTProperties::Reboot_Period_Seconds => 180 ms applies to t1;
end controllers.i;
```

Physics Verification (3)

Proven properties / Safe ellipsoids



The screenshot shows the AADL Navi interface. On the left, a tree view shows the project structure under 'YVTest', including 'default_test1_i_Instance.aaxl'. The main editor displays AADL code for a controller process:

```

process controllers
  features
    transfer: in data port TransferData;
  end controllers;

  process implementation controllers.i
    subcomponents
      t1:thread controller_thread.quadrotor;
      t2:thread controller_thread.i;
    end subcomponents;
    connections
      c1: port transfer->t1.transfer;
    end connections;
    properties
      YVTProperties::Reboot_Period_Secs => 1;
      -- Unschedulable parameters
      Priority => 2 applies to t1;
      Period => 20 ms applies to t1;
    end properties;
  end controllers.i;
end process;
        
```

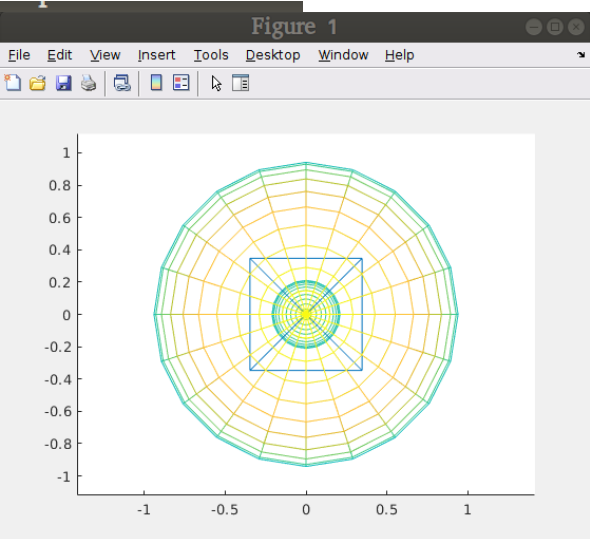


Figure 1: A 2D plot showing a grid of concentric ellipses centered at (0,0) within a square boundary. The x and y axes range from -1 to 1.

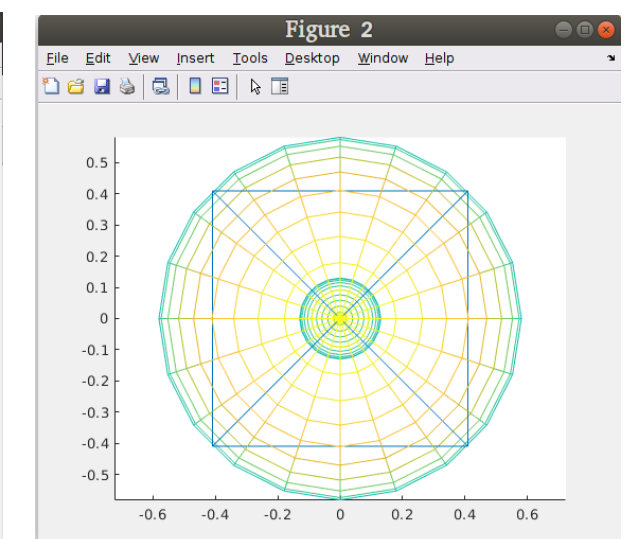


Figure 2: A 2D plot showing a grid of concentric ellipses centered at (0,0) within a square boundary. The x and y axes range from -0.6 to 0.6.

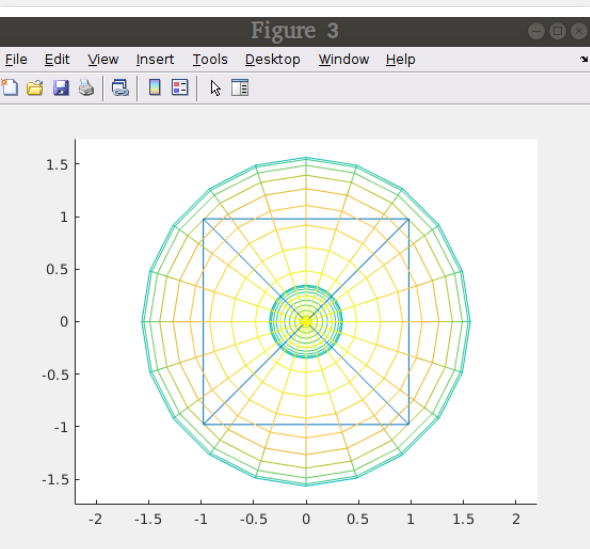


Figure 3: A 2D plot showing a grid of concentric ellipses centered at (0,0) within a square boundary. The x and y axes range from -1.5 to 1.5.

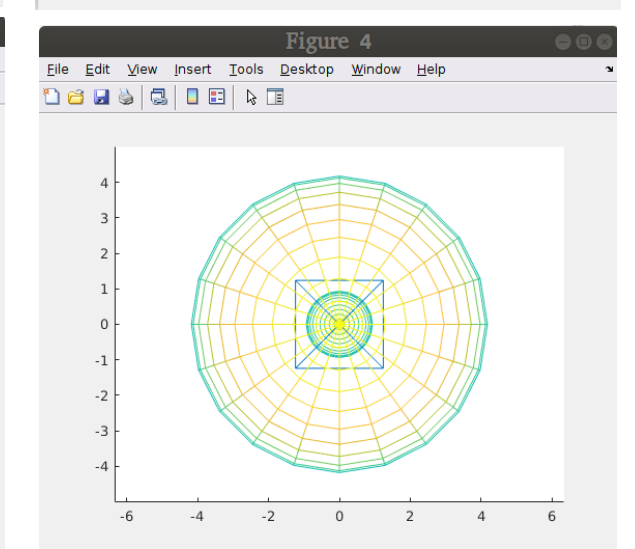
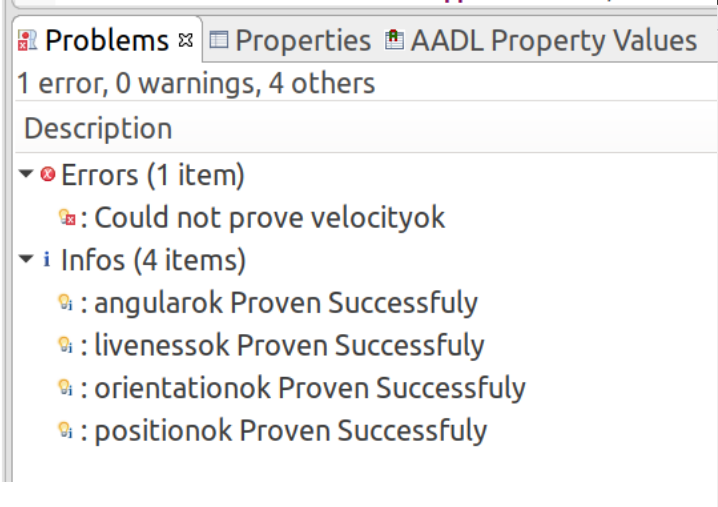


Figure 4: A 2D plot showing a grid of concentric ellipses centered at (0,0) within a square boundary. The x and y axes range from -6 to 6.



The screenshot shows the 'Problems' and 'Properties' panels. The 'Problems' panel indicates '1 error, 0 warnings, 4 others' and lists the following:

- Errors (1 item):
 - : Could not prove velocityok
- Infos (4 items):
 - : angularok Proven Successfully
 - : livenessok Proven Successfully
 - : orientationok Proven Successfully
 - : positionok Proven Successfully

Summary and Status (1)

So Far

Control Verification of Software Rejuvenation

- Verified refresh frequency
- Verified control blackout duration
- Verified trajectory tracking

Timing Verification

- Resilience Timing Verification
- Temporal Protection

Protection

- Efficient memory isolation (HV)
- VM+HV scheduling coordination
- Micro-reboot and persistent state implementation

Verification tool (YVT)

- Architectural model with timing and control models
- Integration of verification algorithms driven by AADL tool (OSATE)

Summary and status (2)

Looking ahead

System state recovery

- Smart Kalman Filter Initialization
- Multiple checkpoints

Scheduling

- Optimize control actuation update frequency for single setpoint
- Optimize control actuation update frequency for control trajectories

YVT Enhanced Verification Techniques

- YVT already contains some new techniques
more to come:
 - Timing verification of control trajectories

Extended Enforcement Techniques

- To explore: enabling trajectories with multiple checkpoints

Verified Transition Target Rejuvenation

- Implementation
- Verified YVT Model