RHIMES PI Meeting

YOLO Transition to Navy Systems

May 27, 2020

Presenter: Dionisio 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.

DM20-0428

Team Members

SEI

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

CMU / ECE

- Raffaele Romagnoli
- Paul Griffioen

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

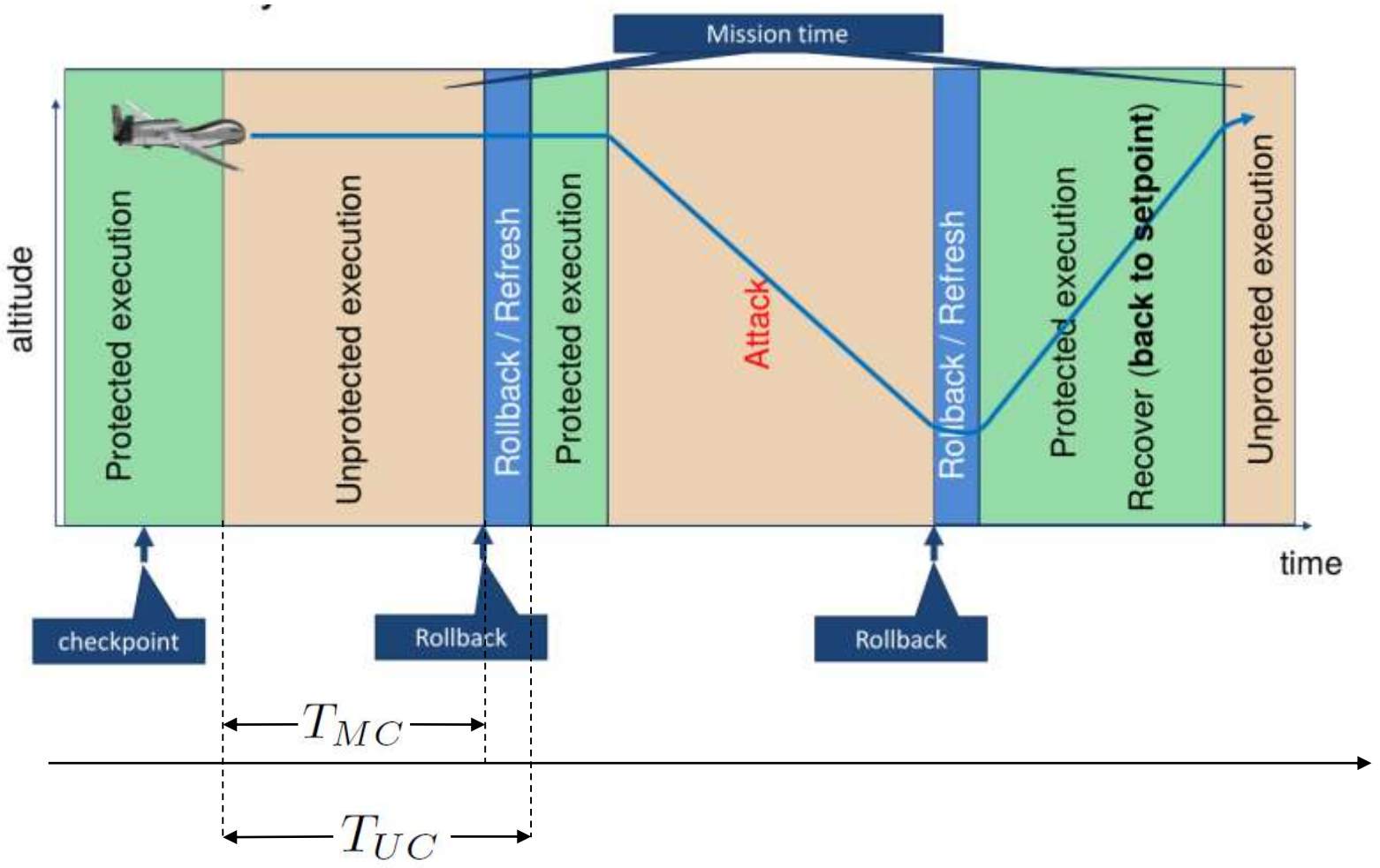
- Ensure: cure not worst than illness
  Verify:
  - Reboot not too frequent
  - Controller blackout not too long
  - State is recoverable or protected (including checkpoint)

Financials

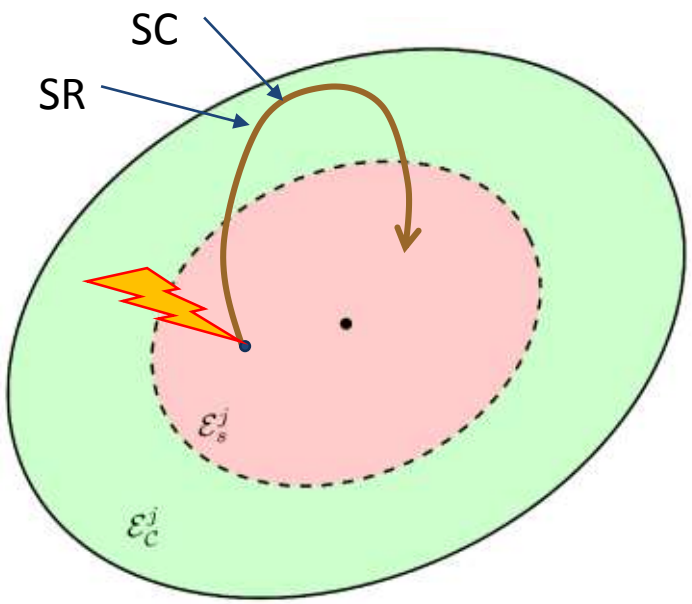
Software Rejuvenation

- Mission Control (MC)
  (Tracking Control - TC)
- Software Refresh (SR)
- Secure Control (SC)
- Unknown Control (UC)

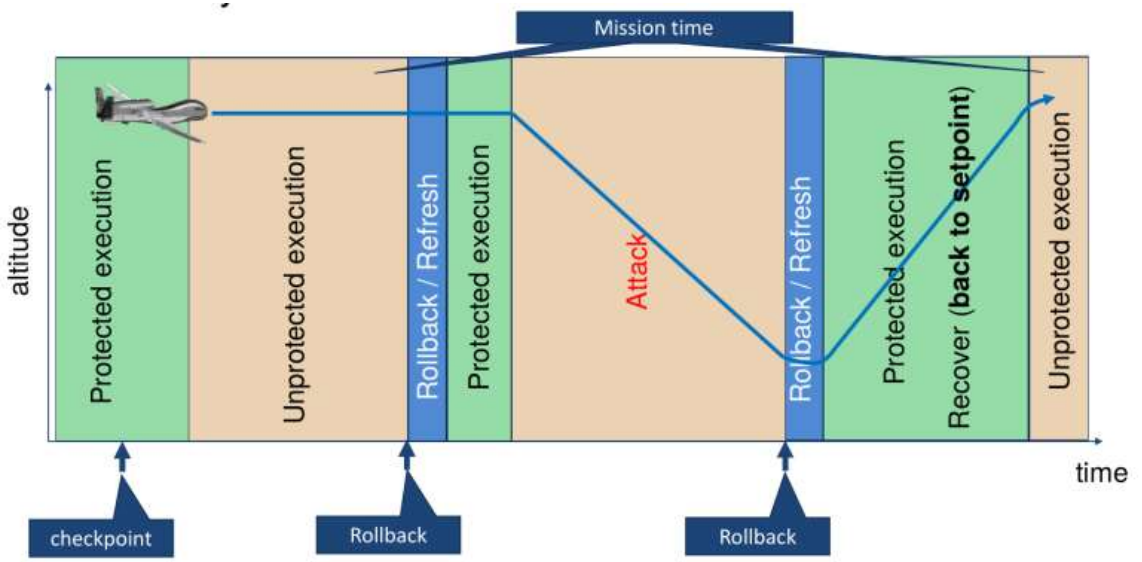
$$T_{MC} \triangleq T_{UC} - T_{SR}$$



Refresh Period Design



Safety Set



Lyapunov Theory

$$x \in \mathbb{R}^n \qquad \dot{x} = Ax + Bu$$

$$u = -Kx \quad \dot{x} = (A - BK)x = A_{cl}x$$

$$V(x) > 0, V(0) = 0, \dot{V}(x) < 0$$
$$P > 0 : V(x) = x^T P x,$$
$$\dot{V}(x) = A_{cl}^T P + P A_{cl} < 0$$

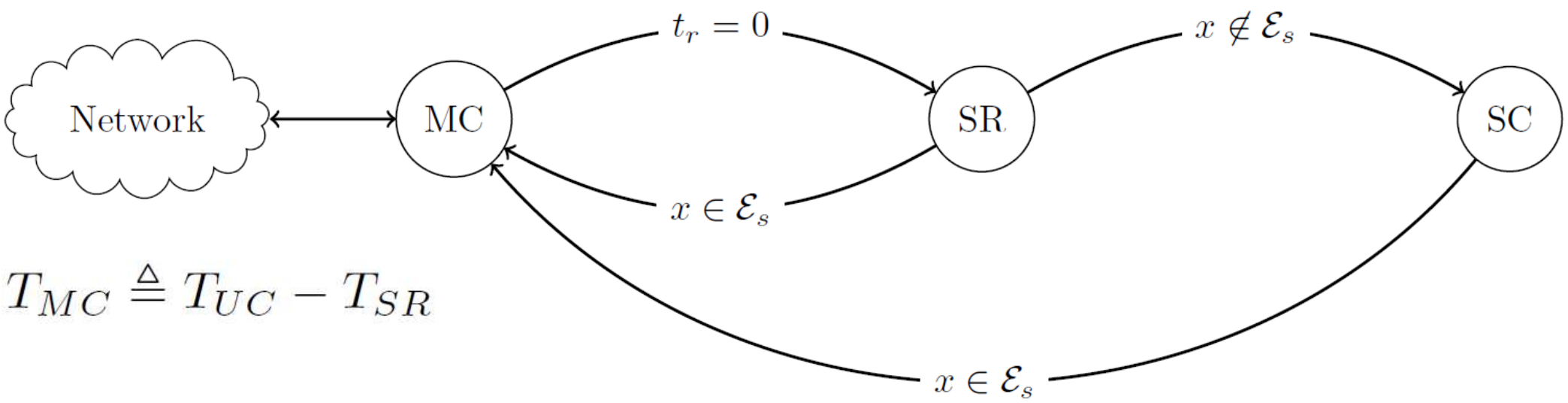
$$\mathcal{E}_c^j = \left\{x : x^T P x < 1\right\}$$
$$\mathcal{E}_s^j = \left\{x : x^T P x < s\right\} 0 < s < 1$$

Reach set analysis

$$\dot{x} = Ax + Bu \qquad \forall u \in U$$
$$\forall x \in \mathcal{E}_s^j$$

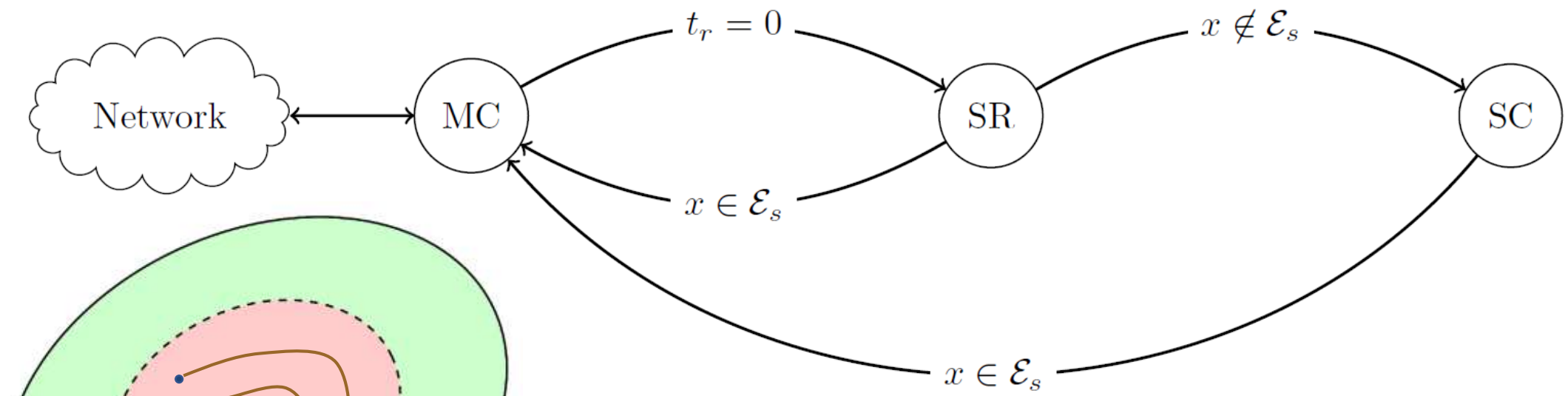
$$\mathcal{R}(t, \mathcal{E}_s^j, U) \in \mathcal{E}_{C^j}, \quad 0 \le t \le T_{UC}$$

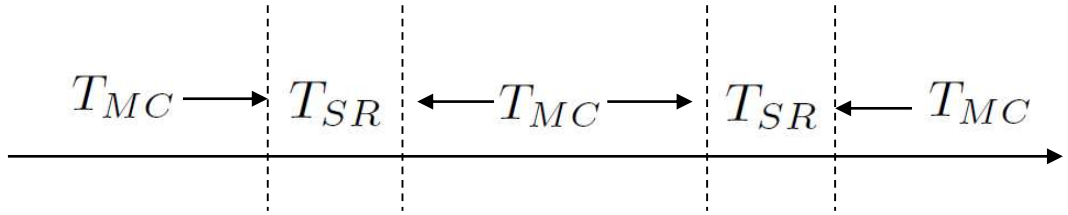
Software Rejuvenation Scheme



$$T_{MC} \triangleq T_{UC} - T_{SR}$$

Software Rejuvenation: Liveness

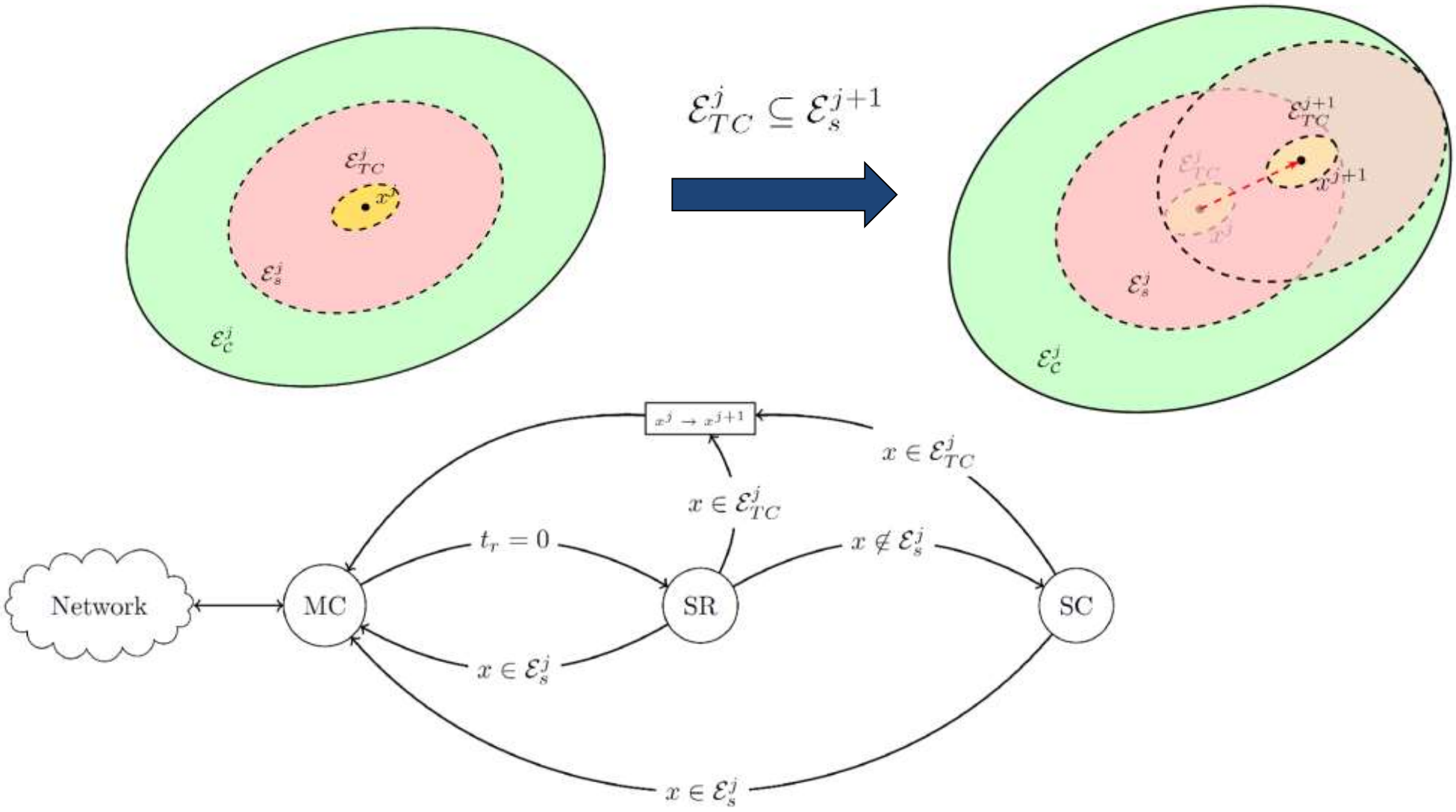




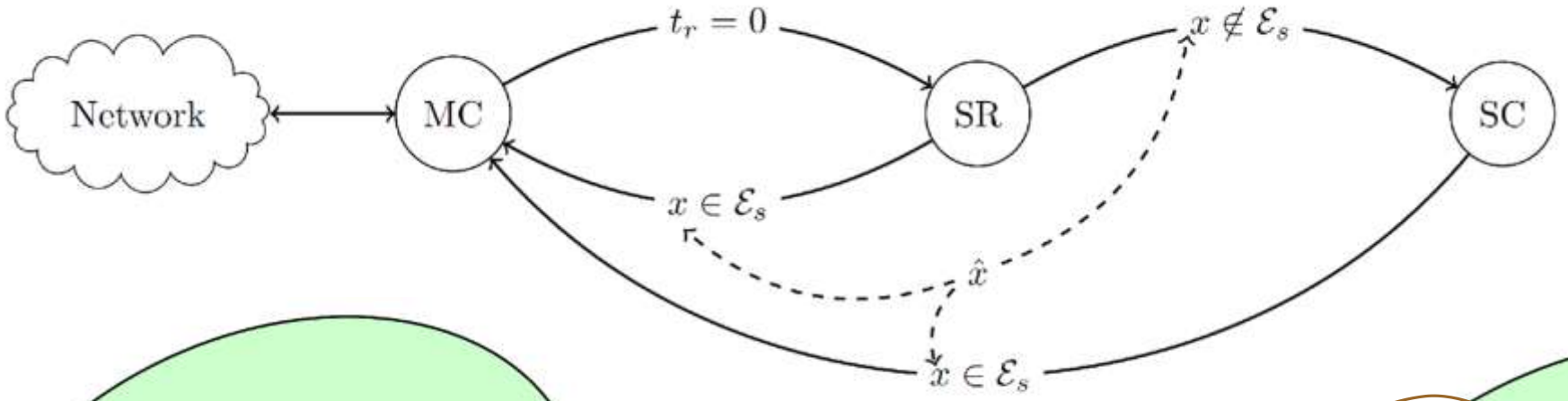
$V(t_0) \qquad V(t_1) \quad V(t_2)$

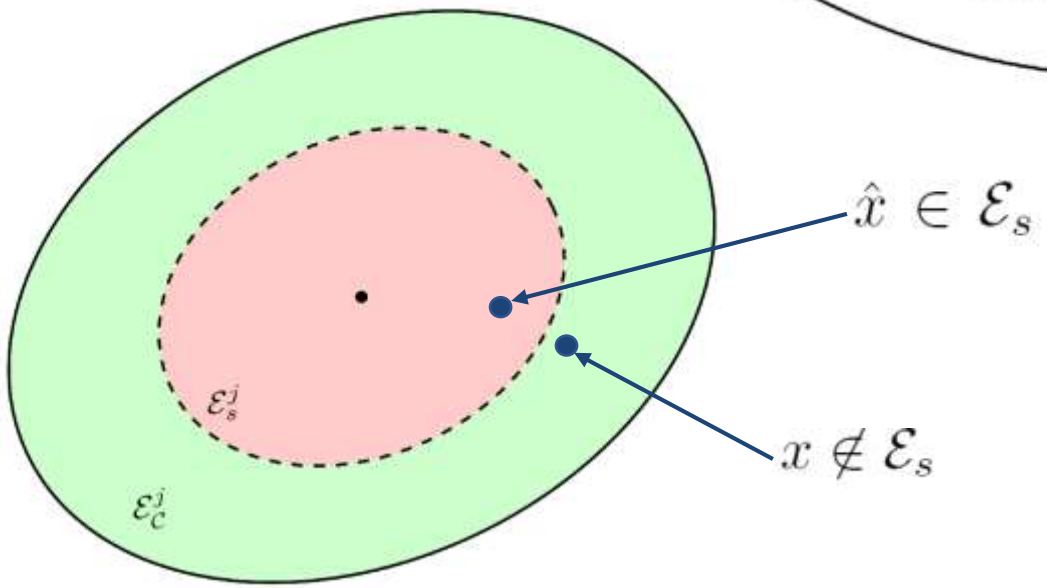
$V(t_0) - V(t_1) > 0 \qquad V(t_1) - V(t_2) < 0 \implies V(t_0) - V(t_1) > V(t_2) - V(t_1)$

Software Rejuvenation: Tracking

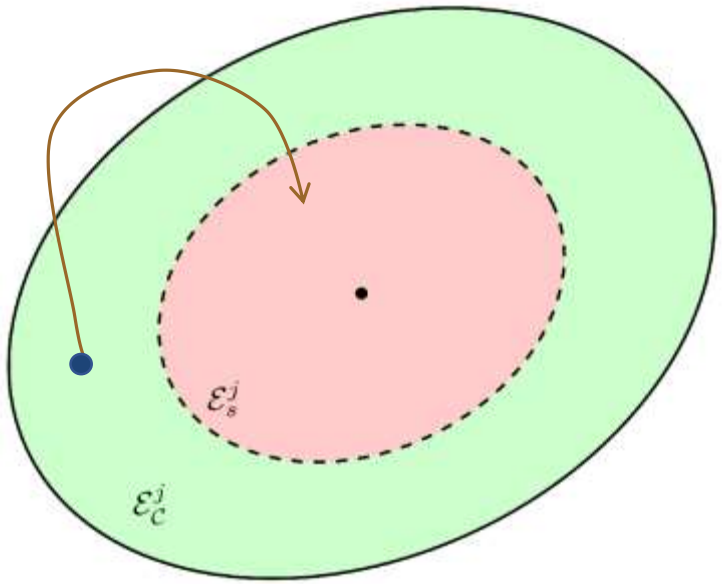


Software Rejuvenation: Robustness





$u = -K\hat{x}$



$$\mathcal{E}_c^j = \{x : x^T P x < 1\} \qquad \mathcal{E}_s^j = \{x : x^T P x < s\}\, 0 < s < 1$$

Software Rejuvenation: Robustness

System

$$\dot{x} = Ax + Bu + Dd, \quad \|d\| \leq \bar{d}$$
$$y = Cx + n, \quad \|n\| \leq \bar{n},$$

Observer

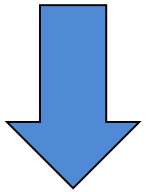
$$\dot{\hat{x}} = A\hat{x} + Bu - L(C\hat{x} - y)$$
$$= A_o\hat{x} + Bu + LCx + Ln,$$

Controller

$$u = -K\hat{x},$$

Estimation Error

$$e = \hat{x} - x$$



$$\dot{x} = A_{cl}x - BKe + Dd$$
$$\dot{e} = A_o e - Dd + Ln$$

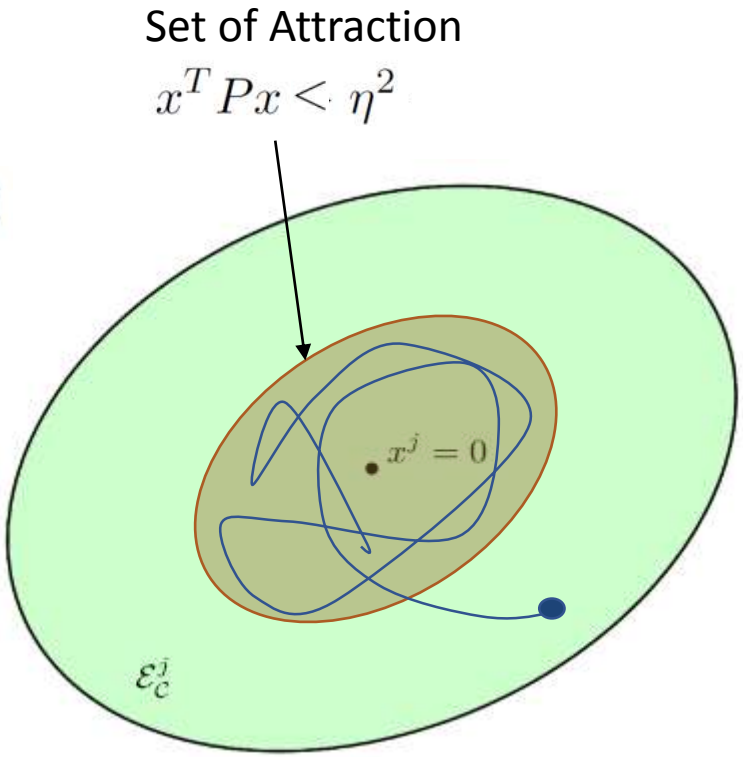
Quadratic Boundedness

$$\dot{x} = A_{cl}x + B_\delta \delta, \quad \|\delta\| \leq \bar{\delta}$$

is **quadratic bounded** (QB) with parameters $\eta > 0, \gamma > 0$ if there exists a symmetric definite positive matrix P such that

$$x^T P x > \eta^2 \rightarrow x^T P(\Gamma x + B_\delta \delta) < -\gamma \quad \forall \ \|\delta\| \leq \bar{\delta}.$$

$$\dot{x} = A_{cl}x \Rightarrow P \Rightarrow \eta > 0, \gamma > 0$$



Quadratic Boundedness

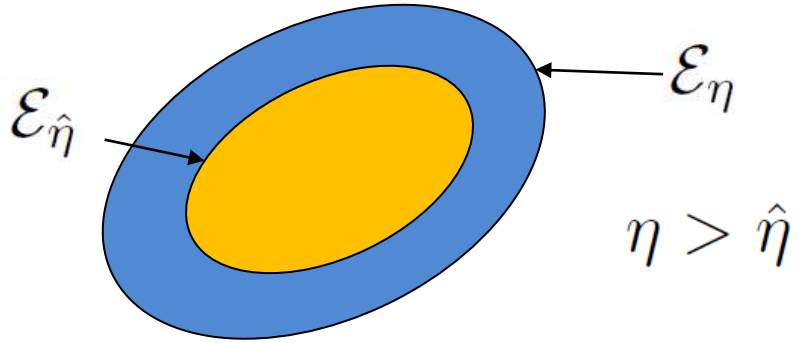
Theorem

$$\dot{x} = A_{cl}x + B_\delta \delta, \quad \|\delta\| \leq \bar{\delta}$$

$$P > 0, Q > 0 \quad Q = -A_{cl}^T P - P A_{cl}$$

$$m \triangleq \sqrt{\overline{\lambda}(P^{1/2} B_\delta B_\delta^T P^{1/2})}$$

$$q \triangleq \underline{\lambda}(P^{-1/2} Q P^{-1/2})$$



$$\hat{\eta} > 0$$

$$x^T P x = \hat{\eta}^2 \rightarrow \frac{d(x^T P x)}{dt} \leq -\hat{\gamma}(\hat{\eta}) \; \forall \; \|\delta\| \leq \bar{\delta}$$

$$\hat{\gamma}(\hat{\eta}) = q\hat{\eta}^2 - 2m\bar{\delta}\hat{\eta}$$

$$\eta = \frac{\bar{\delta} m + \sqrt{\bar{\delta}^2 m^2 + q\gamma}}{q}$$

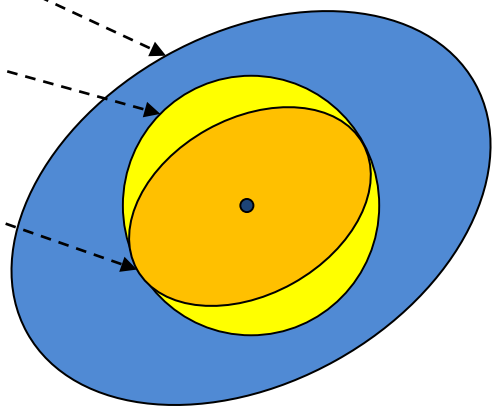
Robustness: design procedure

$$\dot{x} = A_{cl}x \longrightarrow P_p, Q_p \longrightarrow \mathcal{E}_1(P_p)$$

$$\dot{x} = A_{cl}x - BKe + Dd \longrightarrow \bar{e}_1 \triangleq \frac{q_p}{2m_p} - \bar{d}, \longrightarrow \mathcal{B}_{\bar{e}_1}$$

$$\mathcal{B}_{\bar{e}_1} \longrightarrow \epsilon_1 = \bar{e}_1\sqrt{\underline{\lambda}(P_p)}. \longrightarrow \mathcal{E}_{\epsilon_1}(P_p)$$

After each software refresh $\hat{x} = 0$



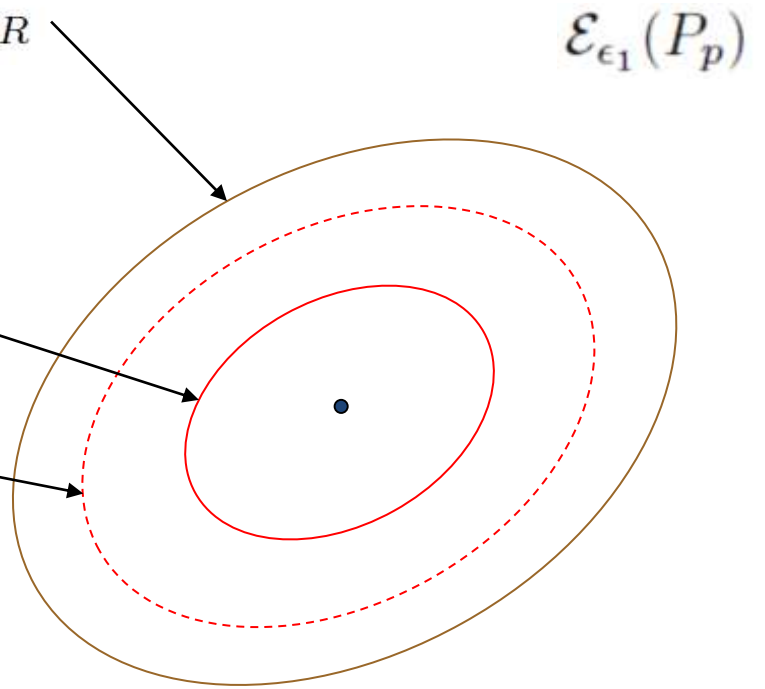
Robustness: design procedure

$\mathcal{E}_{\epsilon_1}(P_p)$ →

- $\bar{\epsilon}_s$ → $\mathcal{R}(t, \mathcal{E}_{\bar{\epsilon}_s}(P_p), U) \subseteq \mathcal{E}_{\epsilon_1}(P_p),\ 0 \le t \le T_{SR}$
- $\underline{\epsilon}_s \triangleq \underline{\eta}_{p,min} + 2\bar{e}_{min}\sqrt{\bar{\lambda}(P_p)}$

→ $\epsilon_s \in (\underline{\epsilon}_s, \bar{\epsilon}_s)$

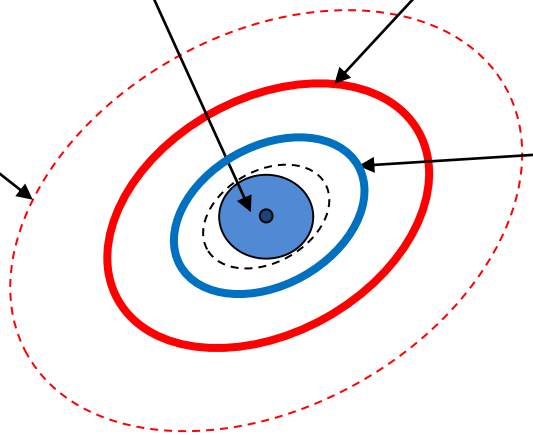
$\mathcal{R}(t, \mathcal{E}_{\epsilon_s}(P_p), U) \subseteq \mathcal{E}_{\epsilon_1}(P_p),\ 0 \le t \le T_{UC}$



$$\dot{e} = A_o e - Dd + Ln.$$

Robustness: design procedure

$$\epsilon_s \in (\underline{\epsilon}_s, \bar{\epsilon}_s) \longrightarrow \bar{e}^* \triangleq \frac{q_p \epsilon_s - 2m_p \bar{d}}{2q_p \sqrt{\bar{\lambda}(P_p)} + 2m_p}, \longrightarrow \epsilon_{\hat{s}} = \epsilon_s - \bar{e}^* \sqrt{\bar{\lambda}(P_p)}. \longrightarrow \hat{x} \in \mathcal{E}_{\epsilon_{\hat{s}}} \Rightarrow x \in \mathcal{E}_{\epsilon_s}$$



$$\underline{\eta}^* = \epsilon_{\hat{s}} - \bar{e}^* \sqrt{\bar{\lambda}(P_p)}.$$

set of attraction

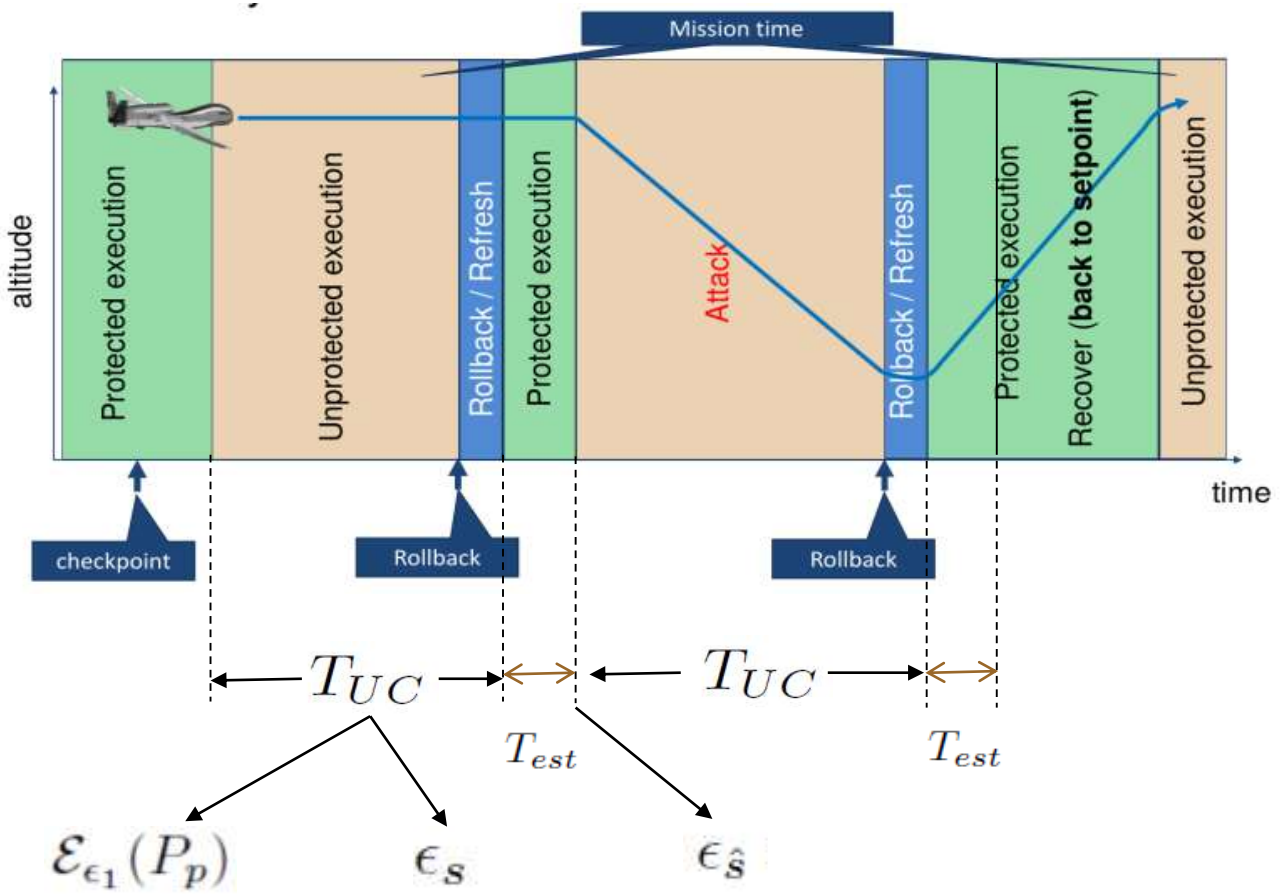
recovery in a finite amount of time

Robustness: design procedure

After each software refresh

$$\bar{e}_1 \longrightarrow \bar{e}^* \Longrightarrow T_{est}$$

$$\dot{e} = A_o e - Dd + Ln.$$



Robustness: Results

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u + \begin{bmatrix} 0 \\ 1 \end{bmatrix} d$$

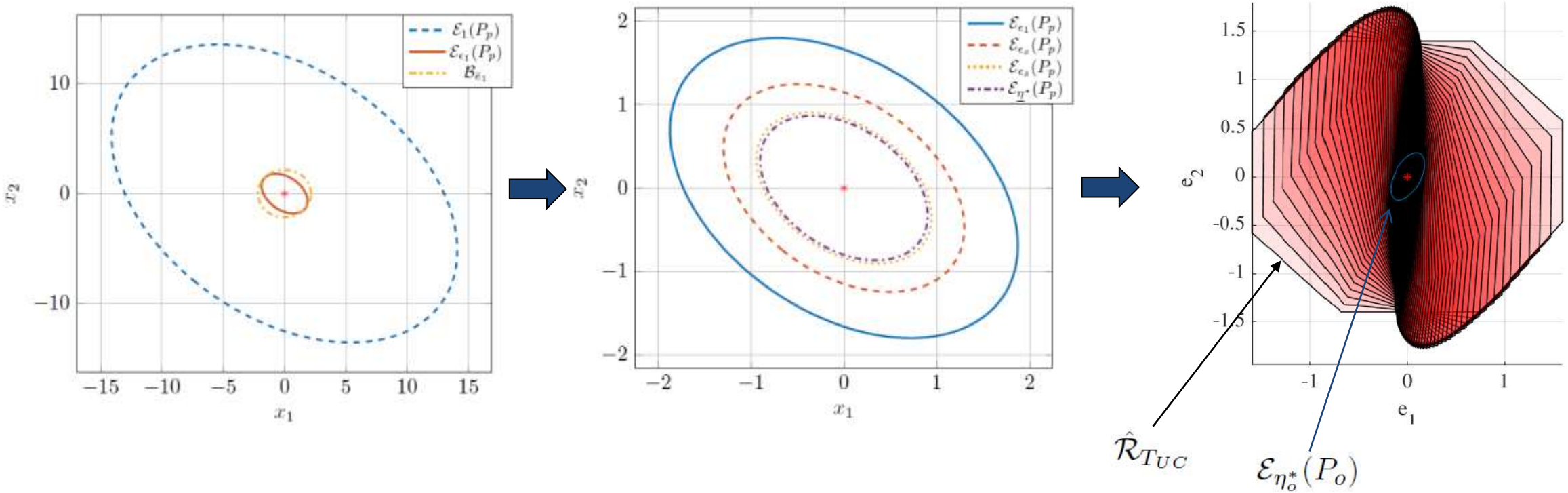
$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + n$$

$e_1 = 2.1607$ $\quad \epsilon_1 = 0.133$ $\quad \bar{e}_{min} = 0.129$

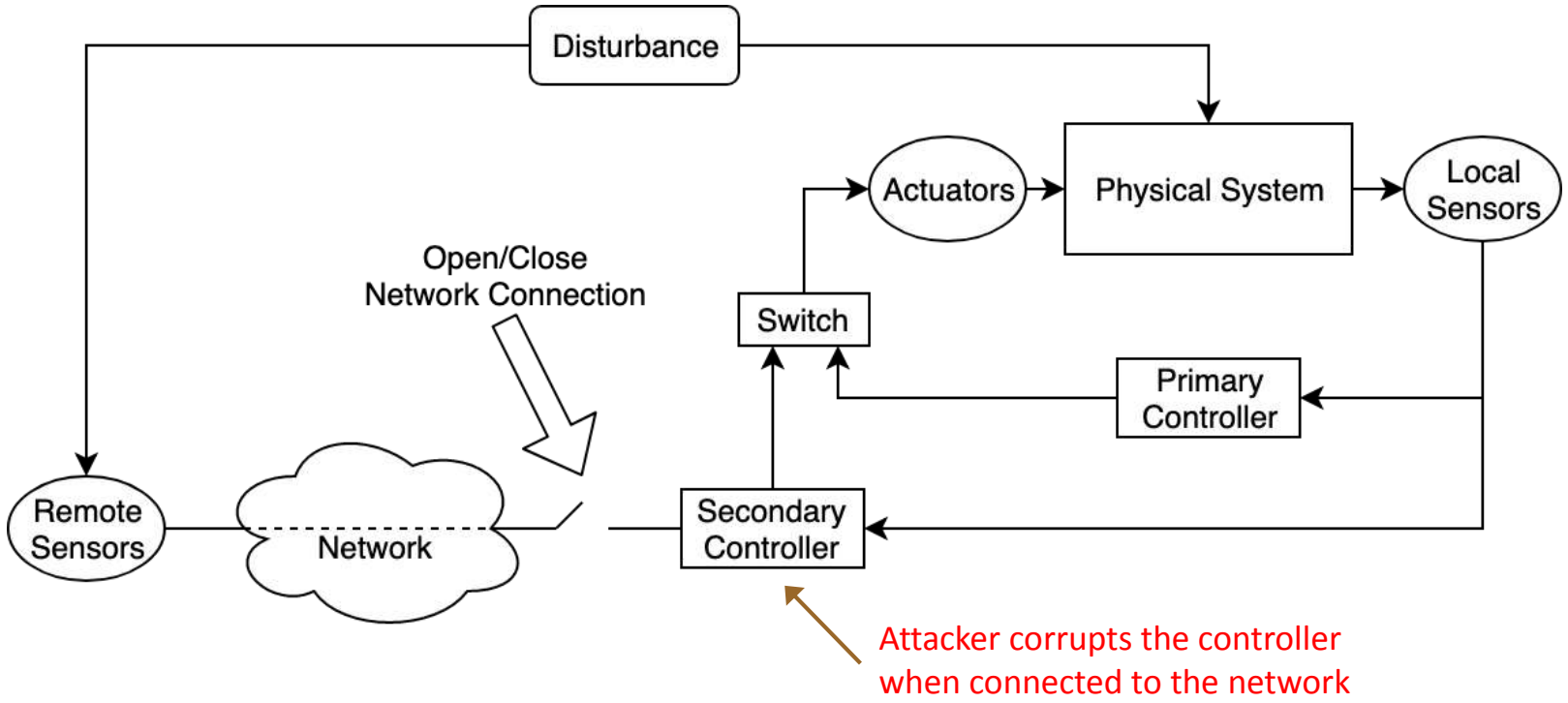
$\bar{\epsilon}_s = 0.113$ $\quad \underline{\epsilon}_s = 0.0878$ $\quad \underline{\eta}_{p.min} = 0.064$

$\epsilon_s = 0.092$ $\quad \bar{e}^* = 0.1354 > \bar{e}_{min}$

$T_{UC} = 0.32$ s $\quad T_{est} = 1.05$ s $\quad \epsilon_{\hat{s}} = 0.067$



Software Rejuvenation for a complementary scenario



Disturbances may take the system state near an unsafe region

To recover from a dangerous situation, it is sometimes necessary to connect to the network

System Model

System

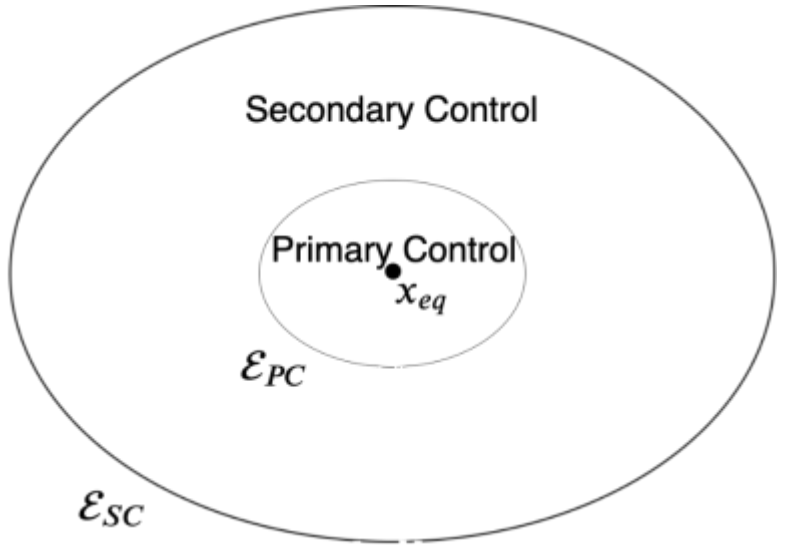
$$\dot{x} = Ax + Bu + Dd$$

Primary Controller (PC)

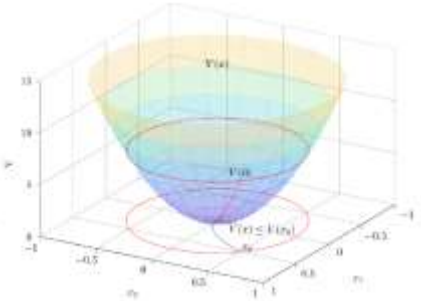
- Activated when not connected to the network
- Stabilizes the system at x_{eq} when $d=0$
- $u_{PC} = -K_{PC}x$

Secondary Controller (SC)

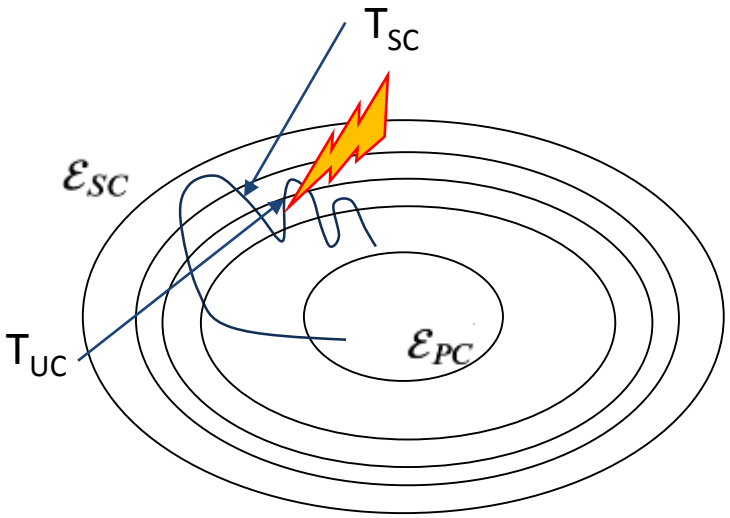
- Activated when connected to the network
- Remotely measures the disturbance and rejects it
- $u_{SC} = -K_{PC}x - K_{SC}d \qquad BK_{SC} = D$

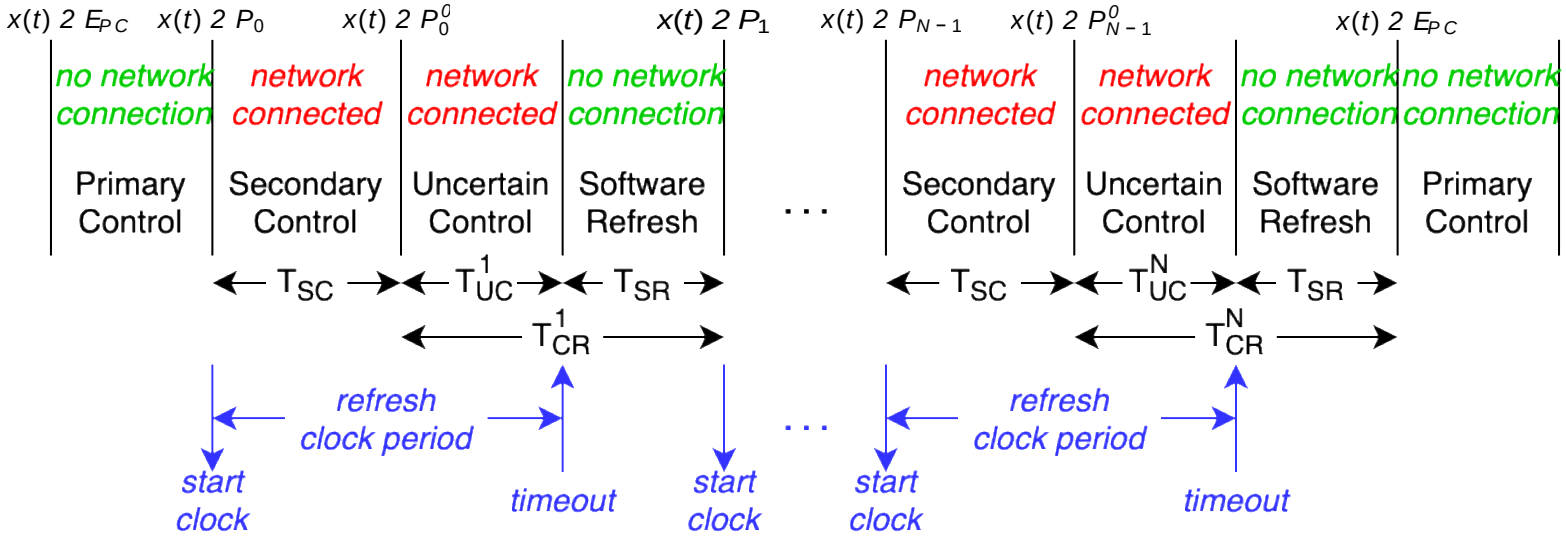


$$\dot{x} = (A - BK_{PC})x = A_{cl}x \longrightarrow V(x) > 0, V(0) = 0, \dot{V}(x) < 0$$
$$P > 0 : V(x) = x^T P x,$$
$$\dot{V}(x) = A_{cl}^T P + P A_{cl} < 0$$



Vulnerable Safety





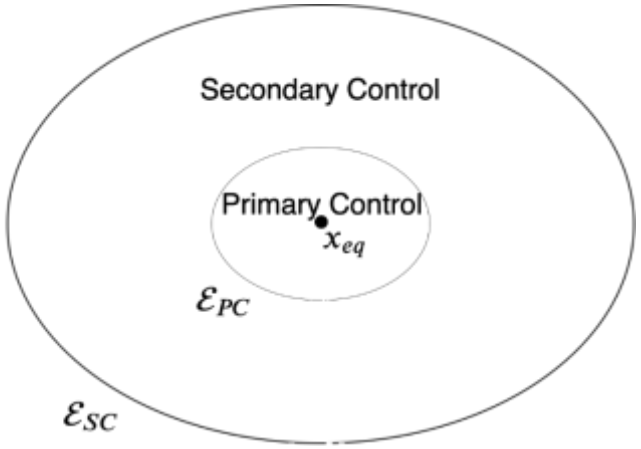
By connecting to the network, the system becomes vulnerable to attacks to ensure safe recovery from disturbances

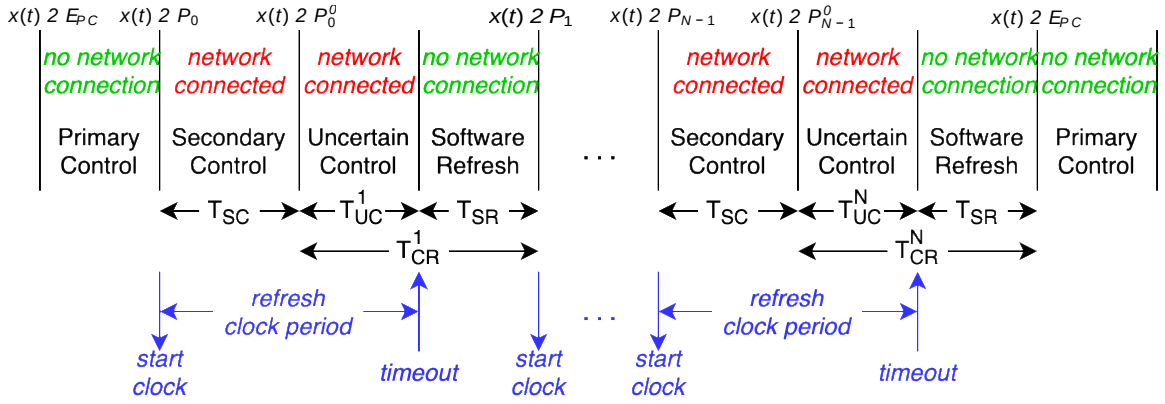
We utilize software rejuvenation to ensure that SC is able to drive $x(t)$ back to $\mathcal{E}_{PC}$ after a disturbance

Algorithm

Algorithm 1 Software Rejuvenation Algorithm

```
1:  Close Network Connection
2:  while 1
3:      while x(t) ∈ E_PC
4:          Primary Control (protected)
5:      end while
6:      for i = 1 : N
7:          t = 0 (initialize and begin timer)
8:          Open Network Connection
9:          while t < T_SC + T_UC^i
10:             Secondary Control (insecure)
11:         end while
12:         Close Network Connection
13:         Software Refresh
14:         if x(t) ∈ E_PC
15:             break
16:         end if
17:     end for
18: end while
```





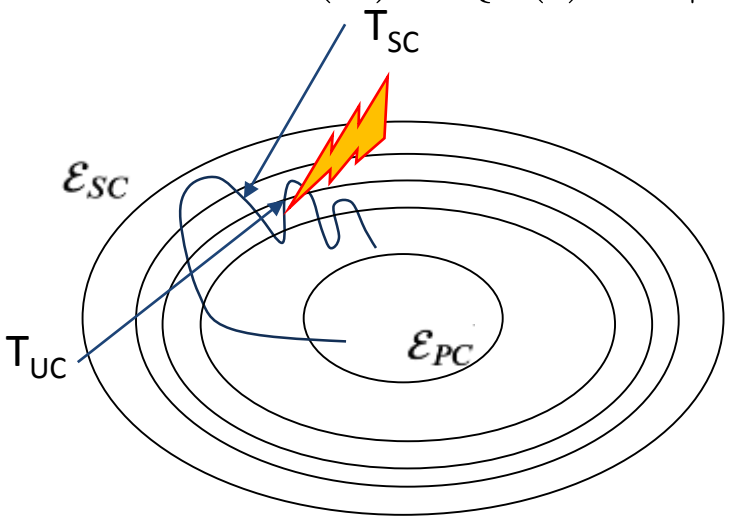
Safety Conditions

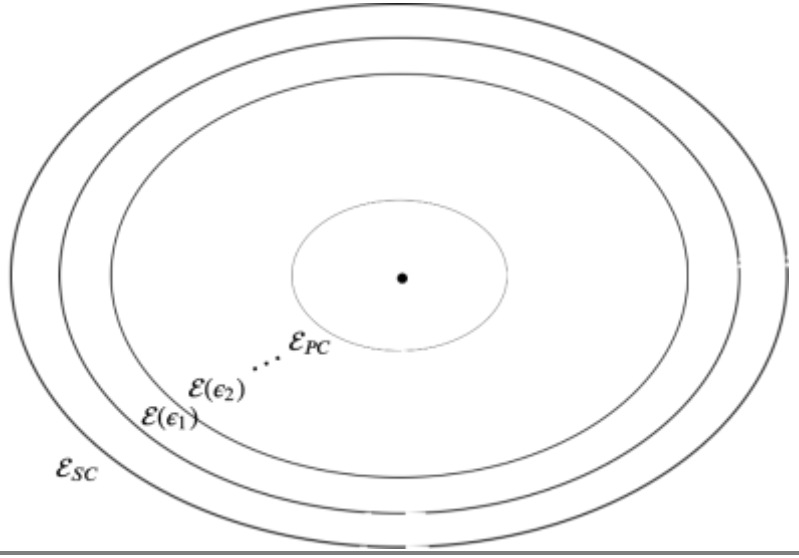
Theorem

Given specific values of T_{SC} and T_{SR}, the state of the system can always be driven back to $\mathcal{E}_{PC}$ within N software refreshes if there exists a sequence of T_{UC}^{i} such that $\epsilon_N \leq \epsilon$ and $\mathcal{P}_i \subseteq \mathcal{E}(\epsilon_i)$ for $i = 1, \cdots, N$.

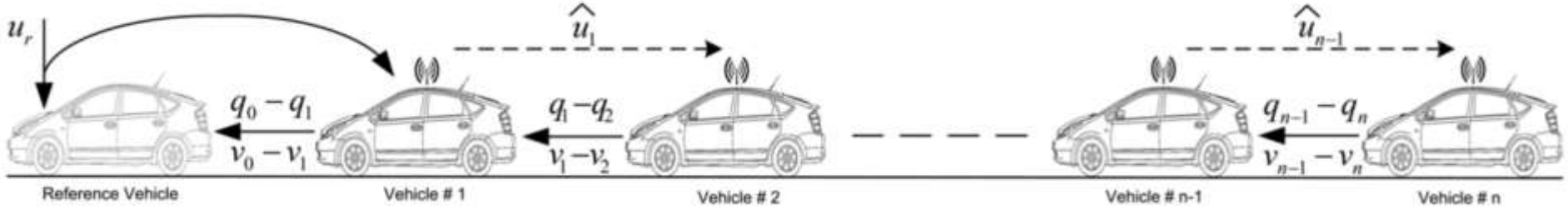
Definitions

- $\mathcal{P}_i \triangleq$ a convex polytope over-approximating the set of all possible states after the i^{th} software refresh
- $\mathcal{E}(\epsilon_i) \triangleq \{x(t) \in \mathcal{C} | V_{SC}(x(t)) \leq \epsilon_i,\ \epsilon_i \in [0,1],\ \epsilon_i > \epsilon_{i+1}\}$

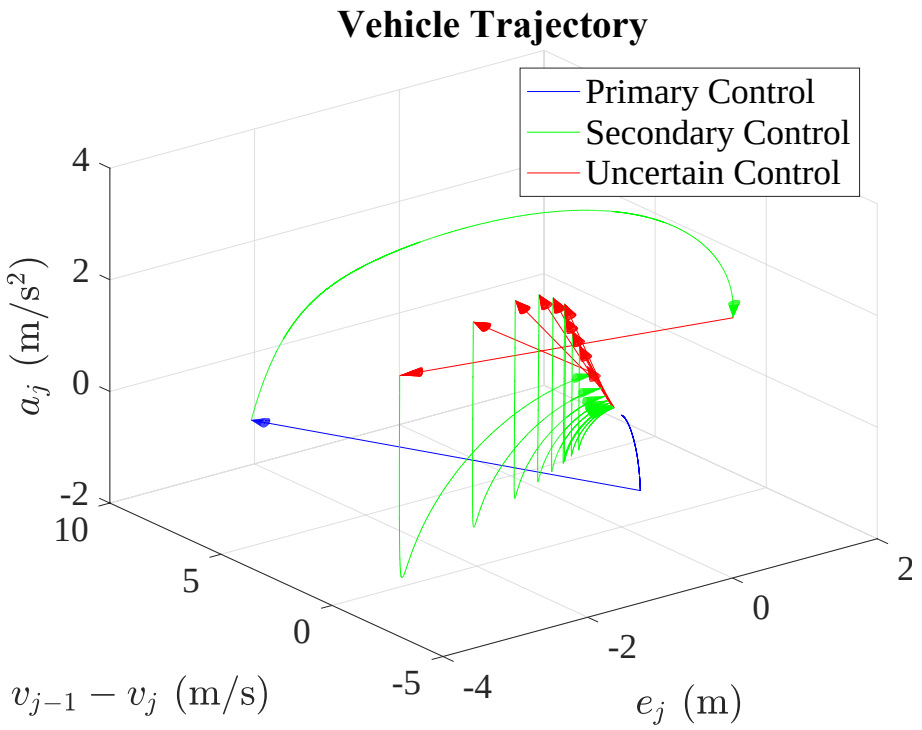


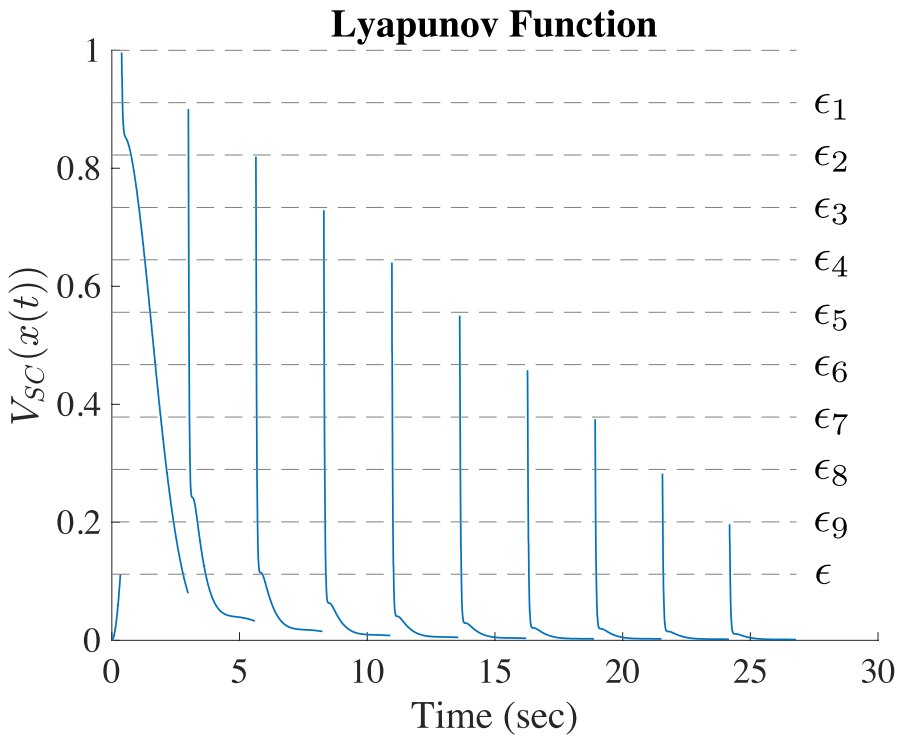


Simulations: Car Platooning



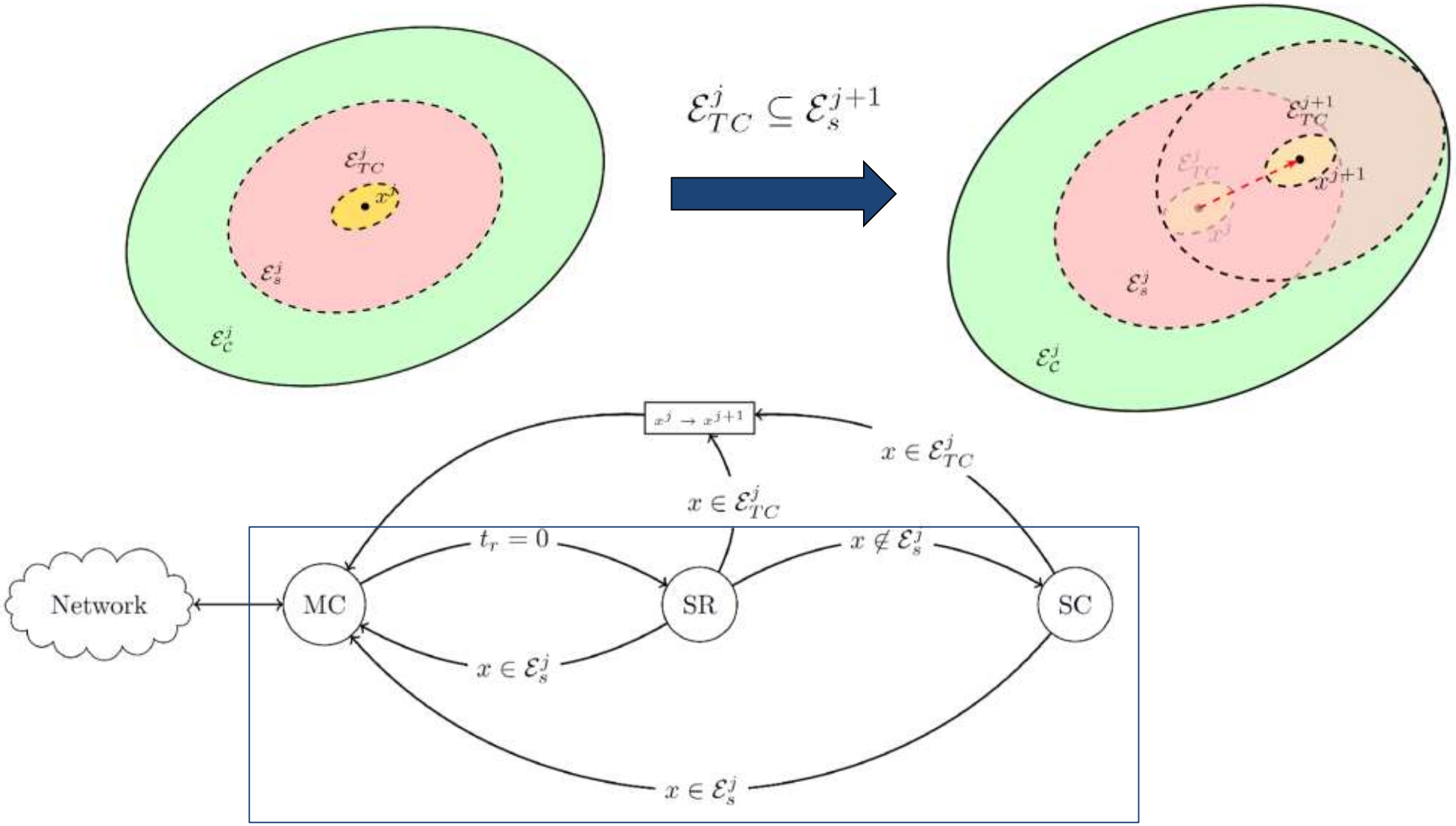
S. Oncu et al., "String stability of interconnected vehicles under communication constraints." *51st CDC*, Maui, HI, 2012, pp. 2459-2464.



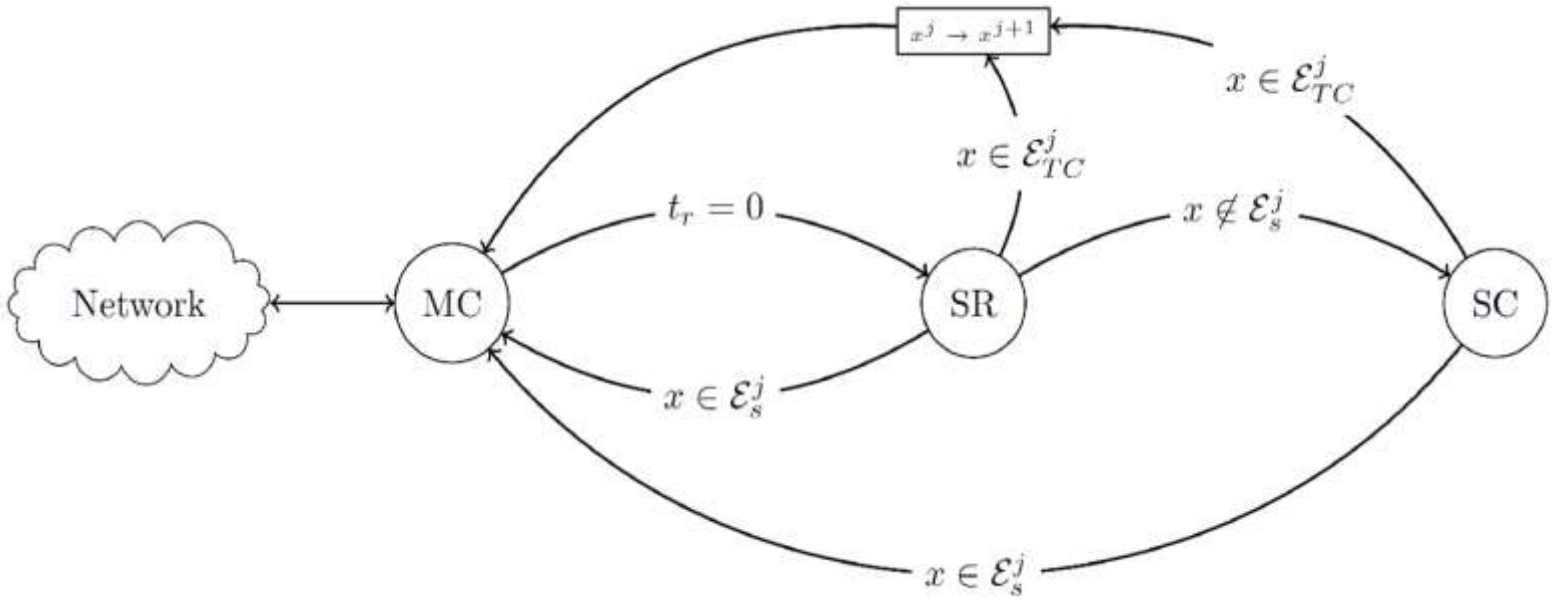


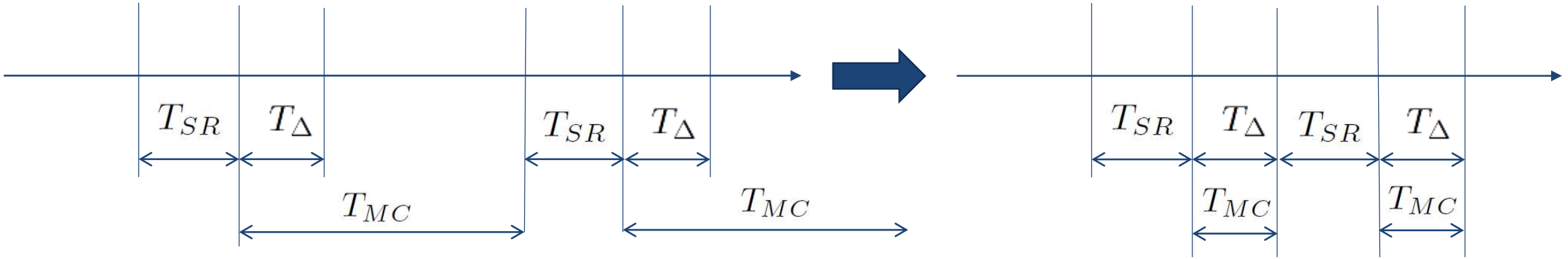
$$T_{SW} = 0.05\text{s}, T_{SC} = 2.6\text{s}, T_{SR} = 0.01\text{s}, N = 10$$

Software Rejuvenation: Persistent attack

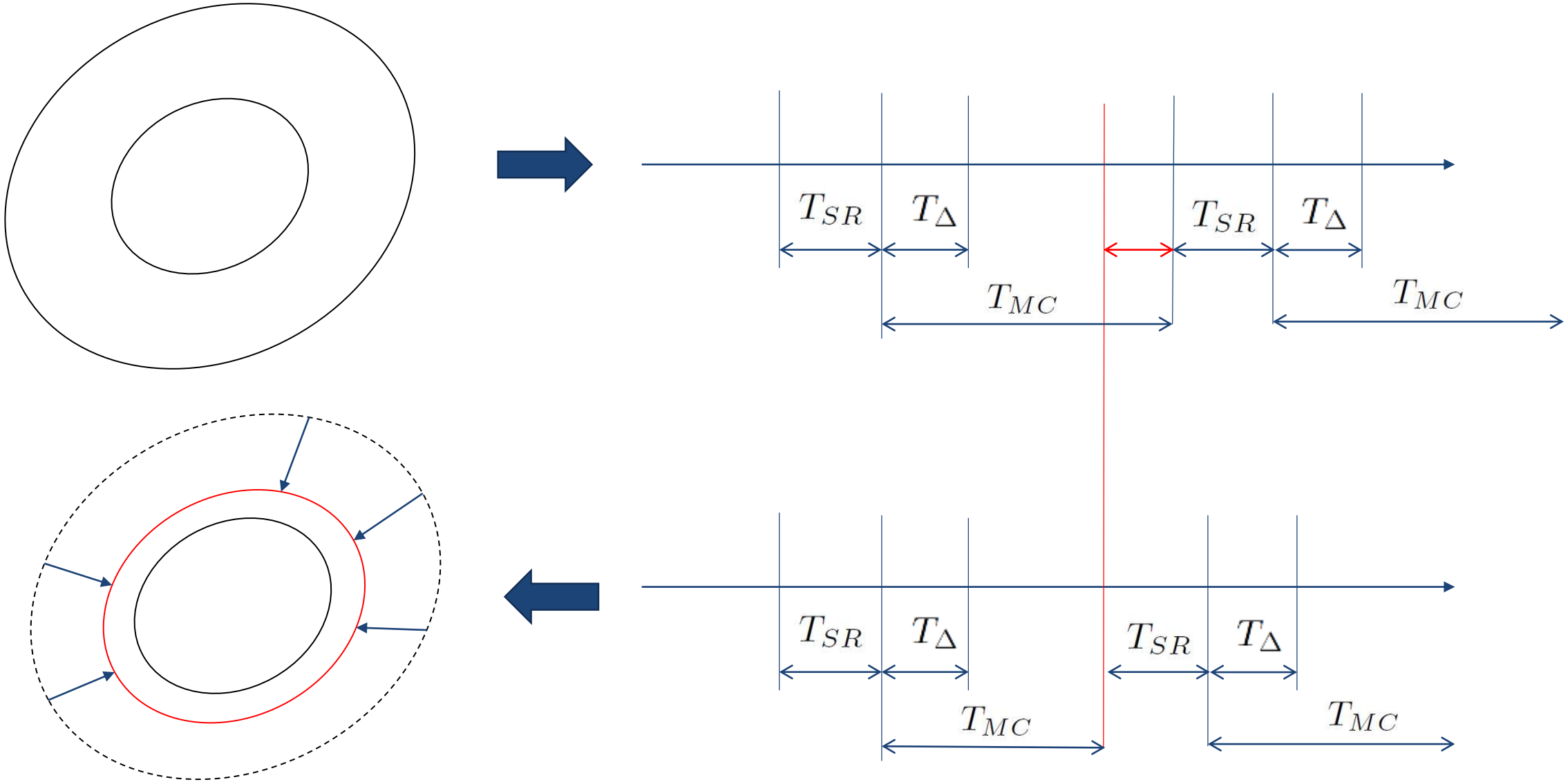


Software Rejuvenation: Persistent attack





Software Rejuvenation: Constrained Environment



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