Online Control Adaptation for Safe and Secure Autonomous Vehicle Operations

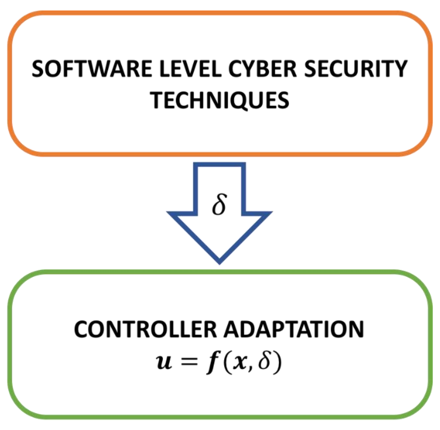
AHS'17

Mahmoud Elnaggar[1], Jason D. Hiser[2], Tony X. Lin[3], Anh Nguyen-Tuong[2], Michele Co[2], Jack W. Davidson[2], and Nicola Bezzo[1,3]

[1]Department of Systems and Information Engineering, [2]Department of Computer Science, [3]Department of Electrical and Computer Engineering
University of Virginia

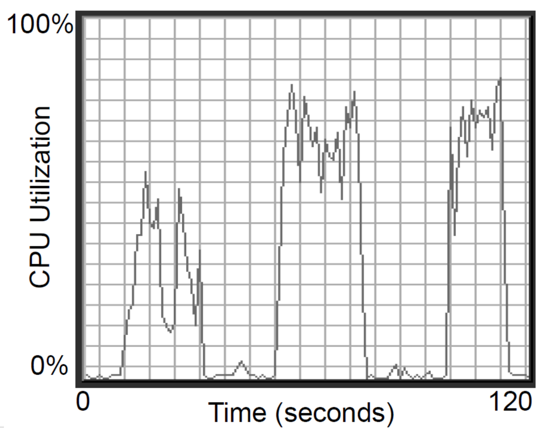
INTRODUCTION

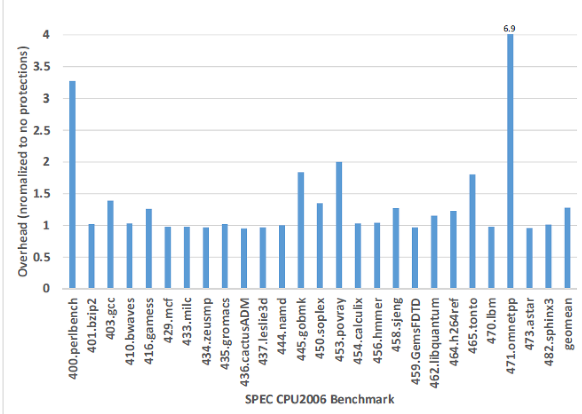
- Modern cyber-physical systems are not built with cyber-security in mind.
- Adding cyber-security incurs runtime **overheads** that results in **performance degradation** and **safety issues**.

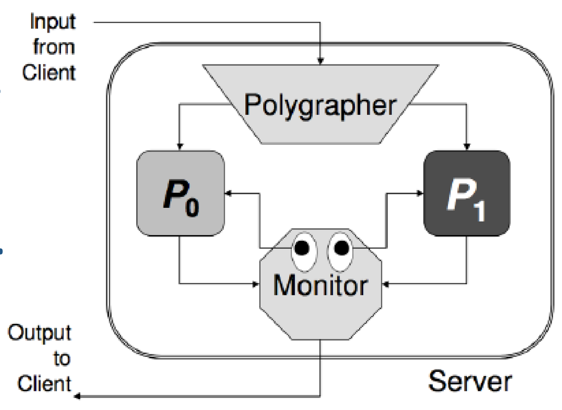


CYBER-SECURITY TECHNIQUES

- **System-level software security** techniques:
  - Anti-virus and intrusion-detection systems. monitor the entire system for indications of compromise.
- **Application-level software security techniques:**
  - Control Flow Integrity (CFI):
    - Instruments the application to enforce the intended control flow at run time.
    - The overhead on an application varies depending on the inputs it processes.
  - N-variant system (*Double Helix*[1]):
    - Employs the systematic application of artificial diversity to prevent large classes of attacks.
    - Provides formal proofs that certain classes of attacks are not possible and system can recover from attacks and continue operation.
    - The performance overheads can reach up to 400%.







PROBLEM FORMULATION

- An autonomous vehicle (AV) is tasked to complete a mission over an obstacle populated environment $W = F \cup O$
- The set F represents the obstacle-free region of the environment and vice versa O is the region occupied by obstacles.
- The discrete dynamical model of the AV:

$$\boldsymbol{x}(k+1) = \boldsymbol{A}'_d(\delta(k))\boldsymbol{x}(k) + \boldsymbol{B}'_d(\delta(k))\boldsymbol{u}(k)$$
$$\boldsymbol{y}(k) = \boldsymbol{C}\boldsymbol{x}(k)$$

$$\boldsymbol{A}'_d(\delta(k)) = e^{\boldsymbol{A}(t_s+\delta(k))}$$
$$\boldsymbol{B}'_d(\delta(k)) = \int_0^{t_s+\delta(k)} e^{\boldsymbol{A}\lambda}\boldsymbol{B}d\lambda$$

- Given the current state of the system x, the desired input u with no delay, and the maximum expected delay δ_{max} the objective is to find a **control policy** $\hat{u} = f(x, u, \delta_{max}, t)$ such that $x(t) \not\subset O, \forall t \geq 0, x(0) \subset F$.

ONLINE CONTROLLER ADAPTATION

- Model Predictive Control (MPC).

$$J(\boldsymbol{x}(k), \boldsymbol{u}(k)) = \min_{\boldsymbol{u}(k)} \sum_{k=0}^{h-1} \boldsymbol{e}_x^T(k)\boldsymbol{Q}\boldsymbol{e}_x(k) + \boldsymbol{e}_u^T(k)\boldsymbol{R}\boldsymbol{e}_u(k)$$
$$\text{subject to } \boldsymbol{x}(k+1) = \boldsymbol{A}'_d(\delta(k))\boldsymbol{x}(k) + \boldsymbol{B}'_d(\delta(k))\boldsymbol{u}(k)$$

- The delay is time varying and not known a priori.
- Estimation of the delay using exponential weighted moving average algorithm (EWMA). $\delta_e(k) = (1-\alpha)\delta_e(k-1) + \alpha\delta(k-1)$
- Unsafe regions are inflated to construct the set S that satisfies:

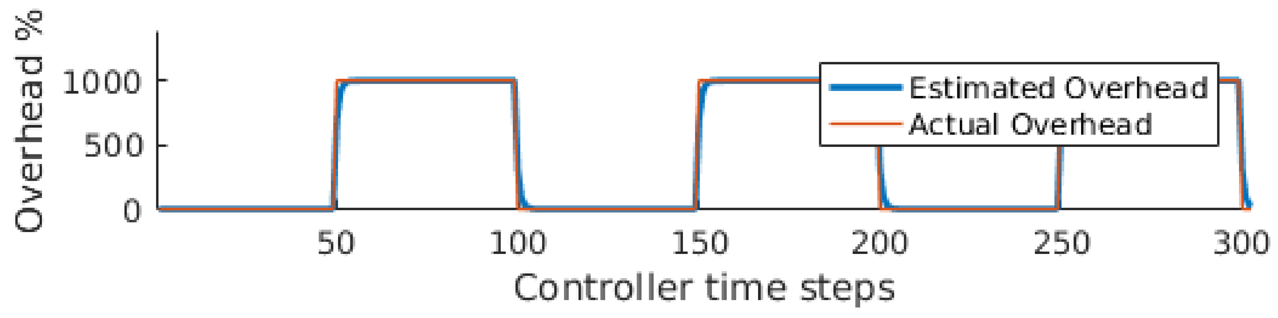
$$\forall \boldsymbol{x}(k) \subset \mathcal{S}, \exists \boldsymbol{u}_c \in \boldsymbol{U} \text{ such that } \boldsymbol{x}_{max}(k+1) \not\subset \mathcal{S}$$
$$\boldsymbol{x}_{max}(k+1) = \boldsymbol{A}'_d(\delta_{max})\boldsymbol{x}(k) + \boldsymbol{B}'_d(\delta_{max})\boldsymbol{u}_c$$

Risk-based approach:

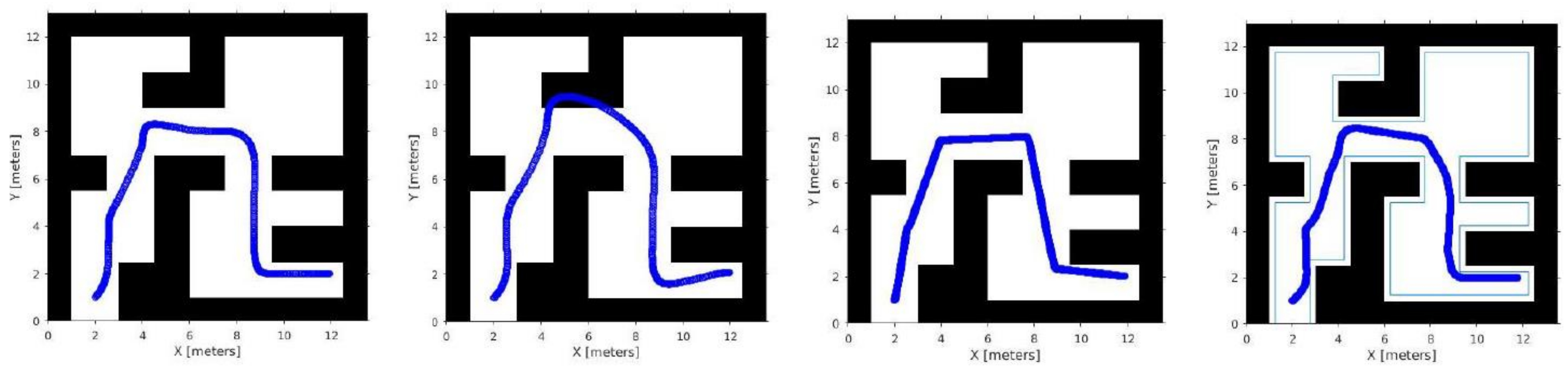
1. MPC is used to compute a controller input u_{max} considering maximum delay δ_{max}.
2. Computation of risk factor r that indicates the accuracy of the last estimated delay. $$r = \left\|\frac{\delta(k-1) - \delta_e(k-1)}{\delta_{max}}\right\|$$
3. Finally, the adapted controller input $\hat{u}(k)$ is applied to the AV. $$\hat{\boldsymbol{u}}(k) = \boldsymbol{u}_{max} + r\Delta\boldsymbol{u}$$ $$\Delta\boldsymbol{u} = \boldsymbol{u}_e - \boldsymbol{u}_{max}$$
4. If the state of the AV lies inside the inflated region, a conservative control input is generated. $\hat{\boldsymbol{u}}(k) = MPC_{Cons}(\delta_{max})$

[1]M. Co, et al., "Double helix and raven: A system for cyber fault tolerance and recovery", CISRC '16.

SIMULATIONS



Imposing cyclic controller runtime overheads



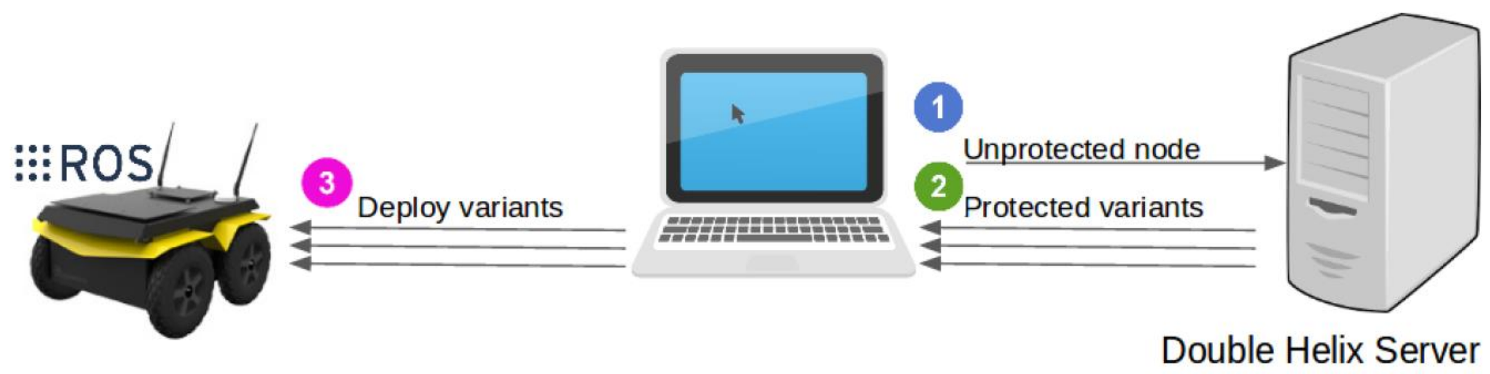
(a) No Overhead Delay (b) No adaptation (c) Conservative navigation (d) Adaptive navigation

EXPERIMENTS

Scenario	Task Execution Time(s)
No overhead	6.3
Overhead + conservative controller	43.4
Overhead + online adaptation	18.1

A. Integrating Software Level Cyber Security Techniques On A Real AV:

- *Double Helix* is used to protect ROS nodes for mapping and navigation.



- Performance overhead imposed by the protected controllers reached 30%.

Type of overhead	*Slam_gmapping*	*Move_base*
CPU load	17.8%	12.5%
Memory consumption	3.0%	2.56%

B. Online Adaptation Control With Unknown Overhead:

Scenario	Task Execution Time(s)
No overhead	34
Overhead + conservative controller	50
Overhead + online adaptation	42

Without online controller adaptation

With online controller adaptation

CONCLUSION AND FUTURE WORK

- Outcome: Trade-off between security, performance, to guarantee safety.
- Planned extension to unmanned aerial vehicles (UAVs).
- Use Machine Learning techniques to estimate the delays and adapt the controller input accordingly.