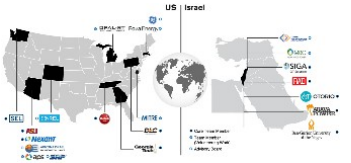
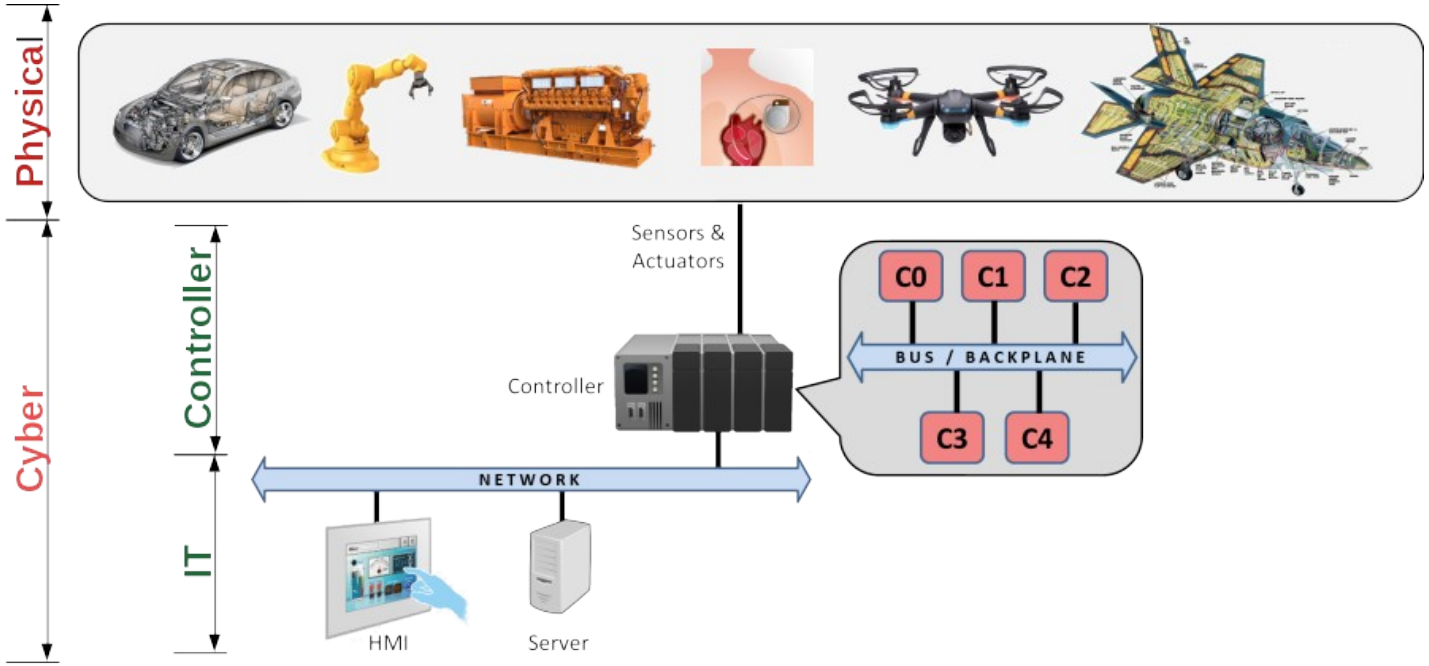
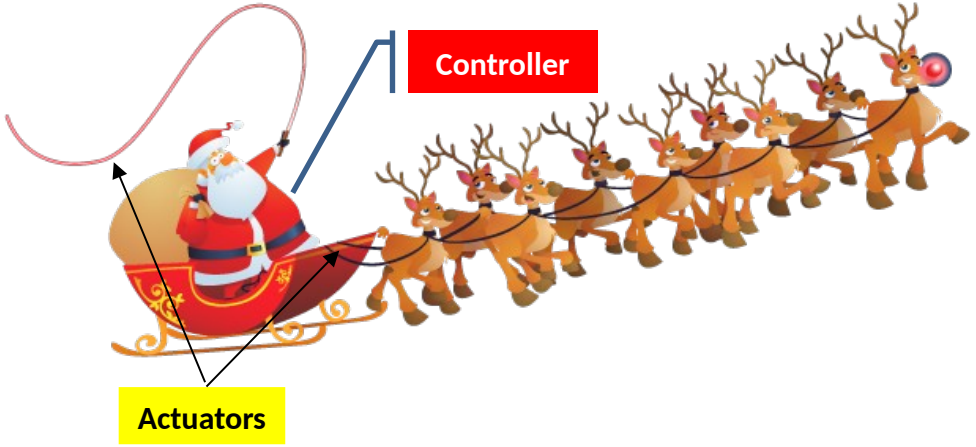


Cyber-Attack Resilience for CPS

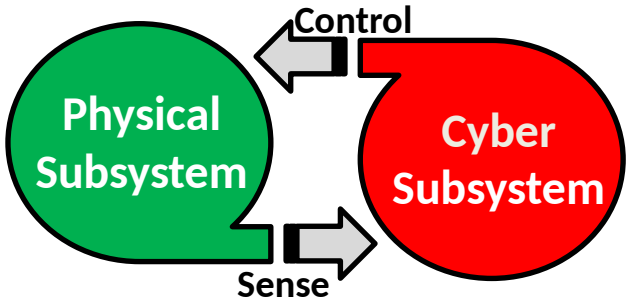
Dr. J. Sukarno Mertoguno
School of Cybersecurity & Privacy
Georgia Institute of Technology
karno@gatech.edu

CPS Infrastructure





Cyber Physical System



CPS: 2 loosely coupled subsystems

- **Physical** Subsystems, governed by physics
- Controller (**Cyber**) Subsystems, periodically sense/monitor & control Physical Subsystems

Goal: to have the **physical systems** behave properly and as expected, regardless of fault or disruption (cyber or otherwise).

CPS stability concerns

Cyber **P**hysical **S**ystems

Supporting
Is a "means to an end"

Primary Concern
"Stability & Integrity"

Cyber-attack resilient solutions should be primarily defined and motivated by **physical requirements**

The goal is for the **physical** subsystem to be **stable**, and **not** necessarily the **cyber** subsystems

CPS controller properties

Periodicity

- Continuous observe and control loop (scan cycle, usually ~1-300 Hz)

Tolerates some loss of inputs or data

- Sensitive to latency variations
- Not performing open-ended, general-purpose tasks like IT

Inertia

- Physical systems have ***inertia***
- Effect: can tolerate some bad cycles and still maintain stability
  - Missed output
  - Wrong output (sensor blip, etc.)
- In context of cyber attack:

Inertia provides some natural fault tolerance

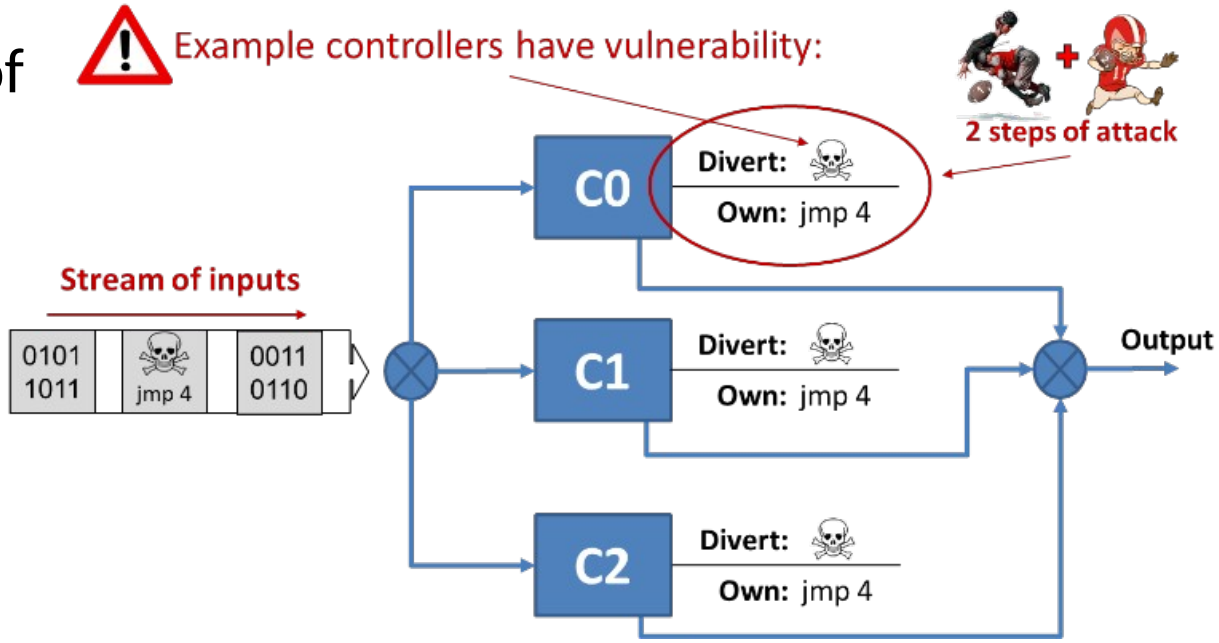
- Not immediately uncorrectable
- How long is system-dependent

Fault Tolerance and Cyber Attack

Many systems already employ some type of **fault tolerance** for **physical and random** failures:

- Redundancy with voting/consensus
- Quad Redundant Control (QRC)
- Byzantine Fault Tolerance (BFT)

Cyber attack → Common Mode Failure



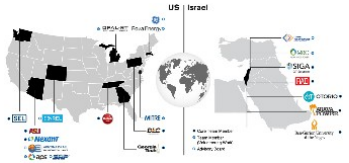
Successful attack requires:

1. Success on derailing targeted program --> targeted program loses control
2. Success on capturing control --> attacker controls program execution

How to transform Fault Tolerant into attack Tolerant?

BFT++

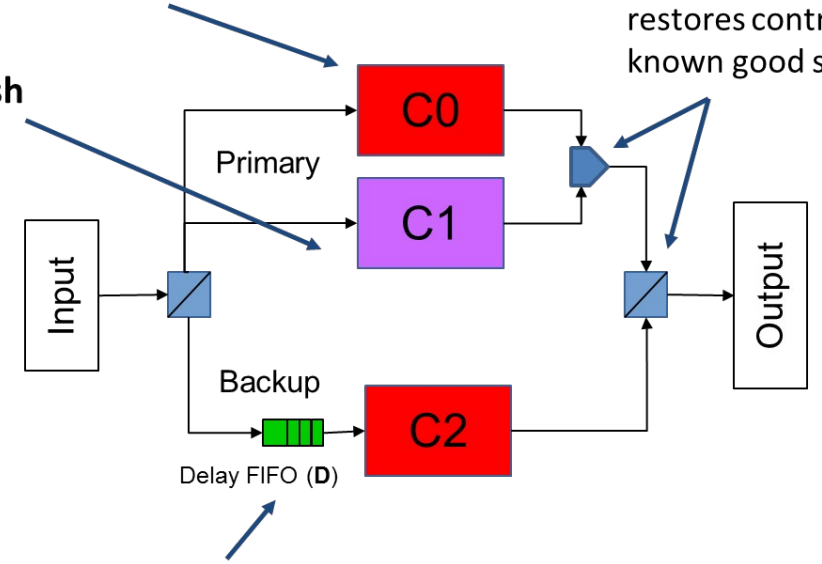
Techniques work together to provide resiliency against cyber attack-induced failures



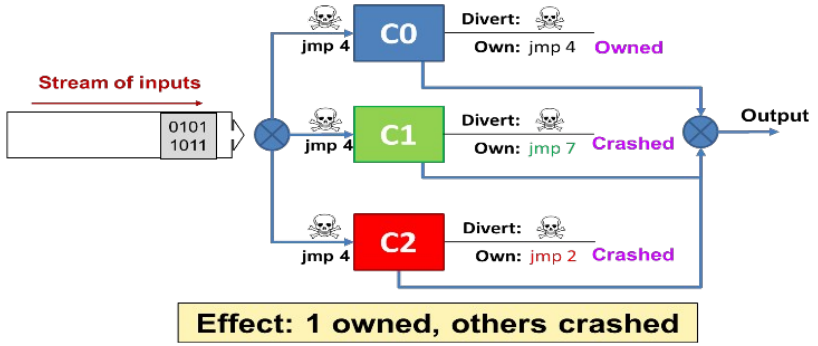
Artificial diversity
Protects controllers from being owned

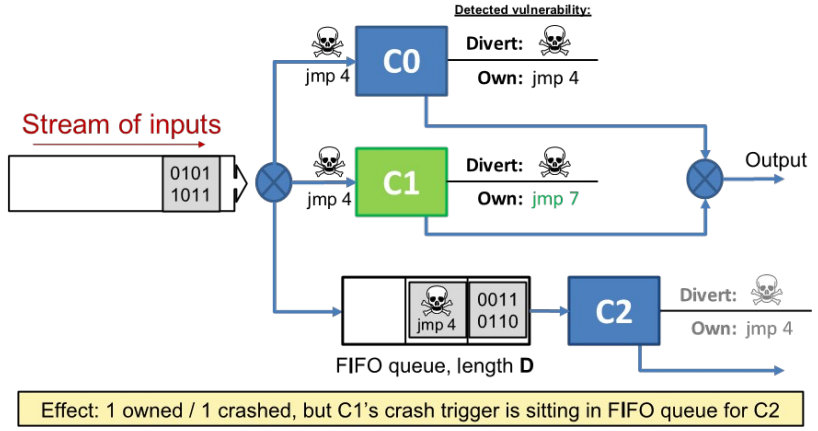
Controller recovery
Detects variance and restores controller(s) to known good state

Engineered FastCrash
Limits time spent in corrupted state



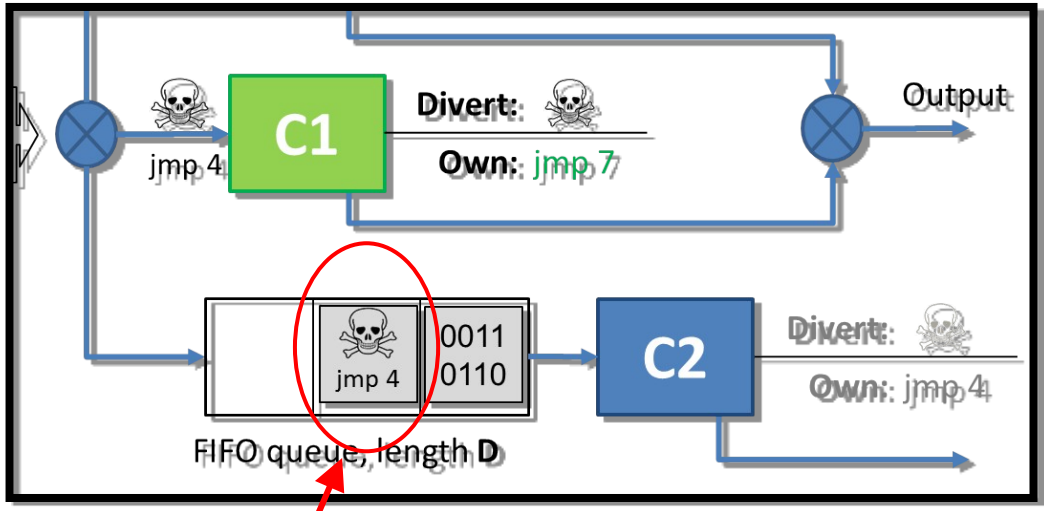
Delayed input sharing
Keeps backup in clean state and isolates malicious trigger





BFT++ is a Mechanism and is knowledge independent, hence deterministic, simple and robust

BFT++ applicability



- **Malicious** input is **captured** within Delay FIFO
- Can be sent to **SCATOPSY** for analysis
- Can be filtered out in the future

BFT++ is applicable when:

$$T_{crash} \leq D * T_{SC} \leq T_d - T_r$$

(system dependent)

T_{crash}	=	Time/latency for engineered crash once corrupted
T_{SC}	=	Scan Cycle Period
D	=	Time delay for backup system (length of FIFO queue, unit = # of epoch or scan-cycle)
T_d	=	Maximum control loss tolerable by physical system
T_r	=	Recovery latency

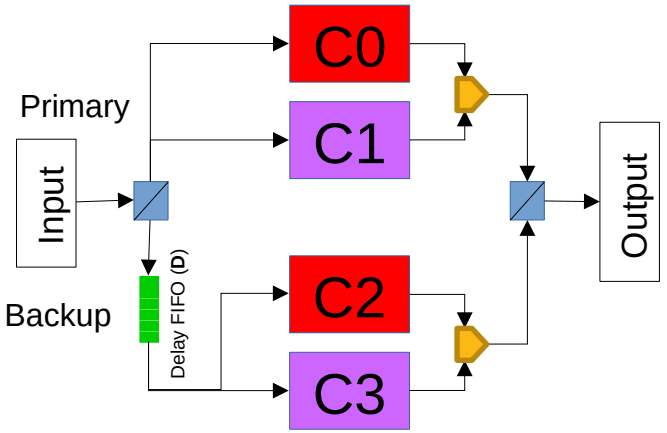
Quicker system crashes → Shorter erroneous period → More Resilience System

Brittle is Better !!!

BFT++ variants

BFT++ v1 (Vanilla) – NRL

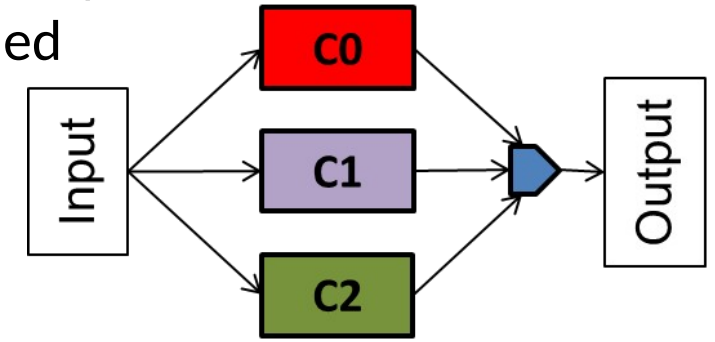
The Original implemented on QRC



BFT++ v2 – Georgia Tech

More robust, more costly: protection is diversified

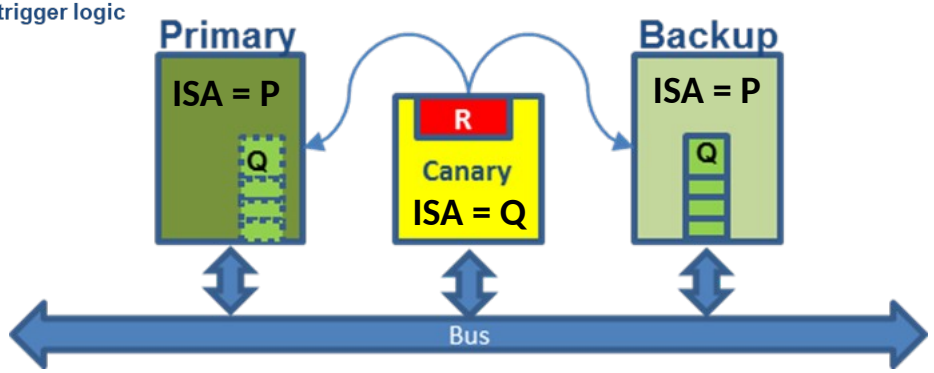
Diversified protection sets provides comprehensive protection with reasonable distributed overhead



BFT++ v4 (Rum Raisin)

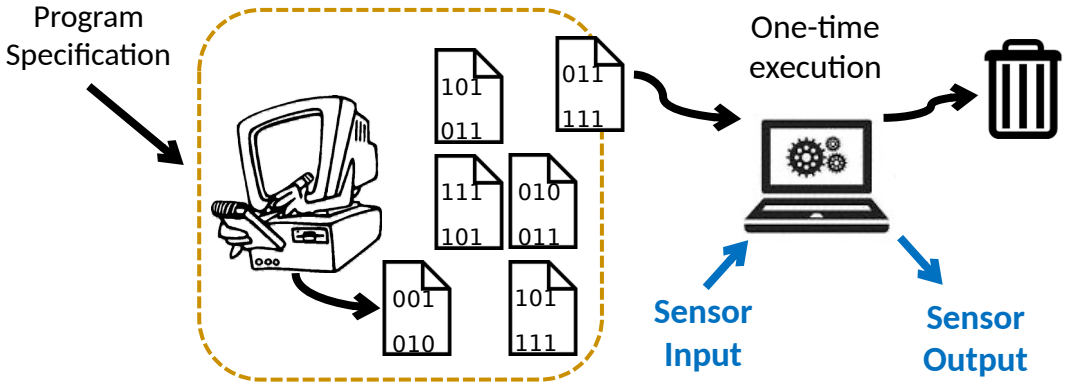
Variation of the original: ISA diversity

Q = delay queue
R = recovery trigger logic

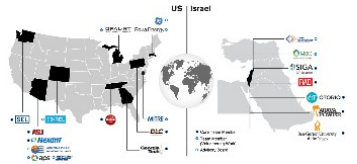


BFT++ v3 – Columbia University

Lightweight, probabilistic guarantee: requires no redundancy

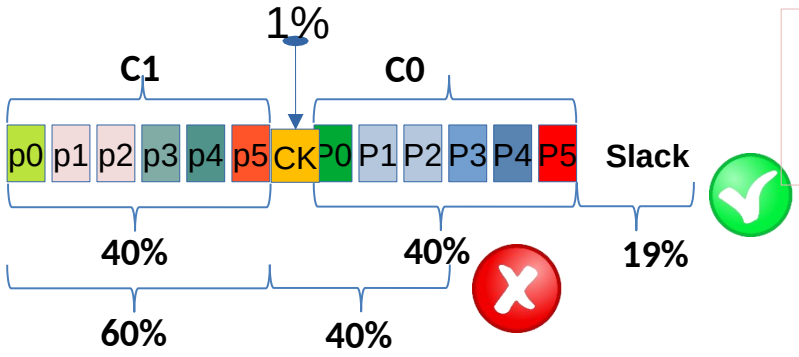


Sub-Process BFT++:
Diversified Redundancy on Single Processor



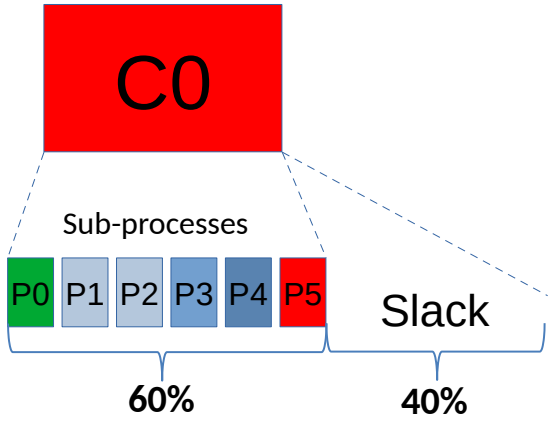
Design parameters:

- Slack availability
- Number of sensitive processes
  - Depth relative to input
- Pre-emptive vs. Co-operative scheduling

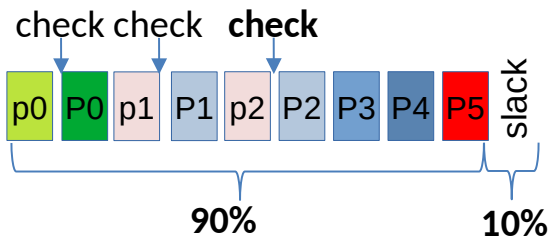


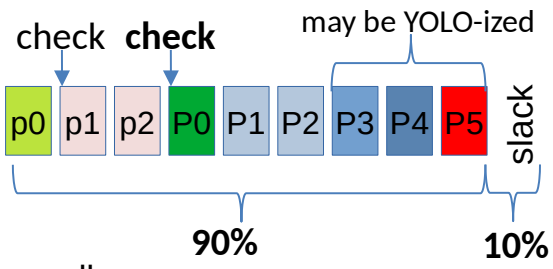
Legends:

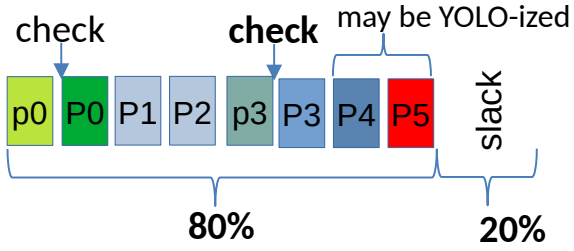
- **C0** Original program
- **C1** Diversified program
- If **check** fails, drop input

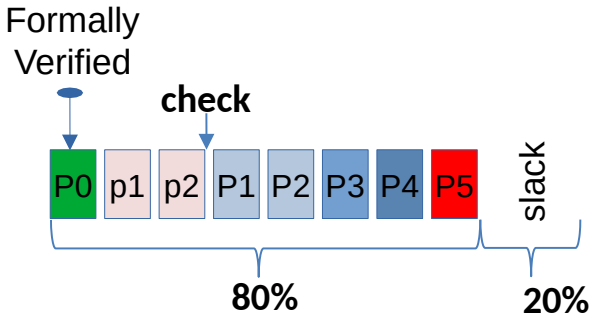


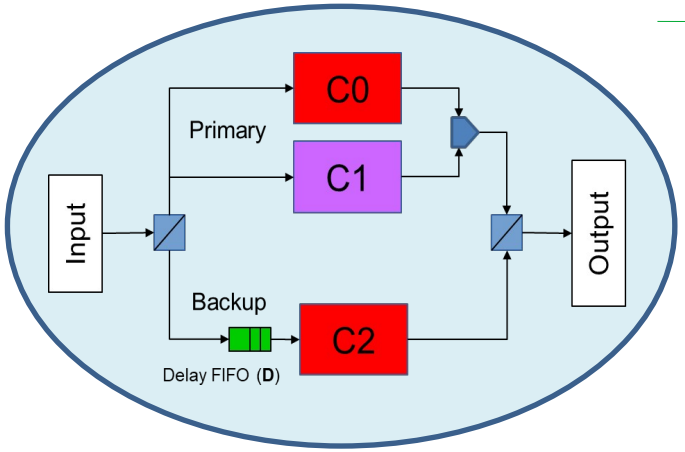
Many SubProcess BFT++ design options:

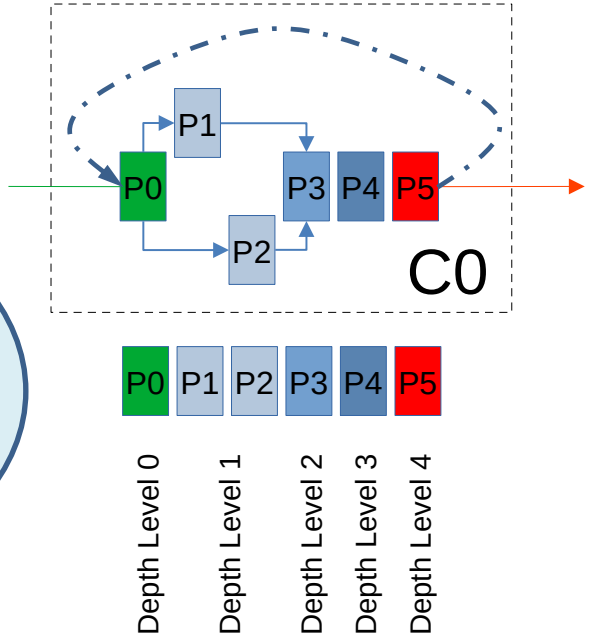




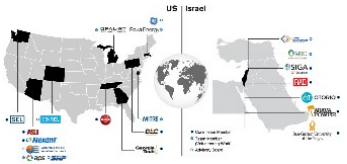








Cyber-Attack Resilience for CPS – Part A

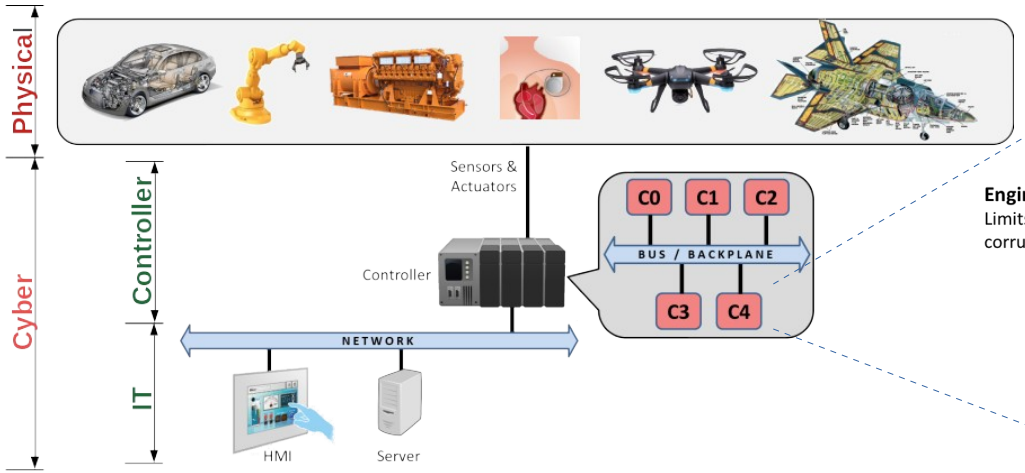


Goal:

- Continuous, uninterrupted operation under direct cyber-attack campaign
- Develop an economical and robust cyber attack resilience at controller level (level 0 & 1), relying on the physical properties of the controlled physical systems.

Past Methods:

- BFT++ is a new approach to resiliency, leveraging established Fault Tolerant systems.
- Proposed SubProcess BFT++ will reduce the deployment cost for BFT++ cyber-attack-resilience.

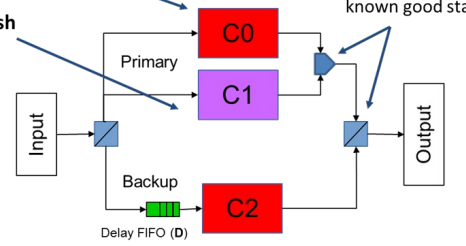


$$T_{crash} \leq D * T_{SC} \leq T_d - T_r$$

Artificial diversity
Protects controllers from being owned

Controller recovery
Detects variance and restores controller(s) to known good state

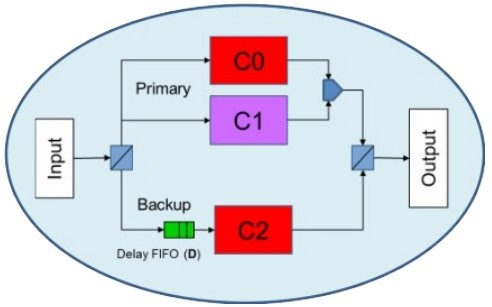
Engineered FastCrash
Limits time spent in corrupted state



Delayed input sharing
Keeps backup in clean state and isolates malicious trigger

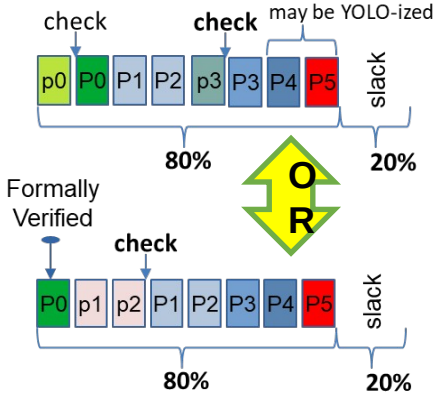
Successful attack requires:
1. Success on derailing targeted program → targeted program loses control
2. Success on capturing control → attacker controls program execution

Cyber-Attack Resilience for CPS – Part B



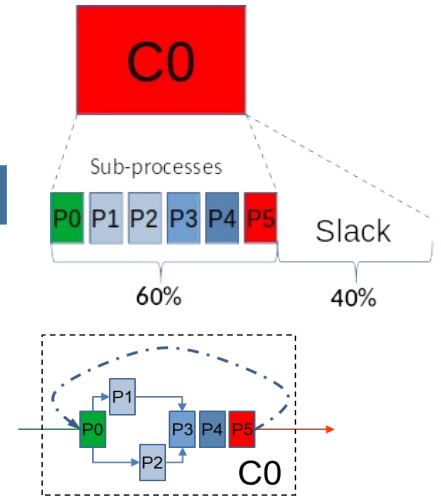
BFT++

Similar



SubProcess BFT++

SubProcess BFT++ Engineering **Tool**

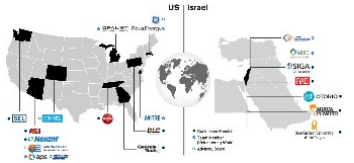


We plan to integrate the SubProcess BTF++ engineering tool into Schweitzer Engineering Laboratories (SEL) PLC design tools and environment.

Impact:

- Providing cyber attack resilience for application which cannot afford device redundancy, alleviate the need for redundant device in BFT++,
- Significantly widen the applicability of BFT++ and resilience against direct cyber-attack
- Automated isolation of offending data, can be communicated to other system components, e.g. SCATOPSY, RAM$^{2.}$, to prevent repeat attack.
- Integration into SEL design environment for ease of deployment and dissemination.

Schedule



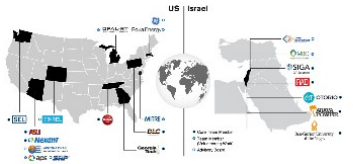
114	14.0 - Cyber-attack tolerance	05/01/22	10/30/24	
115	14.1 - Develop SubProcess BFT++ software architecture into SEL development environment integration plan	05/01/22	04/30/23	
116	*M14.1 - Integration plan developed*	04/30/23	04/30/23	
117	14.2 - Insert BFT++ software into SEL dev environment with all features	05/01/22	10/30/23	
118	*M14.2 - All features implemented, as prescribed*	10/30/23	10/30/23	
119	14.3 - Test FT++ software in SEL dev environment	05/01/22	04/30/24	
120	*M14.3 - Integration and validation passed*	04/30/24	04/30/24	
121	14.4 - Demonstrate a case of BFT++ implementation on SEL's PLC	05/01/22	10/30/24	
122	*M14.4 - Full capability demonstration*	10/30/24	10/30/24	

Status

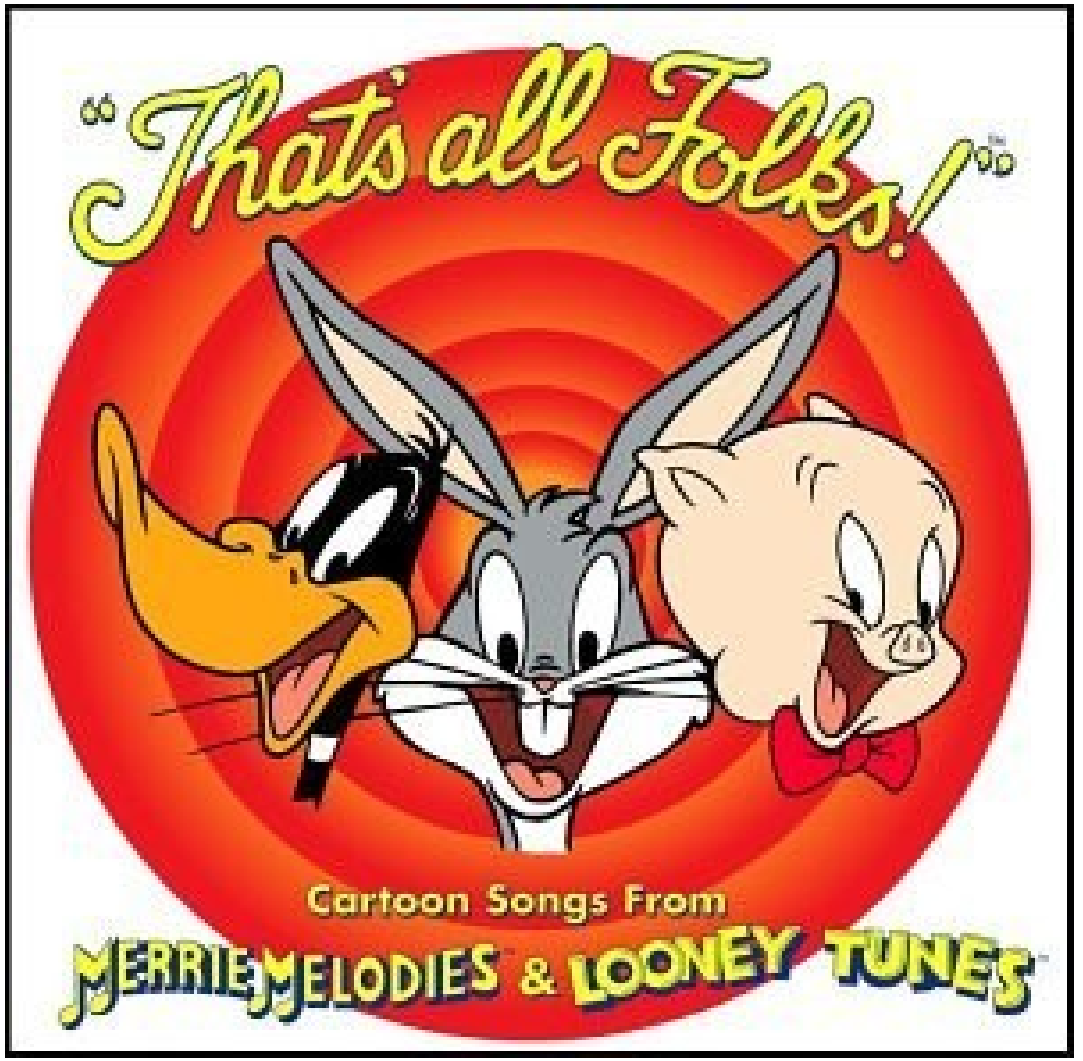
- Starting in May 2022
- Post Doc joining Jan. 24th
- Student starting in Fall semester

- Initial research will use an open source PLC environment: **ClassicLadder**.
  - For experimentation platform and
  - For analyzing generated codes for PLCs
    - Understanding scheduling structure
    - Studying design trade offs for integrating sub-process BFT++
- Future: integration into SEL design tools and environment
- Commercialization through integration into Vendor's development environment (SEL, and other vendors).

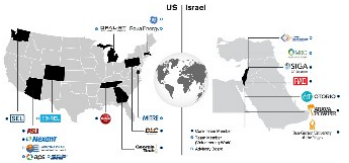
The End



Cyber Physical System



Physics Rules !!!


US | Israel