Deployment Architectures for Cyber-Physical Control Systems

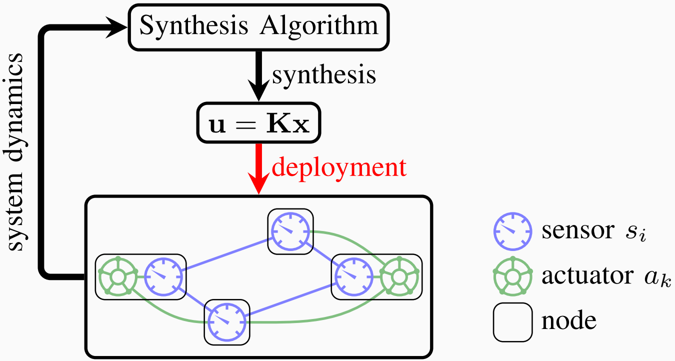
Shih-Hao Tseng[1], (pronounced as "She-How Zen")
joint work with James Anderson[2]

July 3, 2020

[1]Department of Computing and Mathematical Sciences, California Institute of Technology
[2]Department of Electrical Engineering, Columbia University

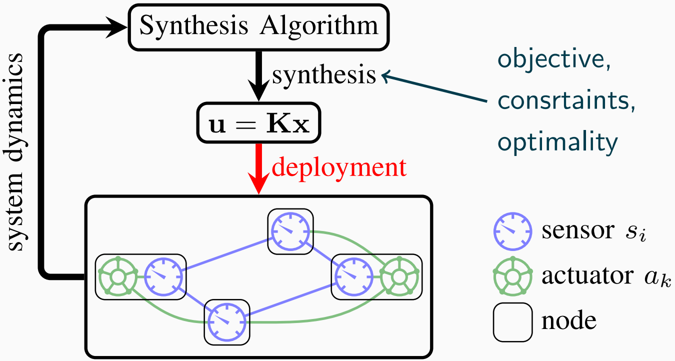
Controlling a Cyber-Physical System (CPS)

- A model-based approach to control design involves two phases: the *synthesis phase* and the *deployment phase*.



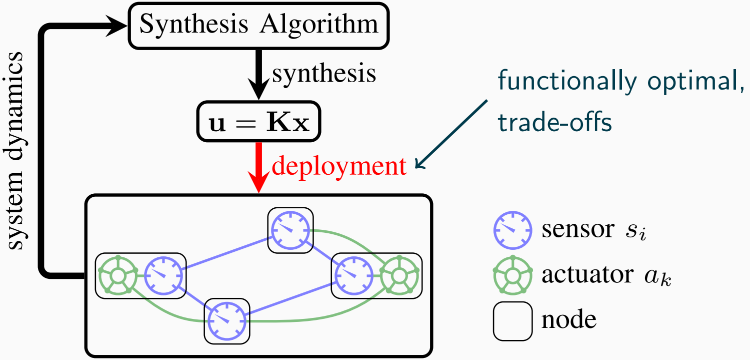
Controlling a Cyber-Physical System (CPS)

- A model-based approach to control design involves two phases: the *synthesis phase* and the *deployment phase*.



Controlling a Cyber-Physical System (CPS)

- A model-based approach to control design involves two phases: the *synthesis phase* and the *deployment phase*.



Approaches to Deployment

- Control community: top-down approach, from the plant dynamic/constraints to realizations (block diagrams/state space representations) or circuits.
  + systematic solution.
  - realizations: how to implement?
  - circuits: too specific.

Approaches to Deployment

- Control community: top-down approach, from the plant dynamic/constraints to realizations (block diagrams/state space representations) or circuits.
  - + systematic solution.
  - − realizations: how to implement?
  - − circuits: too specific.
- Networking/system community: bottom-up approach, adopting carefully designed gadgets/protocols and a coordination algorithm.
  - + know how each gadget/protocol works
  - − system-agnostic, too hardware-specific

Alternative Approach to Deployment

- Interfacing: To avoid binding the design to some specific hardware, we specify the basic components of the system.
- We express our design, the *architectures*, using those basic components.
- We can easily map our architectures to the real CPS – as long as the system supports all basic functions.

Terminology

- **Controller model:** a (linear) map from the state vector to the control action.
- **Realization:** a control block diagram/state space dynamics based on some controller model.
- **Architecture:** a cyber-physical system structure built from basic components.
- **Synthesize:** derive a controller model (and some realization).
- **Deploy/Implement:** map a controller model (through a realization) to an architecture.

System Model

- Sensors s_i, $i = 1, \ldots, N_x$ and actuators a_k, $k = 1, \ldots, N_u$.
- Plant dynamics:

$$x[t+1] = Ax[t] + Bu[t] + d_x[t]$$

  where $x[t] \in \mathbb{R}^{N_x}$ is the state vector, $u[t] \in \mathbb{R}^{N_u}$ is the control, and $d_x[t] \in \mathbb{R}^{N_x}$ is the disturbance.
- State-feedback: There is a sensor associated to every state.
- Suppose the system is open-loop stable, i.e., $(zI - A)^{-1}B \in \mathcal{RH}_\infty$.

System Level Synthesis (SLS)

- To synthesize a state-feedback closed-loop controller for the system, SLS introduces the *system response* $\{\mathbf{\Phi_x}, \mathbf{\Phi_u}\}$ transfer matrices such that

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{u} \end{bmatrix} = \begin{bmatrix} \mathbf{\Phi_x} \\ \mathbf{\Phi_u} \end{bmatrix} \mathbf{d_x},$$

where

$$\mathbf{\Phi_x} = (zI - A - B\mathbf{K})^{-1},$$
$$\mathbf{\Phi_u} = \mathbf{K}(zI - A - B\mathbf{K})^{-1}.$$

under the feedback policy $\mathbf{u} = \mathbf{K}\mathbf{x}$.

Wang et al., "A System Level Approach to Controller Synthesis," 2019.
Anderson et al., "System Level Synthesis," 2019.

System Level Synthesis (SLS)

- The *system level synthesis* problem takes the form:

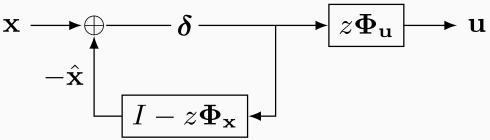
$$
\begin{aligned}
\min \quad & g(\boldsymbol{\Phi}_{\mathbf{x}}, \boldsymbol{\Phi}_{\mathbf{u}}) \\
\text{s.t.} \quad & \begin{bmatrix} zI - A & -B \end{bmatrix} \begin{bmatrix} \boldsymbol{\Phi}_{\mathbf{x}} \\ \boldsymbol{\Phi}_{\mathbf{u}} \end{bmatrix} = I \\
& \boldsymbol{\Phi}_{\mathbf{x}}, \boldsymbol{\Phi}_{\mathbf{u}} \in z^{-1}\mathcal{RH}_{\infty} \\
& \begin{bmatrix} \boldsymbol{\Phi}_{\mathbf{x}} \\ \boldsymbol{\Phi}_{\mathbf{u}} \end{bmatrix} \in \mathcal{S}.
\end{aligned}
$$

- Once the problem is solved, the optimal controller is given by

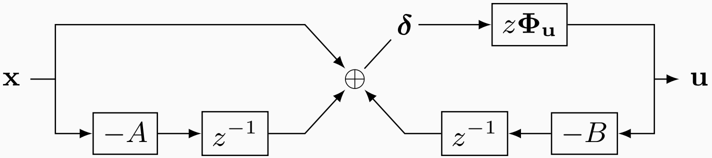
$$
\mathbf{K} = \boldsymbol{\Phi}_{\mathbf{u}} \boldsymbol{\Phi}_{\mathbf{x}}^{-1}
$$

Wang et al., "A System Level Approach to Controller Synthesis," 2019.
Anderson et al., "System Level Synthesis," 2019.

SLS Controller Realizations



(a) Standard SLS Realization:
$\mathbf{K} = (z\mathbf{\Phi_u})(z\mathbf{\Phi_x})^{-1}$.



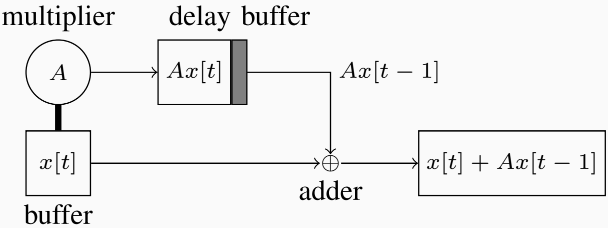
(b) New Realization for Open-Loop Stable Plant:
$\mathbf{K} = (z\mathbf{\Phi_u})(I + z^{-1}B(z\mathbf{\Phi_u}))^{-1}(I - z^{-1}A)$.

Theorem 1
Let $A \in \mathbb{R}^{N_x \times N_x}$ be Schur stable. The dynamic state-feedback controller $\mathbf{u} = \mathbf{K}\mathbf{x}$ realized via

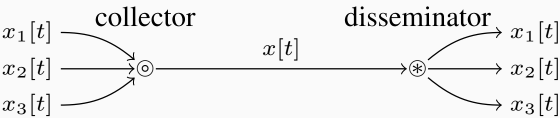
$$\delta[t] = x[t] - Ax[t-1] - Bu[t-1],$$
$$u[t] = \sum_{\tau \geq 1} \Phi_u[\tau]\delta[t+1-\tau],$$

is internally stabilizing and is described by the block diagram in (b).

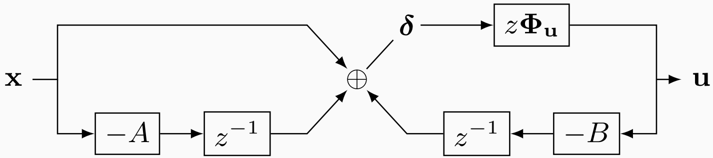
Basic Components

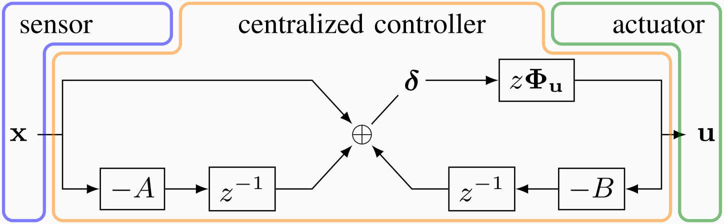


(a) Computation and Storage Components

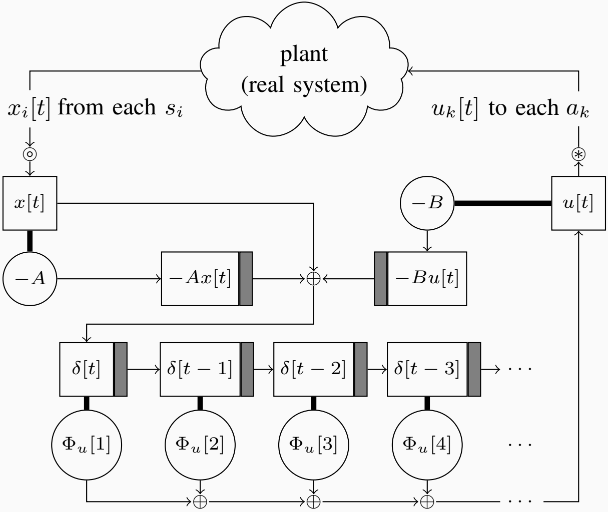


(b) Communication Components

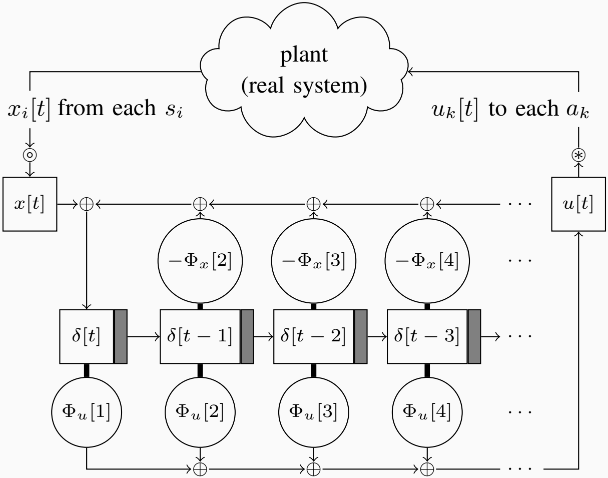

$\mathbf{x}$
$\boldsymbol{\delta}$
$z\mathbf{\Phi_u}$
$\mathbf{u}$
$\oplus$
$-A$
z^{-1}
z^{-1}
$-B$


sensor
centralized controller
actuator
$\mathbf{x}$
$\boldsymbol{\delta}$
$z\mathbf{\Phi_u}$
$\mathbf{u}$
$\oplus$
$-A$
z^{-1}
z^{-1}
$-B$

Centralized Architecture



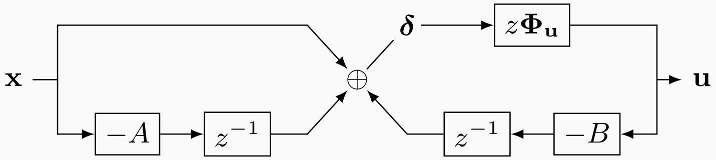
Centralized Architecture: Comparing with the Standard

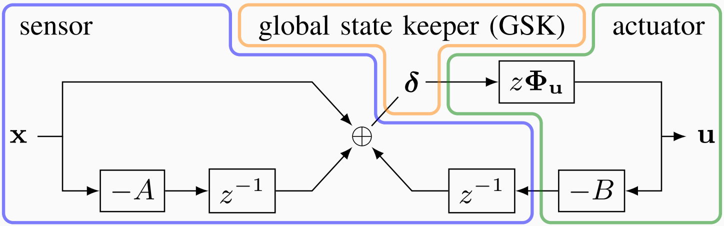


Centralized Architecture

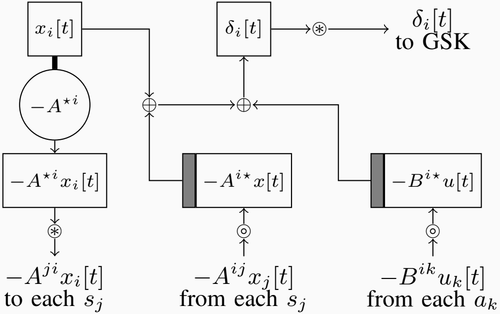
- When the horizon $T > 3$, $N_x \geq N_u$ (under-actuated), $N_x \geq 2$, the new architecture is "cheaper" than the standard one in terms of computation and storage.
- Single point of failure.
- Poor scalability: high communication and computational load on the centralized controller.

Global State Architecture



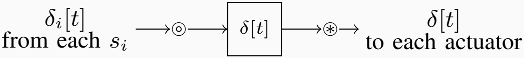


Global State Architecture

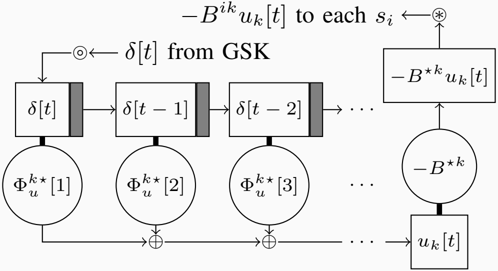


(a) Sensor s_i

Global State Architecture



(b) Global State Keeper (GSK)

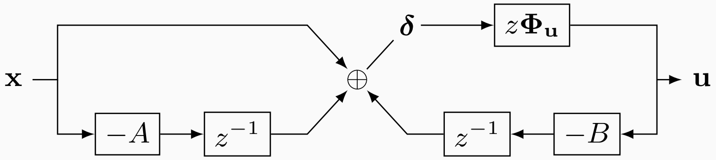


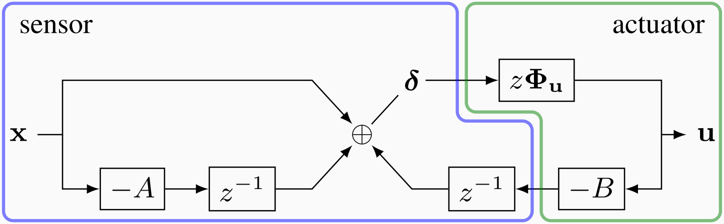
(c) Actuator a_k

Global State Architecture

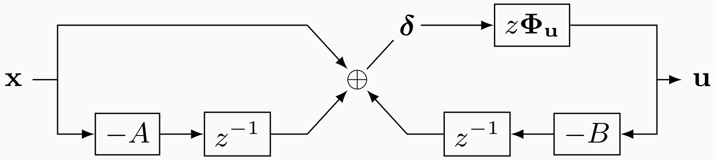
- Better scalability than the centralized architecture: lower computation workload at the GSK.
- The cyber pattern is similar to its physical structure.
- Single point of failure.

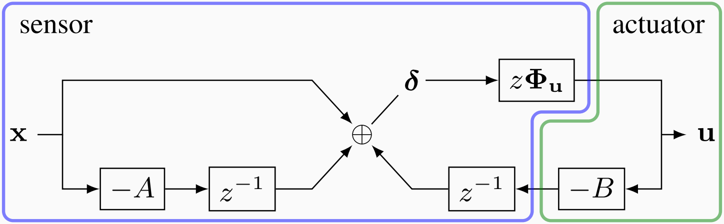
Naive Distributed Architecture



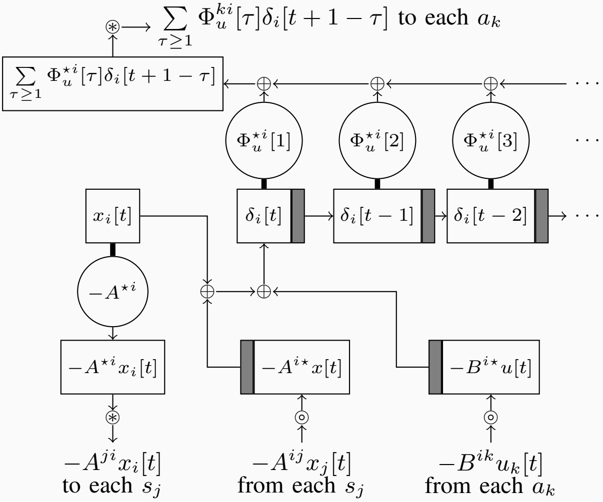


Memory Conservative Distributed Architecture



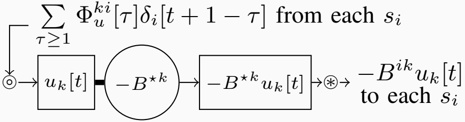


Memory Conservative Distributed Architecture



(a) Sensor s_i

Memory Conservative Distributed Architecture



(b) Actuator a_k

Distributed Architectures

- No single point of failure.
- The cyber pattern is similar to its physical structure, while having a lot more sensor-actuator communications.
- The sensor-actuator communications are reducible by localization $\rightarrow$ high scalability.

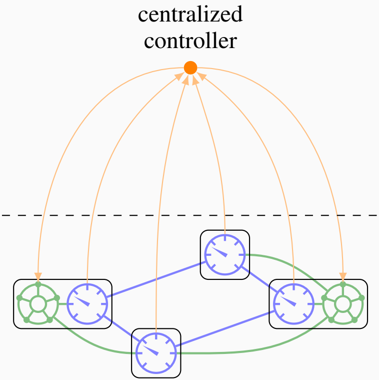
Comparison Amongst the Proposed Architectures

C: centralized, G: global state,

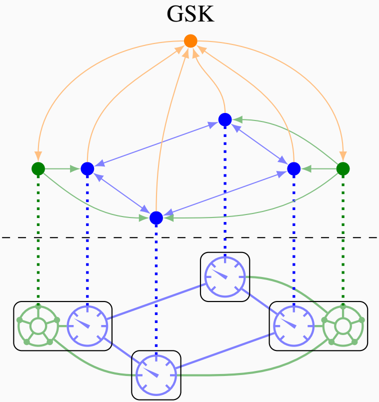
N: naive distributed, M: memory conservative distributed

- Single point of failure: C, G
- Overall memory usage: G $>$ N $>$ M $>$ C
- Single node memory usage: C $>$ G,N,M
- Single node computation loading: C $>$ G,N,M
- Single node communication loading: C,G $>$ N,M

Comparison of the Cyber-Physical Structures

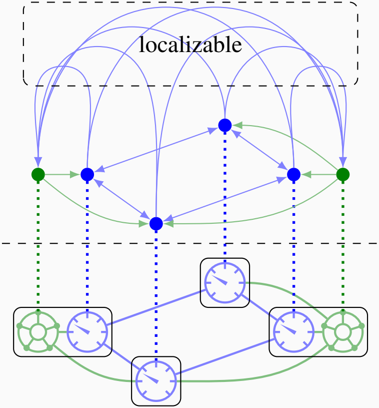


(a) Centralized Architecture

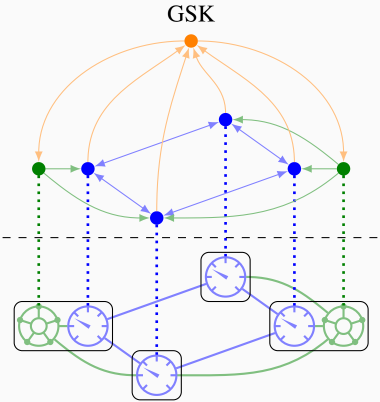


(b) Global State Architecture

Comparison of the Cyber-Physical Structures



(c) Distributed Architectures



(b) Global State Architecture

Conclusion and Future Directions

- We derived a new internally stabilizing SLS realization for open-loop stable plant.
- We proposed and compared four different deployment architectures for it: centralized, global state, naive distributed, and memory conservative distributed.
- Future directions:
    - removing open-loop stability requirement – using robust SLS, we can decompose $A = A_u + A_s$ where A_u is unstable and A_s is stable.
    - decentralized is not distributed: clustered architecture.