# Using Structured System Theory to Identify Malicious Behavior in Distributed Systems

**Shreyas Sundaram**
**Department of Electrical and Computer Engineering**
**University of Waterloo**

Collaborators

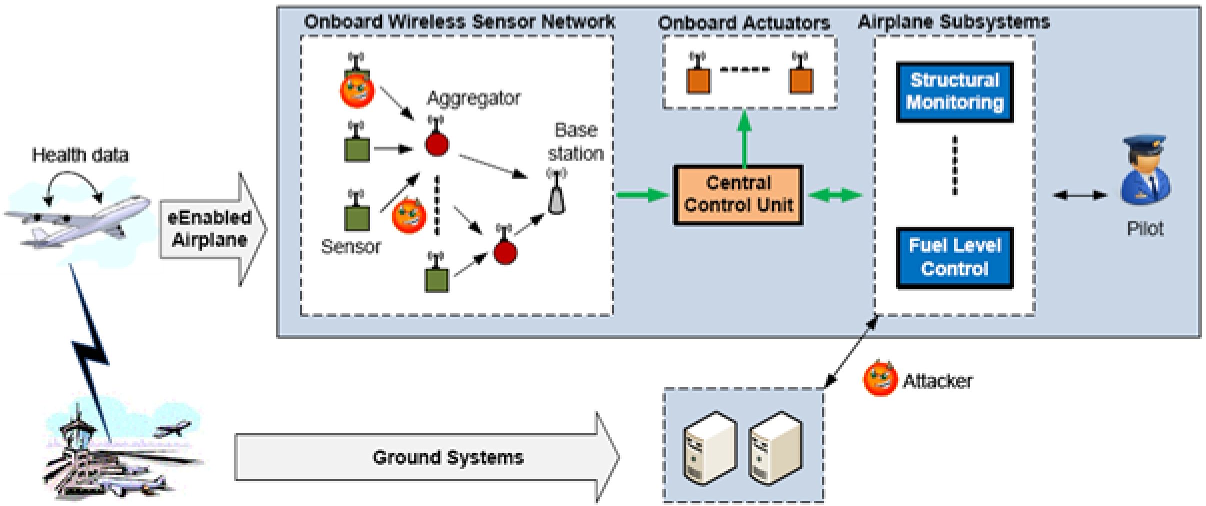
**Miroslav Pajic, Rahul Mangharam, George Pappas**
Electrical and Systems Engineering
University of Pennsylvania

**Christoforos Hadjicostis**
Electrical and Computer Engineering
University of Cyprus

# Outline

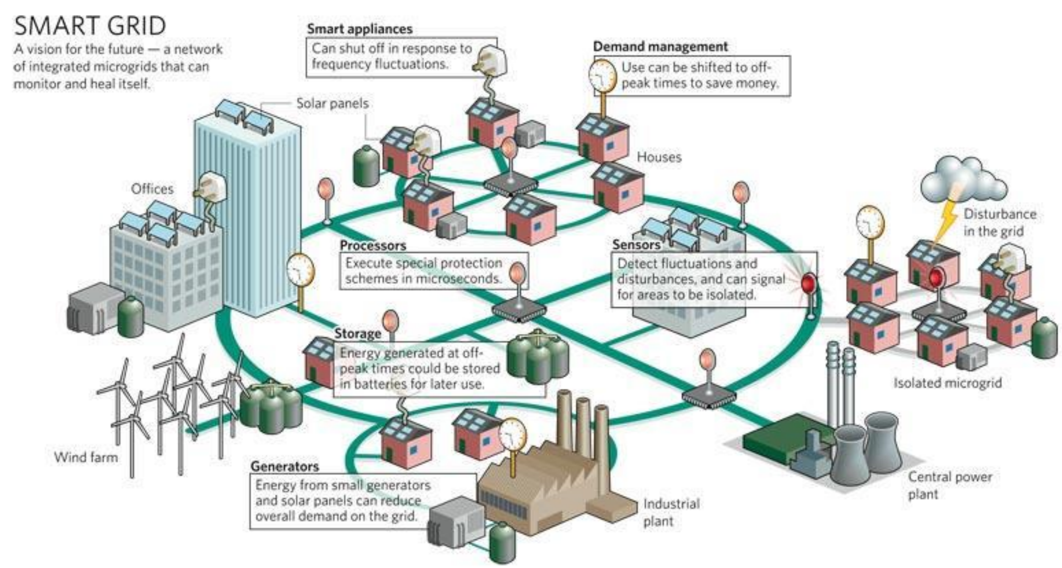
- Introduction
- Background on Linear and Structured System Theory
- Resilient Information Dissemination in Multi-Agent Networks
- Designing an Intrusion Detection System for a Wireless Control Network

# Distributed Systems and Networks in Safety-Critical Applications




**Airplane Health Monitoring and Management System**

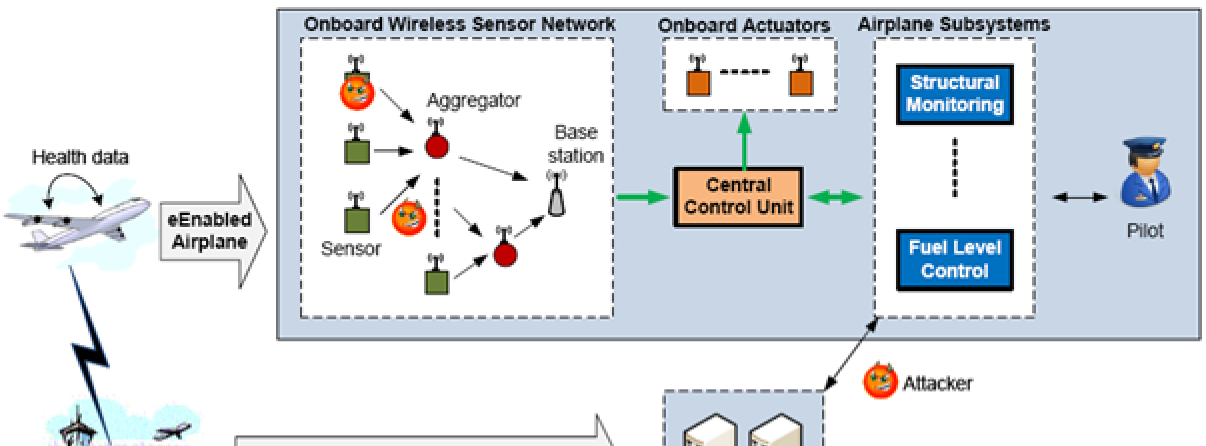
**Electrical Power Grid**




**UAV Teams**

Credits: Sampigethaya et al. (Digital Avionics Conference 2007), Urban Ecoist, Boeing

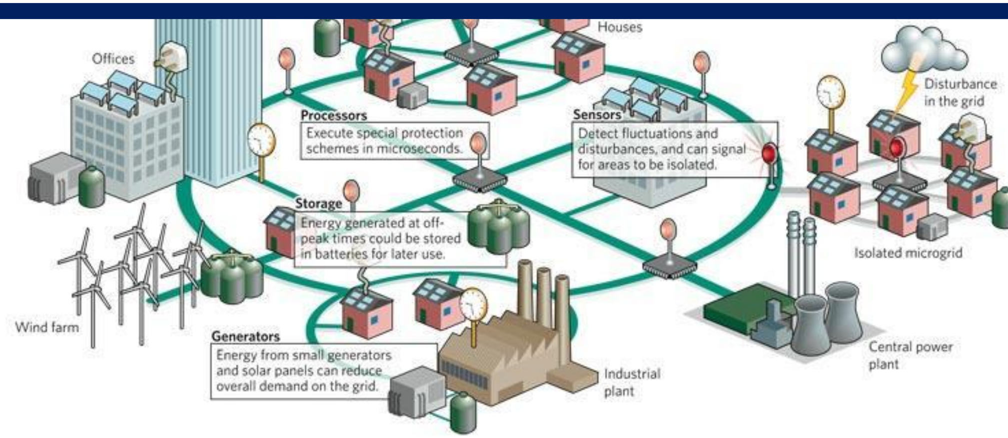
# Distributed Systems and Networks in Safety-Critical Applications




Airplane Health Monitoring and Management System

**Core requirement:** disseminate information through network

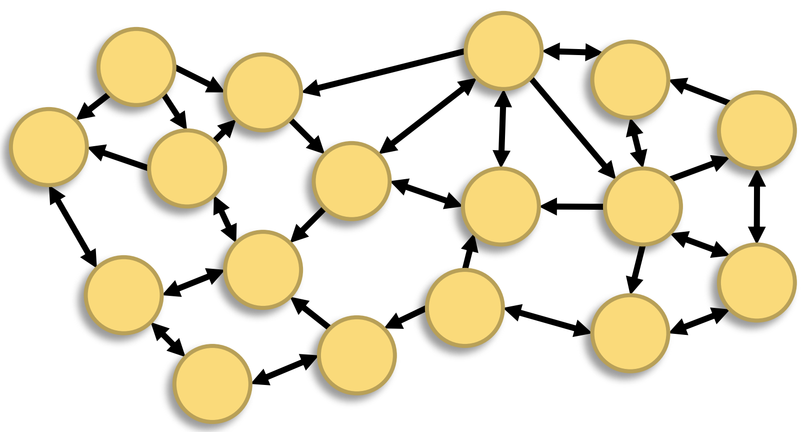
- quickly, efficiently, reliably, securely, …




Credits: Sampigethaya et al. (Digital Avionics Conference 2007), Urban Ecoist, Boeing

# Problem Formulation

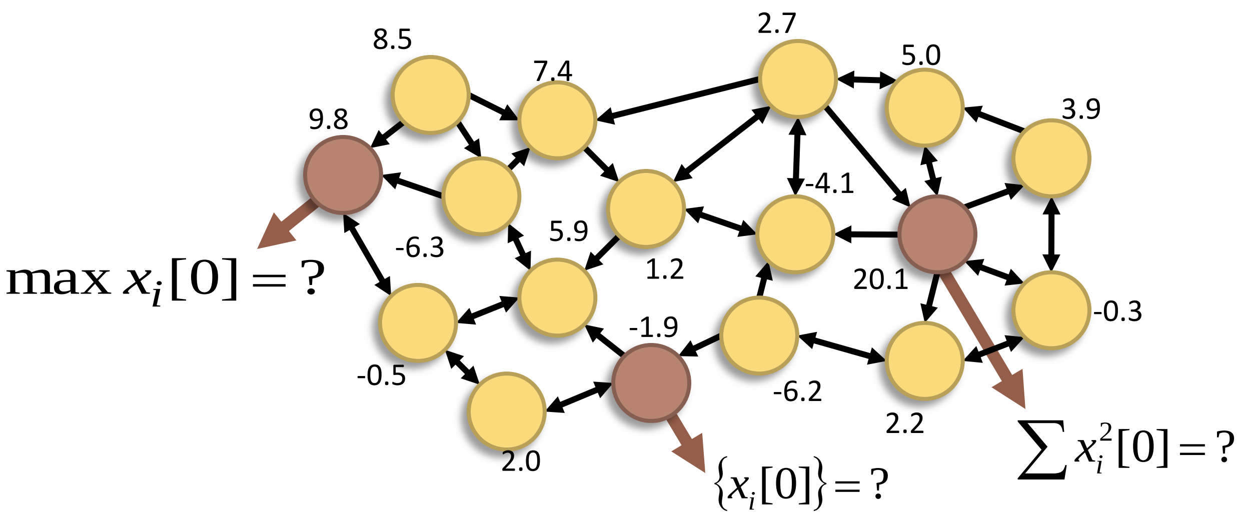
- Consider a network with nodes $\{x_1, x_2, \ldots, x_N\}$
  - e.g., sensors, computers, robots, "agents", …



- Each node can only interact with certain other nodes
  - Assume fixed interaction topology in this talk

# Function Calculation

- Each node $x_i$ has some initial value $x_i[0]$
  - e.g., temperature measurement, position, vote, ...
- Some nodes must **calculate certain functions** of these values
- Special cases
  - **Data accumulation**: some nodes gather all values
  - **Distributed consensus**: all nodes calculate same function



# Existing Work on Distributed Function Calculation

- Distributed function calculation schemes well studied over past few decades
  - Flooding, tree-based schemes, gossip, token passing, …
  - Communication complexity, computational complexity, time complexity, fault tolerance, …

- Many excellent books on this topic
  - **Dissemination of Information in Communication Networks**, Hromkovic et. al., 2005
  - **Communication Complexity**, Kushilevitz and Nisan, 1997
  - **Distributed Algorithms**, Lynch, 1997
  - **Elements of Distributed Computing**, Garg, 2002
  - **Parallel and Distributed Computation**, Bertsekas and Tsitsiklis, 1997
  - …

# Linear Iterative Strategies

- Investigate **linear iterative strategies** for distributed function calculation
- At each time-step k, every node updates its value as

$$x_i[k+1] = w_{ii} x_i[k] + \sum_{j \in nbr(i)} w_{ij} x_j[k]$$

- Extensively studied for **asymptotic consensus**
  - For some vector **a**,

$$\lim_{k \to \infty} x_i[k] = \mathbf{a}^T \mathbf{x}[0], \quad \forall i \in \{1, 2, \ldots, N\}$$

- Also studied by communications community: **network coding**
- Survey papers:
  - **[Olfati-Saber, Fax & Murray**, *Proc. IEEE*, 2007**], [Ren, Beard & Atkins**, *Proc. ACC*, 2005], [**Yeung, Li, Cai, Zhang**, 2006]

# Potential for Incorrect Behavior

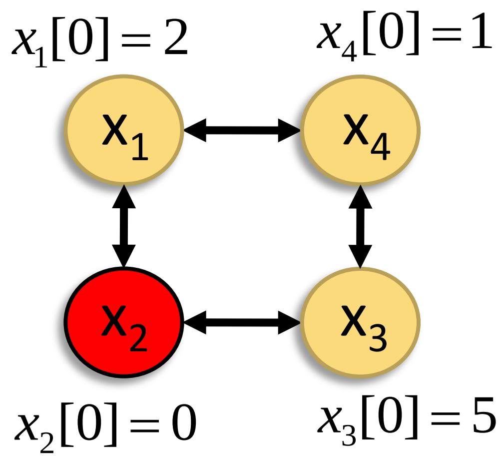
- Here, we ask:

"What happens if some nodes don't follow the linear iterative strategy?"

  - **Faulty nodes:** update their values incorrectly due to hardware faults, or stop working altogether
  - **Malicious nodes:** willfully update their values incorrectly to prevent other nodes from calculating functions

- Related works:
  - [Jadbabaie, Lin & Morse '03], [Gupta, Langbort & Murray '06], [Pasqualetti, Bicchi & Bullo '07], [Sundaram & Hadjicostis '08], [Teixeira, Sandberg & Johansson '10], [Chapman & Mesbahi '10]

# Building Intuition




- Node $x_1$ wants to obtain all initial values via some algorithm
- Node $x_2$ is malicious and pretends $x_3[0] = 9$
- Node $x_4$ behaves correctly and uses $x_3[0] = 5$
- Node $x_1$ doesn't know who to believe
  - i.e., is node $x_3$'s value equal to 9 or 5?
- Node $x_1$ needs another node to act as tie-breaker

# Main Results

- If network has up to **b malicious nodes**, we show:

| | | |
|---|---|---|
| Node $x_i$ has **2b or fewer** node-disjoint paths from some node $x_j$ | ⇒ | Malicious nodes can update their values in such a way that $x_i$ **cannot calculate any function** of $x_j$'s initial value |
| Node $x_i$ has **2b+1 or more** node-disjoint paths from every other node | ⇒ | $x_i$ can obtain **all initial values** after running linear strategy for **at most N time-steps** with **almost any weights** |

# Main Results

- If network has up to **b malicious nodes**, we show:

| | | |
|---|---|---|
| Node $x_i$ has **2b or fewer** node-disjoint paths from some node $x_j$ | → "Easy" | Malicious nodes can update their values in such a way that $x_i$ **cannot calculate any function** of $x_j$'s initial value |
| Node $x_i$ has **2b+1 or more** node-disjoint paths from every other node | → "Tricky" | $x_i$ can obtain **all initial values** after running linear strategy for **at most N time-steps** with **almost any weights** |

# Using Structured System Theory to Analyze Linear Iterations

- To prove resilience, we use the following approach:

Linear strategy is resilient to b malicious nodes in a given network

A linear system has some property "**P**"

(Structured System Theory)

Original network has connectivity 2b+1

A graph has some property "**Q**"

# Background on
# Linear and Structured
# System Theory

# Properties of Linear Systems

$$x[k+1] = Ax[k] + Bu[k]$$
$$y[k] = Cx[k]$$

State: $x \in \Re^n$,
Output: $y \in \Re^p$,
Input: $u \in \Re^m$

- **Controllability:** drive state to desired value using input
- **Observability:** determine state from output, with **input known** (or zero)
- **Strong Observability:** determine state from output, with **input unknown**
- **Invertibility:** determine input from output, with **state known**

# Properties of Linear Systems

$$x[k+1] = Ax[k] + Bu[k]$$
$$y[k] = Cx[k]$$

State: $x \in \Re^n$,
Output: $y \in \Re^p$,
Input: $u \in \Re^m$

- **Controllability:** drive state to desired value using input
- **Observability:** determine state from output, with **input known** (or zero)
- **Strong Observability:** determine state from output, with **input unknown**
- **Invertibility:** determine input from output, with **state known**

# Properties of Linear Systems

$$x[k+1] = Ax[k] + Bu[k]$$
$$y[k] = Cx[k]$$

State: $x \in \Re^n$,
Output: $y \in \Re^p$,
Input: $u \in \Re^m$

- **Controllability:** drive state to desired value using input

**Standard Approach:**
**Use algebraic tests to determine if properties hold**

- **Strong Observability:** determine state from output, with **input unknown**

- **Invertibility:** determine input from output, with **state known**

# Linear Structured Systems

$$x[k+1] = Ax[k] + Bu[k] \qquad x \in \Re^n, y \in \Re^p, u \in \Re^m$$
$$y[k] = Cx[k]$$

- System is **structured** if every entry of the matrices (A,B,C) is either zero, or an independent free parameter

- Used to represent and analyze dynamical systems with unknown/uncertain parameters [Lin '74, Dion et al., '03]

- **Structured system theory:** determines properties of systems based on the zero/nonzero structure of matrices

# Structural Properties

**"Structured system has property P"**:

Property P holds for **at least one choice** of free parameters in the matrices (A, B, C)

- Structural properties are **generic**!

| |
|---|
| Structured system has property P $\Rightarrow$ Structured system will have property P for **almost any choice** of free parameters |

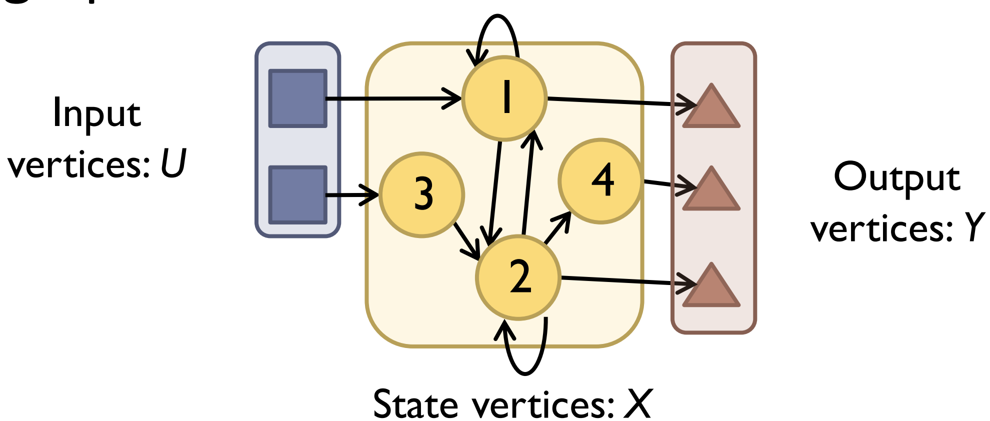
- Use graph based techniques to determine if structural properties hold

# Example of Structured System and Associated Graph

- Structured system can be represented as a graph
- Structured system:

$$x[k+1]=\begin{bmatrix}\lambda_1 & \lambda_2 & 0 & 0\\ \lambda_3 & \lambda_4 & \lambda_5 & 0\\ 0 & 0 & 0 & 0\\ 0 & \lambda_6 & 0 & 0\end{bmatrix}x[k]+\begin{bmatrix}\lambda_7 & 0\\ 0 & 0\\ 0 & \lambda_8\\ 0 & 0\end{bmatrix}u[k],\quad y[k]=\begin{bmatrix}\lambda_9 & 0 & 0 & 0\\ 0 & \lambda_{10} & 0 & 0\\ 0 & 0 & 0 & \lambda_{11}\end{bmatrix}x[k]$$

- Associated graph ***H***:

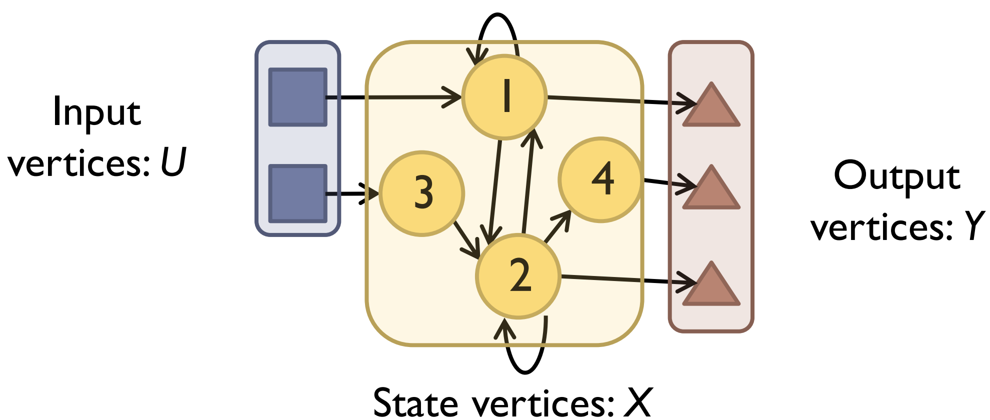


# Example: Test for Structural Invertibility

**Theorem** [*van der Woude, '91*]:

| System is structurally invertible | ⟺ | Graph H has **m vertex-disjoint paths** from inputs to outputs |
|---|---|---|

- e.g.,





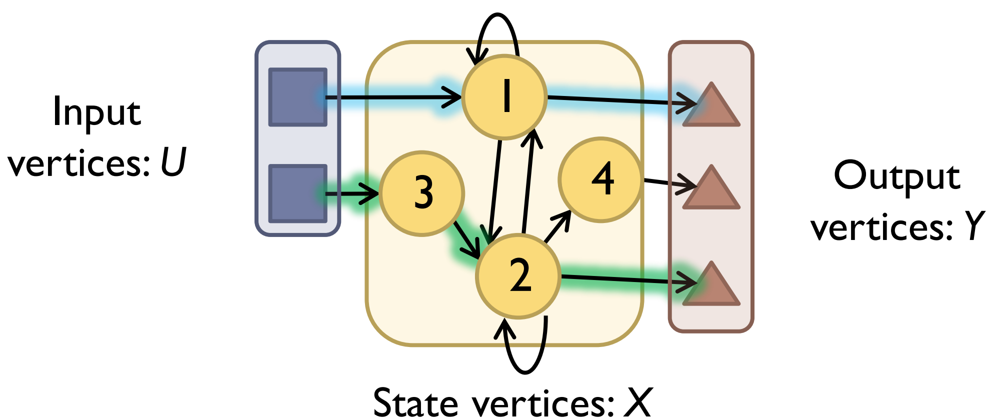
J. W. van der Woude, Mathematics of Control Systems and Signals, 1991

# Example: Test for Structural Invertibility

**Theorem** [*van der Woude, '91*]:

| System is structurally invertible | ⟺ | Graph H has **m vertex-disjoint paths** from inputs to outputs |
|---|---|---|

- e.g.,




Structurally Invertible


J. W. van der Woude, Mathematics of Control Systems and Signals, 1991

# References on Structured Systems

- C. T. Lin, "*Structural Controllability*", IEEE TAC, 1974
- K. J. Reinschke, *Multivariable Control: A Graph-Theoretic Approach*, 1988
- J-M. Dion, C. Commault and J. van der Woude, "*Generic Properties and Control of Linear Structured Systems: A Survey*", Automatica, 2003
- D. D. Siljak, *Decentralized Control of Complex Systems*, 1991
- Sundaram & Hadjicostis, CDC 2009, ACC 2010 (*Structural properties over finite fields*, *upper bound on generic controllability/observability indices*)

# Application to Resilient Information Dissemination

# Modeling Faulty/Malicious Behavior in Linear Iterative Strategies

- Linear iterative strategy for information dissemination:
  - **Correct** update equation for node $x_i$:

$$x_i[k+1] = w_{ii}x_i[k] + \sum_{j \in nbr(i)} w_{ij}x_j[k]$$

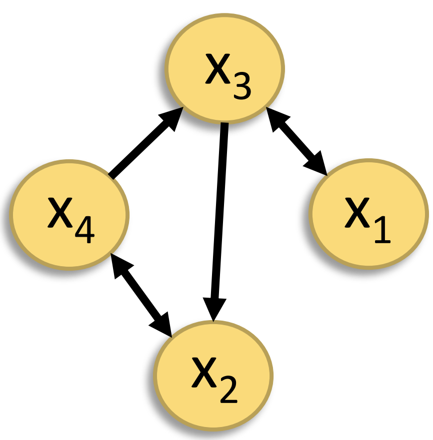
  - **Faulty or malicious** update by node $x_i$:

$$x_i[k+1] = w_{ii}x_i[k] + \sum_{j \in nbr(i)} w_{ij}x_j[k] + f_i[k]$$

  - $f_i[k]$ is an additive error at time-step k
  - **Note:** this model allows node $x_i$ to update its value in a **completely arbitrary** manner

# Modeling the Values Seen by Each Node

- Each node obtains neighbors' values at each time-step
- Let $\mathbf{y}_i[k] = \mathbf{C}_i\mathbf{x}[k]$ denote values seen by $x_i$ at time-step k
  - Rows of $\mathbf{C}_i$ index portions of $\mathbf{x}[k]$ available to $x_i$




For node $x_3$:

$$\mathbf{y}_3[k] = \begin{bmatrix} x_1[k] \\ x_3[k] \\ x_4[k] \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{C}_3} \mathbf{x}[k]$$

# Linear Iteration with Faulty/Malicious Nodes

- Let $S = \{x_{i_1}, x_{i_2}, \ldots, x_{i_b}\}$ be set of faulty/malicious nodes
  - **Unknown** a priori, but bounded by b

- Update equation for entire **system**:

$$\underbrace{\begin{bmatrix} x_1[k+1] \\ \vdots \\ x_N[k+1] \end{bmatrix}}_{\mathbf{x}[k+1]} = \underbrace{\begin{bmatrix} w_{11} & \cdots & w_{1N} \\ \vdots & \ddots & \vdots \\ w_{N1} & \cdots & w_{NN} \end{bmatrix}}_{\mathbf{W}} \underbrace{\begin{bmatrix} x_1[k] \\ \vdots \\ x_N[k] \end{bmatrix}}_{\mathbf{x}[k]} + \underbrace{\begin{bmatrix} \mathbf{e}_{i_1} & \mathbf{e}_{i_2} & \cdots & \mathbf{e}_{i_b} \end{bmatrix}}_{\mathbf{B}_S} \underbrace{\begin{bmatrix} f_{i_1}[k] \\ f_{i_2}[k] \\ \vdots \\ f_{i_b}[k] \end{bmatrix}}_{\mathbf{f}_S[k]}$$

$$\mathbf{y}_i[k] = \mathbf{C}_i \mathbf{x}[k]$$

**Constraint:** weight $w_{ij} = 0$ if node $x_j$ is not a neighbor of node $x_i$

# Recovering the Initial State

- System model for linear iteration with malicious nodes

$$\mathbf{x}[k+1] = \mathbf{W}\mathbf{x}[k] + \mathbf{B}_S \mathbf{f}_S[k]$$
$$\mathbf{y}_i[k] = \mathbf{C}_i \mathbf{x}[k]$$

- **Objective:** Recover initial state $\mathbf{x}[0]$ from outputs of the system, without knowing $\mathbf{f}_S[k]$
- *Almost* equivalent to **strong observability** of system
  - The set S is also unknown here
  - Only know it has at most b elements

# Recovering the Initial State

- Want to ensure that the output trajectory uniquely specifies the initial state
  - Same output trajectory must not be generated by two different initial states and two (possibly) different sets of b malicious nodes
- By linearity, can show:

Can recover initial state in system

$$\mathbf{x}[k+1] = \mathbf{W}\mathbf{x}[k] + \mathbf{B}_S\mathbf{f}_S[k]$$
$$\mathbf{y}_i[k] = \mathbf{C}_i\mathbf{x}[k]$$

for **any unknown set** S of **b** nodes

$\Leftrightarrow$

Linear system

$$\mathbf{x}[k+1] = \mathbf{W}\mathbf{x}[k] + \mathbf{B}_Q\mathbf{f}_Q[k]$$
$$\mathbf{y}_i[k] = \mathbf{C}_i\mathbf{x}[k]$$

is strongly observable for **any known set** Q of **2b** nodes

# Structural Strong Observability

- For any set Q of 2b nodes, strong observability of

$$\mathbf{x}[k+1] = \mathbf{W}\mathbf{x}[k] + \mathbf{B}_Q \mathbf{f}_Q[k]$$
$$\mathbf{y}_i[k] = \mathbf{C}_i \mathbf{x}[k]$$

  is a **structural property**

- Graph of system is given by graph of original network, with additional inputs and outputs

- Using tests for structural strong observability, we show:

| | | |
|---|---|---|
| $x_i$ has **2b+1 node-disjoint paths** from every other node | ⇒ | Above system will be strongly observable for any set Q of 2b nodes |

# Robustness of the Linear Iterative Scheme

- By generic nature of structural properties:

| Network is **2b+1 connected** | ⇒ | For **almost any W**, any node can recover all initial values despite actions of b malicious nodes |
|---|---|---|

- How long will each node need to wait before the values that it receives uniquely specifies the initial state?

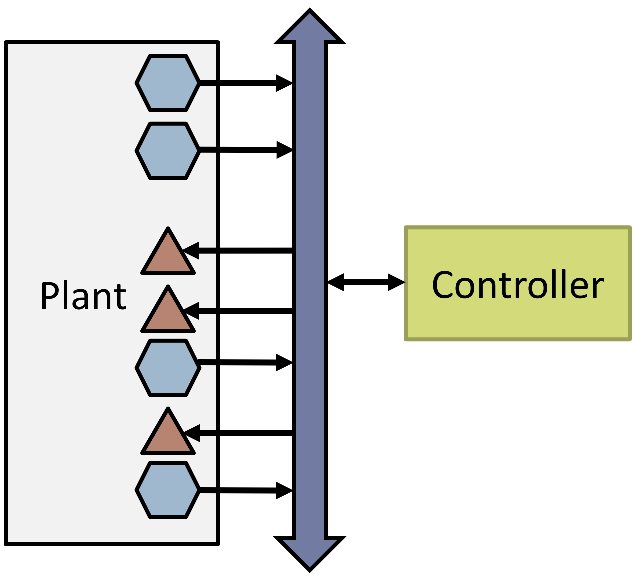
> If a linear system is strongly observable, outputs of system over N time-steps are sufficient to determine initial state

- Any node can obtain **x**[0] after **at most N time-steps**

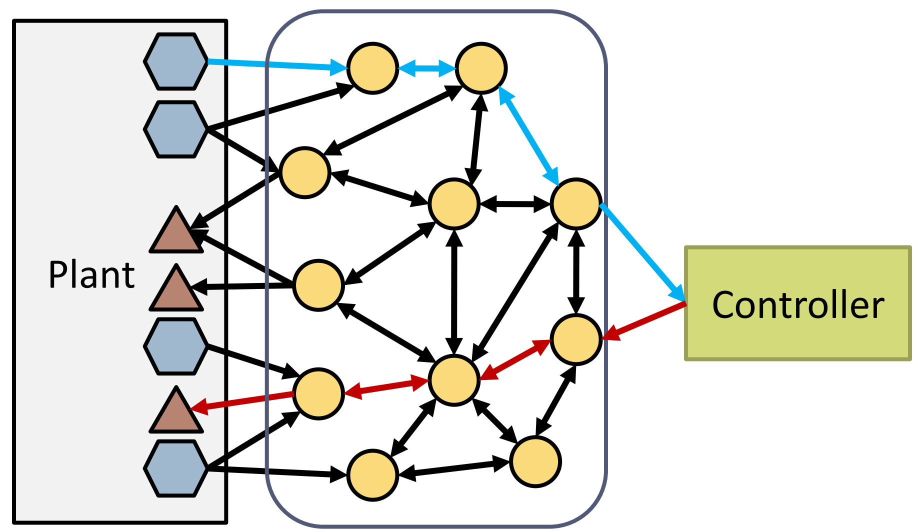
# Application to Feedback Control: Monitoring for Malicious Behavior in a Wireless Control Network

# Control Over Networks

- Sensors ( )and Actuators ( ) are installed on a *plant*
- Communicate with controllers ( ) over a network
- Standard architectures:




Wired Control System




Wireless Control System

**Network primarily used as a *channel***

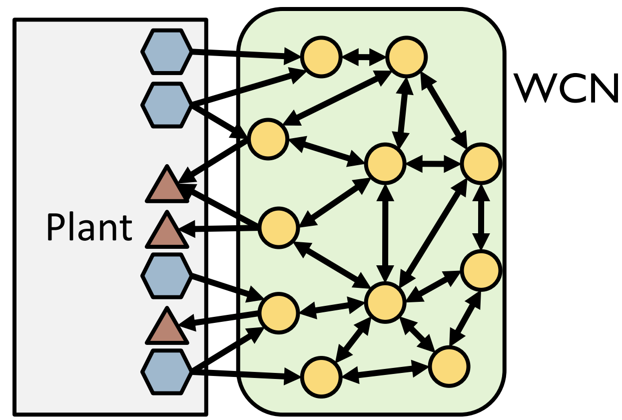
# Much Work on Networked Control

- Studied by many researchers
  - Branicky et al., Elia et al., Gupta et al., Hespanha et al., Martins et al., Murray et al., Nair et al., Seiler et al., Sinopoli et al., Walsh et al., …
- Robustness to "network" issues:
  - Packet dropouts, Bernoulli channel failures
  - Delays in the feedback loop
- Survey paper: *Hespanha et al., Proc IEEE, 2007*

# The Wireless Control Network (WCN)

- Idea: Can we use the wireless network ***itself*** as a controller?
  - Leverage the computational capability *in the network*

- Desired properties:
  - ✓ Lightweight computation - design for resource constrained nodes
  - ✓ Static transmission schedules
  - ✓ Compositionality
  - ✓ Handles multiple actuation and sensing points




- Utility: Network can be used as a backup control mechanism in case primary controller fails
  - Uses existing infrastructure, *does not require additional hardware*

# Wireless Control Network

- Plant:

$$\mathbf{x}[k+1] = \mathbf{A}\mathbf{x}[k] + \mathbf{B}\mathbf{u}[k]$$
$$\mathbf{y}[k] = \mathbf{C}\mathbf{x}[k]$$

- State update procedure, node $z_i$:

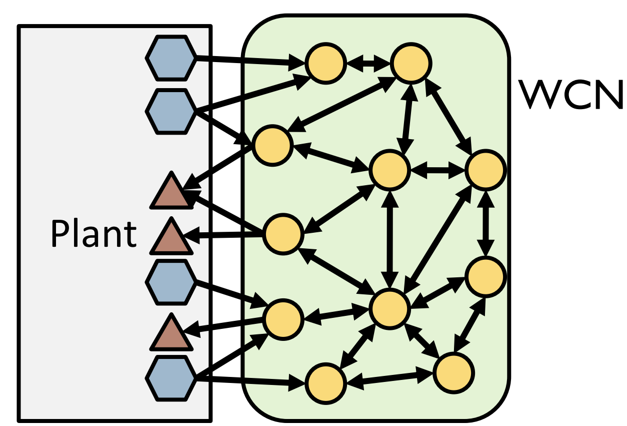
$$z_i[k+1] = w_{ii} z_i[k] + \sum_{z_j \in \mathrm{N}_{z_i}} w_{ij} z_j[k] + \sum_{s_j \in \mathrm{N}_{z_i}} h_{ij} y_j[k]$$

From neighbors From sensors

- Plant update procedure, input $i$:

$$u_i[k] = \sum_{z_j \in \mathrm{N}_{a_i}} g_{ij} z_j[k]$$

From actuator's neighbors




- Network state vector:

$$\mathbf{z}[k] = \begin{bmatrix} z_1[k] & z_2[k] & \cdots & z_N[k] \end{bmatrix}^T$$

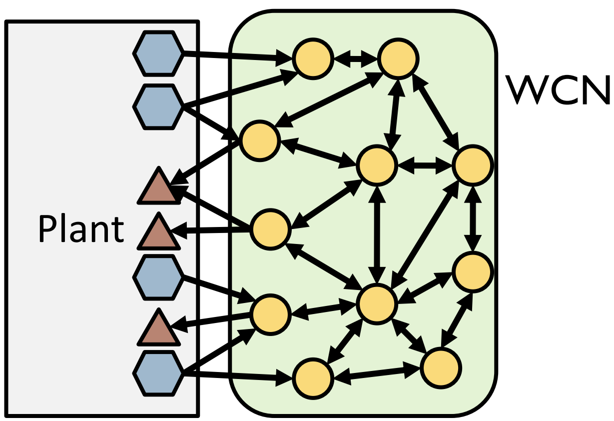
# Wireless Control Network: A Linear System

- Network update procedure:

$$\mathbf{z}[k+1] = \underbrace{\begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1N} \\ w_{21} & w_{22} & \cdots & w_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ w_{N1} & w_{N2} & \cdots & w_{NN} \end{bmatrix}}_{\mathbf{W}} \mathbf{z}[k] + \underbrace{\begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1p} \\ h_{21} & h_{22} & \cdots & h_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ h_{N1} & h_{N2} & \cdots & h_{Np} \end{bmatrix}}_{\mathbf{H}} \mathbf{y}[k]$$

$$\mathbf{u}[k] = \underbrace{\begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1N} \\ g_{21} & g_{22} & \cdots & g_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ g_{m1} & g_{m2} & \cdots & g_{mN} \end{bmatrix}}_{\mathbf{G}} \mathbf{z}[k]$$

**Only elements corresponding to existing links (link weights) are allowed to be non-zero**



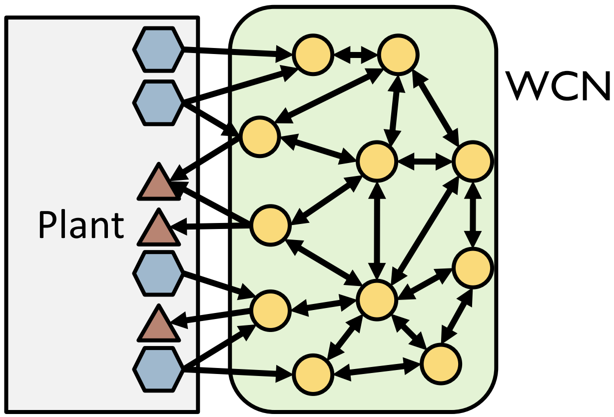
# Closed Loop System

- Overall system state: $\hat{\mathbf{x}}[k] = \begin{bmatrix} \mathbf{x}[k] \\ \mathbf{z}[k] \end{bmatrix}$

- Closed loop system:

$$\hat{\mathbf{x}}[k+1] = \begin{bmatrix} \mathbf{x}[k+1] \\ \mathbf{z}[k+1] \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{A} & \mathbf{BG} \\ \mathbf{HC} & \mathbf{W} \end{bmatrix}}_{\hat{\mathbf{A}}} \begin{bmatrix} \mathbf{x}[k] \\ \mathbf{z}[k] \end{bmatrix} = \hat{\mathbf{A}}\hat{\mathbf{x}}[k]$$

- Matrices **W**, **G**, **H** are **structured**
  - Sparsity constraints imposed by the WCN topology
  - A decentralized control problem



# Stabilizing the Closed Loop System

- **Problem:** Find numerical matrices **W**, **H**, **G** satisfying structural constraints so that

$$\hat{\mathbf{A}} = \begin{bmatrix} \mathbf{A} & \mathbf{BG} \\ \mathbf{HC} & \mathbf{W} \end{bmatrix} \text{ is stable}$$

- Use a numerical design procedure to determine suitable **W**, **H**, **G**
  - Based on Linear Matrix Inequalities
  - Can also be made robust to Bernoulli packet drops

# Abnormal Behavior in the Network

- What if certain nodes in the WCN become faulty or malicious?
- Security of control networks in industrial control systems is a major issue [*NIST Technical Report, 2008*]
  - **Data Historian:** Maintain and analyze logs of plant and network behavior
  - **Intrusion Detection System:** Detect and identify any abnormal activities
- Is it possible to design a Data Historian/Intrusion Detection System for the WCN?

# Intrusion Detection System

- **Obvious Solution (?):** have IDS listen to transmissions of ***every node*** in WCN and double-check whether they are computing correctly
  - Obviously not satisfactory

- Can we get by with listening to the transmissions of only a **subset** T of the nodes?
  - Under what conditions?
  - Which subset should we choose?

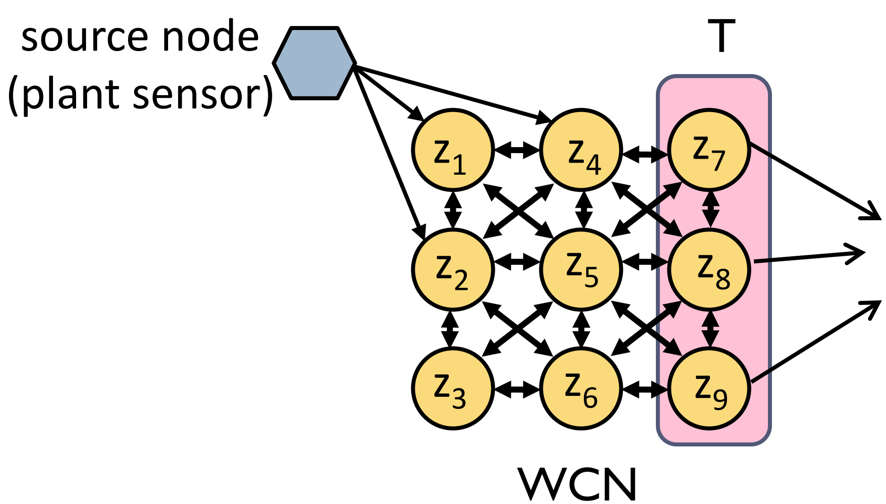
# Monitored Nodes

- Denote transmissions of any set T of monitored nodes by

$$\mathbf{t}[k] = \mathbf{T}\mathbf{z}[k]$$

  - **T** is a matrix with a single 1 in each row, indicating which elements of **z**[k] are being monitored

- Example:




$$\mathbf{t}[k] = \underbrace{\begin{bmatrix} \mathbf{0}_{3x6} & \mathbf{I}_3 \end{bmatrix}}_{\mathbf{T}} \mathbf{z}[k]$$

# System Model with Malicious Nodes

- As before, linear system model for WCN with set S of malicious nodes:

$$\mathbf{z}[k+1] = \mathbf{W}\mathbf{z}[k] + \mathbf{H}\mathbf{y}[k] + \mathbf{B}_S\mathbf{f}_S[k]$$
$$\mathbf{t}[k] = \mathbf{T}\mathbf{z}[k]$$

- **Objective:** Recover **y**[k], $\mathbf{f}_S$[k] and S (initial state **z**[0] assumed known)
  - *Almost* equivalent to **invertibility** of system
  - Problem: Don't know the set of faulty nodes S
    - Assumption: At most b faulty nodes
  - Must ensure that output sequence cannot be generated by a different y[k] and possibly different set of b malicious nodes

# Conditions for IDS/Data Historian Design

IDS can recover **y**[k] and identify up to b faulty nodes in the network by monitoring transmissions of set T

⟺

Can recover inputs and set S in system

$$\mathbf{z}[k+1] = \mathbf{W}\mathbf{z}[k] + \begin{bmatrix} \mathbf{H} & \mathbf{B}_S \end{bmatrix} \begin{bmatrix} \mathbf{y}[k] \\ \mathbf{f}_S[k] \end{bmatrix}$$

$$\mathbf{t}[k] = \mathbf{T}\mathbf{z}[k]$$

for **any unknown set** S of **b** nodes

⟺

Linear system

$$\mathbf{z}[k+1] = \mathbf{W}\mathbf{z}[k] + \begin{bmatrix} \mathbf{H} & \mathbf{B}_Q \end{bmatrix} \begin{bmatrix} \mathbf{y}[k] \\ \mathbf{f}_Q[k] \end{bmatrix}$$

$$\mathbf{t}[k] = \mathbf{T}\mathbf{z}[k]$$

is invertible for

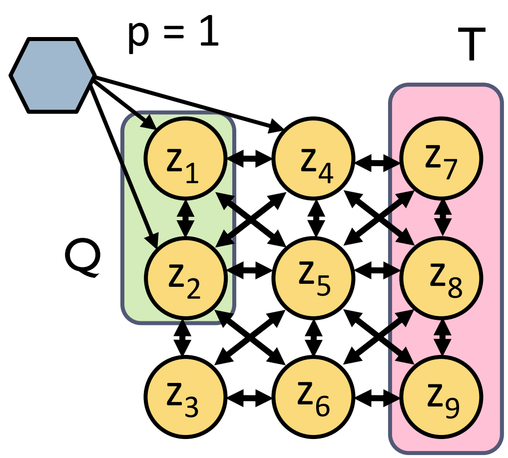
**any known set** Q of **2b** nodes

⟺ **(generically)**

There are p+2b node-disjoint paths from sensors and any set Q of 2b nodes to set T

# Example

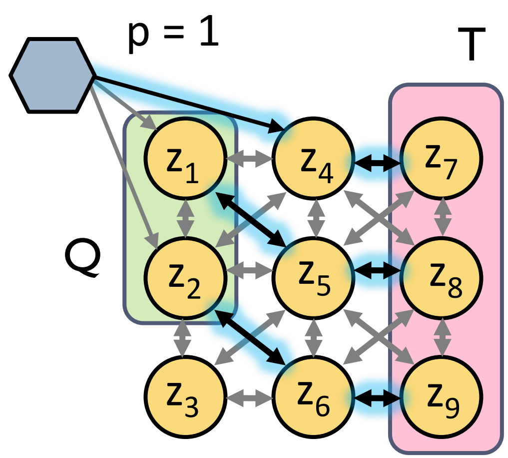
- Suppose we want to identify b = 1 faulty/malicious node and recover the plant outputs in this setting:




- Consider set $Q = \{z_1,z_2\}$

# Example

- Suppose we want to identify b = 1 faulty/malicious node and recover the plant outputs in this setting:




- Consider set Q = $\{z_1, z_2\}$
  - p+2b vertex disjoint paths from sensor and Q to T
- Can verify that this holds for any set Q of 2b nodes
  - Can identify any single faulty/malicious node and recover y[k]

# Summary

- Analyzed linear iterative strategies in networks with malicious nodes

- Key tool: **structured system theory**
  - Provides graph-theoretic analysis of linear systems

- Linear iterative strategies for information dissemination:
  - Robust to b malicious nodes if connectivity of network is 2b+1 or higher (**strong observability**)

- Linear iterative strategies for stabilization with a wireless network
  - IDS can identify b malicious nodes by monitoring a subset of nodes if those nodes have enough disjoint paths from other nodes (**invertibility**)