A Distributed Algorithm for On-Line Diagnosis of Place-Bordered Nets

Şahika Genç, Stéphane Lafortune

Department of Electrical Engineering and Computer Science,
University of Michigan

16th IFAC World Congress in Praha, July 4-8, 2005
Outline

- Introduction
- System Model
- Petri Net Diagnosers
- Distributed Diagnosis with Communication
- Correctness Results
- Conclusion and Future Work
Introduction

- **Accurate** and **timely** diagnosis (detection and isolation) of system faults for safety and reliability.
  - Blackout 2003! (Network Monitoring), MyDoom? (Internet Security)

- Model-based Approach: **Discrete-Event System (DES)** models are adequate for large class of faults.
  - Faults are not detected by sensors.

- Distributed Algorithms: Extendability, scalability, reconfigurability.
  - **Petri nets** (compared to automata) offer better representations of concurrent, asynchronous and **distributed** systems.
Prior Work

• Monitoring and fault diagnosis of dynamic systems using \textit{automata}.
  ○ Many methodologies.

• Monitoring and fault diagnosis of dynamic systems using \textit{Petri net} models.
  ○ \textbf{Alarm supervision} in telecommunication networks by A. Benveniste et al.
  ○ Detection of loss and creation of tokens using \textbf{algebraic techniques} by C. Hadjicostis.
  ○ \textbf{Contextual analysis} of Petri nets for distributed diagnosis by R. Boel and G. Jiroveanu.
  ○ \textbf{Estimating the marking} of a (nondeterministic) Petri net based on the observation of the transition labels by A. Giua.
Diagnoser Approach

- Developed at the University of Michigan under the supervision of S. Lafortune and D. Teneketzis since mid-90s.

- Solution methodology based on diagnoser automata.
  - Theory of diagnosability: Which faults can be diagnosed?
  - On-line diagnosis: How to diagnose?

- Theory successfully applied to
General Architecture

![Diagram of General Architecture]

- **Communication**
  - **Messages**
    - Local Observations
    - System Model
  - **Diagnostics**
    - Module #1
    - Module #2
    - Module #M
General Architecture: A Petri Net Approach
General Architecture: A Petri Net Approach
General Architecture: A Petri Net Approach
System Model: A Labeled Petri net

- A labeled Petri net is defined as a 4-tuple
  - Petri net graph (finite sets of places, transitions, arcs, weights),
  - set of events,
  - transition labeling function,
  - initial state (a vector of size equal to the number of places).

- A transition can fire from a state if and only if the transition is feasible (enabled) from the state.

- When the transition fires, the state transition function gives the resulting state.
System Model: A Labeled Petri net

\begin{figure}
\centering
\includegraphics[width=\textwidth]{system_model.png}
\end{figure}
System Model: A Labeled Petri net

\[ \begin{align*}
  &p_1 \quad t_1 \quad p_4 \\
  &p_2 \quad a \quad p_5 \\
  &p_3 \quad b \quad p_6 \\
  &2 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0
\end{align*} \]

\[ \begin{align*}
  &p_1 \quad t_1 \quad p_4 \\
  &p_2 \quad a \quad p_5 \\
  &p_3 \quad b \quad p_6 \\
  &1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0
\end{align*} \]
System Model

- The system to be diagnosed is a set of $M$ place-bordered labeled Petri nets (modules) where:
  - there are **no common events** between any two modules,
  - there are no **isolated modules**,
  - transitions putting or removing tokens from the common places are labeled with **observable** events, i.e., events that are detected with sensors.
Example: A Heat, Ventilation, Air and Conditioning (HVAC) System
Example: Valve, Pump, and Load
Example: Valve, Pump, and Load
Example: Valve, Pump, and Load
Example: Valve, Pump, and Load
Petri Net Diagnosers

Module #1
Module #2
Module #4
Module #5
Module #6
Coupling
( Common Places )
Subnetworks,
subprocesses, etc.
( Labeled Petri nets )
Network, process, etc.
( System Model )
Module #3

Sahika Genc, Stephane Lafortune, University of Michigan / July 04, 2005
Petri Net Diagnosers
The Petri net diagnoser and the corresponding module have the same Petri net graphs but different dynamics.

Upon observation of an event, the Petri net diagnoser
  - finds the states the module can be in and the fault information and
  - keeps a history of changes in the common places for all neighbor modules.

The modules have common places, i.e., the modules can affect each other’s states.
Petri Net Diagnosers: Diagnoser State

The diagnoser state of module $m$ is a matrix of the form

$$
\begin{pmatrix}
\begin{array}{ccc}
- & - & - \\
 x_s^m(i) & x_f^m(i) & x_l^m(i) \\
- & - & - \\
\end{array}
\end{pmatrix}
$$

where

- $x_s^m(i)$ denotes the state in row $i$ of diagnoser state $x_d^m$,
- $x_f^m(i)$ denotes the corresponding fault label, and
- $x_l^m(i)$ denotes the corresponding message label.
**Petri Net Diagnosers: Faults**

\[ x_d^m = \begin{pmatrix} F_1 & F_2 & F_3 & \ldots \\ - & 1 & 0 & 1 & \ldots & - \\ x_s^m(i) & 1 & 0 & 1 & \ldots & x_l^m(i) \\ - & 1 & 0 & 0 & \ldots & - \end{pmatrix} \]

- **Certain?**
  - Fault of type 1 \((F_1)\) has occurred.
  - Fault of type 2 \((F_2)\) has not occurred.

- **Uncertain?**
  - Fault of type 3 \((F_3)\) *may or may not* have occurred.
Distributed Diagnosis with Communication
Distributed Diagnosis with Communication
Distributed Diagnosis with Communication
Distributed Diagnosis with Communication (DDC-M)

Upon occurrence of an observable event

- identify the (master) module to which the event belongs,

- for all (neighbor) modules whose common places were *effected* during the execution of the event
  - construct the appropriate message,
  - send the **message (without delay)** to the neighbor module,
  - upon receipt of the message the neighbor module updates the diagnoser state to reflect the **changes on the common places**.
Example

Module $m$ ($\mathcal{M}_m$)

\[
\begin{pmatrix}
 a_1 & \ast & \alpha_1 & : & \ast \\
 a_2 & \ast & \alpha_2 & : & \ast \\
 \tilde{\text{state}} & \tilde{\text{fault}} & \tilde{P}_{m,n}
\end{pmatrix}
\]

Module $n$ ($\mathcal{M}_n$)

\[
\begin{pmatrix}
 b_1 & \ast & \beta_1 & : & \ast \\
 b_2 & \ast & \beta_2 & : & \ast \\
 \tilde{\text{state}} & \tilde{\text{fault}} & \tilde{P}_{m,n}
\end{pmatrix}
\]
Example

Module $m$ ($\mathcal{M}_m$)

\[
\begin{pmatrix}
  a_1 & \ast & \alpha_1 : \ast \\
  a_2 & \ast & \alpha_2 : \ast \\
\end{pmatrix}
\]

$\sim$state $\sim$fault $\sim P_{m,n}$

Module $n$ ($\mathcal{M}_n$)

\[
\begin{pmatrix}
  b_1 & \ast & \beta_1 : \ast \\
  b_2 & \ast & \beta_2 : \ast \\
\end{pmatrix}
\]

$\sim$state $\sim$fault $\sim P_{m,n}$

$\downarrow \sigma_o \in \Sigma_{o,m}$
Example

Module $m$ ($M_m$)

\[
\begin{pmatrix}
a_1 & * & \alpha_1 : *\\
a_2 & * & \alpha_2 : *\\
\end{pmatrix}
\]

\[\text{state} \quad \text{fault} \quad \tilde{P}_{m,n}\]

\[\downarrow \sigma_o \in \Sigma_{o,m}\]

\[
\begin{pmatrix}
a_1 + w_1 & * & \alpha_1 w_1(P_{m,n}) : *\\
a_2 + w_2 & * & \alpha_2 w_2(P_{m,n}) : *\\
\end{pmatrix}
\]

\[P_{m,n}\]

Module $n$ ($M_n$)

\[
\begin{pmatrix}
b_1 & * & \beta_1 : *\\
b_2 & * & \beta_2 : *\\
\end{pmatrix}
\]

\[\text{state} \quad \text{fault} \quad \tilde{P}_{m,n}\]
Example

Module $m$ ($\mathcal{M}_m$)

$$
\begin{pmatrix}
  a_1 & \ast & \alpha_1 : \ast \\
  a_2 & \ast & \alpha_2 : \ast \\
  \text{state} & \text{fault} & P_{m,n}
\end{pmatrix}
$$

\[ \downarrow \sigma_o \in \Sigma_{o,m} \]

$$
\begin{pmatrix}
  a_1 + w_1 & \ast & \alpha_1 w_1(P_{m,n}) : \ast \\
  a_2 + w_2 & \ast & \alpha_2 w_2(P_{m,n}) : \ast \\
  \text{state} & \text{fault} & P_{m,n}
\end{pmatrix}
$$

Module $n$ ($\mathcal{M}_n$)

$$
\begin{pmatrix}
  b_1 & \ast & \beta_1 : \ast \\
  b_2 & \ast & \beta_2 : \ast \\
  \text{state} & \text{fault} & P_{m,n}
\end{pmatrix}
$$

\[ \downarrow \text{Message} = \left( \begin{array}{c}
\alpha_1 w_1(P_{m,n}) \\
\alpha_2 w_2(P_{m,n})
\end{array} \right) \]
Example

Module $m$ ($\mathcal{M}_m$)

\[
\begin{array}{c|c|c}
\text{state} & \text{fault} & \ P_{m,n} \\
\hline
a_1 & * & \alpha_1 : * \\
\hline
a_2 & * & \alpha_2 : * \\
\end{array}
\]

\[\downarrow \sigma_o \in \Sigma_{o,m}\]

\[
\begin{array}{c|c|c}
\text{state} & \text{fault} & \ P_{m,n} \\
\hline
a_1 + w_1 & * & \alpha_1 w_1(P_{m,n}) : * \\
\hline
a_2 + w_2 & * & \alpha_2 w_2(P_{m,n}) : * \\
\end{array}
\]

Module $n$ ($\mathcal{M}_n$)

\[
\begin{array}{c|c|c}
\text{state} & \text{fault} & \ P_{m,n} \\
\hline
b_1 & * & \beta_1 : * \\
\hline
b_2 & * & \beta_2 : * \\
\end{array}
\]

\[\downarrow \text{Message} = \left( \begin{array}{c} \alpha_1 w_1(P_{m,n}) \\ \alpha_2 w_2(P_{m,n}) \end{array} \right) \]

$\beta_1 = \alpha_1, \ \beta_2 = \alpha_2$
Example

Module $m$ ($\mathcal{M}_m$)

\[
\begin{pmatrix}
  a_1 & * & \alpha_1 : * \\
  a_2 & * & \alpha_2 : *
\end{pmatrix}
\]

\[\tilde{\text{state}} \quad \text{fault} \quad \tilde{P}_{m,n}\]

\[\downarrow \sigma_o \in \Sigma_{o,m}\]

\[
\begin{pmatrix}
  a_1 + w_1 & * & \alpha_1 w_1(P_{m,n}) : * \\
  a_2 + w_2 & * & \alpha_2 w_2(P_{m,n}) : *
\end{pmatrix}
\]

Module $n$ ($\mathcal{M}_n$)

\[
\begin{pmatrix}
  b_1 & * & \beta_1 : * \\
  b_2 & * & \beta_2 : *
\end{pmatrix}
\]

\[\tilde{\text{state}} \quad \text{fault} \quad \tilde{P}_{m,n}\]

\[\downarrow \text{Message} = \begin{pmatrix}
  \alpha_1 w_1(P_{m,n}) \\
  \alpha_2 w_2(P_{m,n})
\end{pmatrix}\]

\[\beta_1 = \alpha_1, \quad \beta_2 = \alpha_2\]

\[\text{state} = \begin{pmatrix}
  b_1(P_{m,n}) + w_1(P_{m,n}) & b_1(P_{n \setminus m,n}) \\
  b_2(P_{m,n}) + w_2(P_{m,n}) & b_2(P_{n \setminus m,n})
\end{pmatrix}\]

\[\text{message label} (P_{m,n}) = \begin{pmatrix}
  \beta_1 w_1(P_{m,n}) \\
  \beta_2 w_2(P_{m,n})
\end{pmatrix}\]
Outline

- Introduction
- System Model
- Petri Net Diagnosers
- Distributed Diagnosis with Communication
- Correctness Results
- Conclusion and Future Work
Correctness Results

Monolithic Diagnosis

Distributed Diagnosis

Observed sequence of events
\( \sigma_0, \sigma_2, \cdots, \sigma_{R} \)

\( x_d^R \)

\( x_{d,1}^R \)

\( \cdots \)

\( x_{d,M}^R \)

EQUAL

Merge

\( \sigma_0, \sigma_2, \cdots, \sigma_{R} \)
Correctness Results

Monolithic Diagnosis

Distributed Diagnosis

Observed sequence of events

$\sigma_0^1 \sigma_0^2 \ldots \sigma_0^R$

$\mathbf{x}_d^R$

EQUAL

Merge

$\mathbf{x}_{d,1}^R$

$\cdots$

$\mathbf{x}_{d,M}^R$

Recover the monolithic diagnosis information using distributed diagnosis.
Correctness Results

- **Merge** operation combines the diagnoser states of the modules.

- Form the **monolithic system** by combining all the place-bordered labeled Petri nets in the system into one labeled Petri net.
Correctness Results

- **Merge** operation combines the diagnoser states of the modules.

- Form the **monolithic system** by combining all the place-bordered labeled Petri nets in the system into one labeled Petri net.

**Theorem:** Merge operation recovers the monolithic diagnoser state.
Correctness Results

- **Merge** operation combines the diagnoser states of the modules.

- Form the **monolithic system** by combining all the place-bordered labeled Petri nets in the system into one labeled Petri net.

**Theorem:** Merge operation recovers the monolithic diagnoser state.

- The proof of the theorem is by construction of DDC-$M$ and induction on the observed sequence of events.
Software Implementation of DDC-$M$
Progress to Date

COMPLETE ✓

- Online monitoring and diagnosis of modular systems modeled as place-bordered nets
  - DDC-M Algorithm
  - DDC-M Algorithm with fixed-size message labels (to appear in the journal paper)
- A software implementation developed in MATLAB
Progress to Date

COMPLETE ✓

- Online monitoring and diagnosis of modular systems modeled as place-bordered nets
  - DDC-$M$ Algorithm
  - DDC-$M$ Algorithm with fixed-size message labels (to appear in the journal paper)
- A software implementation developed in MATLAB

IN PROGRESS ?

- Relax assumptions made for DDC-$M$ Algorithm
  - Proper partitioning of a system into modules
  - Shared events among the modules
- Analysis of diagnosability properties in the case of Petri net languages
- Diagnose sequences (patterns) of faults