Transient Fault Detection in Networked Control Systems

Xiong-Feng Huang, Chun-Jie Zhou, Shuang Huang, Kai-Xin Huang, and Xuan Li

School of Automation, Huazhong University of Science and Technology, 1037 Luoyu Road, Wuhan 430074, China

Correspondence should be addressed to Chun-Jie Zhou; cjiezhou@hust.edu.cn

Received 3 May 2014; Revised 9 October 2014; Accepted 16 October 2014; Published 12 November 2014

Academic Editor: Kyung-Chang Lee

Copyright © 2014 Xiong-Feng Huang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Transient faults appear unpredictably and frequently in networked control systems (NCSs), which directly affect the safety of system. However, they cannot be accurately identified due to various kinds of causes, so the detection of them needs more systematic and comprehensive system knowledge. Hence, a hierarchical and systematic detection approach based on function, behavior, and structure (FBS) models is proposed in this paper. The FBS models are established according to systematic knowledge of NCSs. And the transient faults are excavated in node level and system level, in which parameter match and model-based detections are employed. Moreover, the key implementation aspects of the proposed approach in NCSs are detailed on parameterization of FBS models, criteria, and feature parameters of faults, fault monitoring, fault detection, and real-time scheduling considering fault detection. At last, a simulation platform of a networked control liquid mixer system is described by Architecture Analysis and Design Language (AADL), and a transient fault is injected before the sample task in the node for fault detection. The simulation results show that the faults are exposed in the gradual detection hierarchies and the effectiveness of the detection scheme is quite promising.

1. Introduction

Networked control systems are widely applied in modern industrial fields, such as aerospace, automotive and transportation [1, 2]. And to improve system safety, the detection and handling of transient faults are concerned because of their hazards [3–7]. However, the transient fault is hard to be detected and located due to their unpredictable nature and short duration. Transient faults are widely various due to diverse design flaws and environmental interference [8–11], and they occur in anywhere and anytime in the system with a higher rate than permanent faults [12]. And transient faults always remain active for a short period of time and eventually disappear without any apparent intervention [13]. With the increasing system scale and complexity of NCSs, transient faults are unavoidable even though many fault-tolerant methods are considered. For these reasons, transient fault handling is challengeable and imperative for establishing safety control system.

Currently, transient fault detection approaches mainly focus on the models from the knowledge of control mechanism and communication or pure embedded realization platform but not systematic system knowledge for NCSs. Some literatures focus on modeling NCSs from the knowledge of control mechanism and communication. Ahmadi et al. [14] modeled NCS considering network time delay, from which the bound of system stability was discussed for fault detection. Xie et al. [15] modeled NCS considering packet dropout and drew the maxim dropout for fault detection. Zhang et al. [16] modeled the communication task scheduling as system models and conducted the fault detection. The works mentioned above implement the fault detection on account of single or more parameters. Considering that the networked control system is software-intensive, the transient fault detection in NCS needs a systematic approach containing the fault detection in embedded realization platforms. For detecting fault in embedded systems, Pinello et al. [17] discussed the execution platform faults and proposed a fault-detection and tolerance approach, Li and Hong [18] discussed the control flow and data flow model for fault detection in a space control system, and Bernardi et al. [19] discussed the fault detection in a software-implemented approach. Some traditional fault detection architectures (i.e., FTA and CCS) focused on how to detect the source of fault. But considering with the short duration of transient fault and online detection requirements,
the sources of transient faults in NCSs are hard to be located and the fault will be detected by subsequent fault propagation effects. At all, to detect the transient faults in NCSs, a novel approach is required considering entire system knowledge. Some literatures have explored this research. To model the system based on entire system knowledge, Chittaro et al. [20] proposed a multimodeling approach with teleological, functional, behavioral, and structural knowledge. Functional and behavioral knowledge are used for fault detection in an intelligent and autonomous system [21], and in [22] entire system knowledge is considered in a system supervisory framework.

In our work, the entire system knowledge is used for system modeling of NCSs and the detection of transient faults is implemented considering the system control and the system realization. As similar to system modeling concepts in [22], we use intentions and means as the knowledge for modeling NCSs, which runs through the conceptual design and the system realization. The intentions contain system functions, and the means contain system structure and behaviors. Function reflects the desired system goals, and it will be achieved by the behaviors of the system. The structure reflects the system realization and it supports the system behaviors. A FBS model covers the system models of NCSs. For fault detection, hierarchical detection is established from node-level to system-level. Node-level detection aims at the faults occurring at the nodes, and the system-level detection concentrates on communication fault, control fault, and fault propagation. Without focusing on locating fault source, the fault propagation effect traces are used to arrange the probes for system monitoring. AADL [23] and its supporting suites are employed to describe the system models. By injecting transient faults and tracking system running, the effects caused by the faults can be identified by parameter matching and model-based detection.

This paper is intended to describe systematic transient fault detection in NCSs. The main contribution is that we proposed a modeling approach based on entire knowledge of NCSs and a transient fault detection approach considering system realization. And to verify the transient fault detection approach, a unified platform based on AADL which reflects the system models is recommended. The remainder of this paper is organized as follows. We analyze the requirements of transient fault detection in NCSs in Section 2. And then we introduce the knowledge for establishing FBS models of NCSs in Section 3. Next, the detailed transient fault detection approach is expatiated in Section 4. Moreover, the implementation of the proposal is explicated in Section 5. A case study is specified on a simulation platform described by AADL and the simulation results are demonstrated in Section 6. Finally, the conclusion and future works are discussed in Section 7.

2. Requirement Analysis of Transient Fault Detection for NCSs

In this section, we first introduce the characteristics of NCSs. And then the transient faults in the system and their causes are inspected. At last, the key issues of the transient fault detection are discussed.

2.1. Characteristics of NCSs. Networked control systems are distributed control systems in which sensors, actuators, controller, plant, and other devices are interconnected by communication networks [24]. A classical architecture of NCS is shown in Figure 1. The architecture contains two parts—plant and a control system based on communication network.

With increasing application, NCSs are becoming more complex and the scales of them are becoming bigger and bigger. And to implement the control processes, more and more software is embedded in NCSs. All those cause the external interferences and the internal dysfunction to be unavoidable. Thus, fault detection and tolerance are the problems that could not be avoided for establishing a safety control system. As far as we know, transient fault occurs more frequently than permanent fault [11], and there exist some

![Figure 1: A classical architecture of NCSs.](image-url)
reasonable solutions to permanent fault. So here we focus on the detection of transient fault in NCSs.

2.2. Transient Faults in NCSs. Transient fault is a dominant type of fault in control systems [11], which occur with burstiness and randomness. They are caused by external interferences (i.e., EMI [25], attack [26], and high-energy particle [27]) and internal dysfunctions (design flaw [28], software bugs [29], etc.) which could not be predicted. High occurrence rate and unpredictability lead the transient faults in NCSs hard to be enumerated. Meanwhile, the complex realization of NCSs causes transient faults to be unavoidable; even reasonable fault tolerant control design is considered. On the other hand, a fault could be caused by many factors [30], that leads transient fault hard to be located and eliminated. In NCSs, the source of transient fault may propagate to affect the subsequent operations and brings in a new fault [31]. In the design of transient fault detection, not only traditional fault detection means but also the fault propagation needs to be considered.

2.3. Key Issues of Transient Fault Detection. Because of the natures mentioned above, the implementation of transient fault detection in NCSs generally contains the following issues.

(1) Modeling of NCSs. NCSs is crossing process control, communication, and computer fields, so that single knowledge is insufficient for modeling. The organization of entire knowledge is still a challengeable problem. In the representation of system models, quantitative models are hard to be established and qualitative models are coarse for entire system knowledge. A reasonable combination of these two types of models will be envisaged.

(2) Hierarchical detection design. Most often, in the presence of transient fault, diverse phenomenon may be presented at different system levels, and the occurrence time and destructive effect are discrepant. The choices of abstraction hierarchies and fault phenomena have a significant impact on implementing real-time and low-cost detection.

(3) Real-time assurance. When a fault occurs, we hope it will be detected as soon as possible. And fault detection and system monitoring will bring additional tasks to NCSs. All those need to be taken into account in the implementation and previous scheduling design.

To establish a reasonable transient fault detection approach for NCSs, we use the FBS model for entire knowledge representation, which will be explicaded in the next section.

3. Knowledge-Based FBS Models for Modeling NCSs

3.1. Knowledge for Modeling NCSs. System knowledge consists of four types to cover the intentions and the means. Teleological knowledge and functional knowledge are to represent the system intentions, and they can be reflected by functions which reveal the desired actions of the system. Each function is achieved by a group of behaviors which are drawn from the system behavioral knowledge. Behavioral knowledge is about possible behaviors in the system which describes how system or components can work and interact. Generally, variables and parameters are used to characterize the states and the laws which rule their transitions. Structural knowledge is about system topology which describes the constitution of the system and the connection among the components. One behavior is supported by one or more components. And a same behavior may be supported by different components in different system structure. The four aspects of knowledge cover from system design to system implementation. Moreover, they can be correlated systematically by the terms—"achieved by" or "support." The models of NCSs could be constructed under this knowledge.

3.2. FBS Models of NCSs. Function models, behavior models, and structure models compose the FBS models. FBS models and their relationships are shown as Figure 2. Function models originate from the designer's intentions and the roles of components in the physical processes. Behavior models describe how components work and interact. Structure models describe the system topology and resources. The abstraction hierarchies are interrelated by "achieved by" and "achieves." A fault occurs in some abstraction hierarchy and it influences the other one. For instance, a faulty behavior in the sensor node will make it lose the sensing function.

The function is a characteristic action or activity that needs to be performed to achieve a desired objective. Its realization relies on services provided by the physical components which constitute the system. Each function is achieved by the behaviors revealed in the functional modules. For NCS, the functions can be roughly divided into two categories, namely, communication and control. Furthermore, they can be subdivided into sensing, decision-making, actuating and communicating. Function models of NCS can be quantized by a set of services and their quality, for example, quality of communication and quality of control.

Behaviors describe all the possible potential operations of components in NCS, which can be represented as a sequence of states and transitions. Generally, the behaviors in NCS are expressed as a verb, that is, send, receive, calculate, and rotate, and each behavior can be refined [32] according to the granularity of structural representation. Finite state machine (FSM) [33, 34] is used to describe the states and transitions formally. In our research a behavior is formally represented by a tuple as \( \{ \text{Node\_ID}, \text{Beh\_ID}, \text{Beh\_FSM} \} \). \text{Node\_ID} denotes the node the behavior carries out, \text{Beh\_ID} reveals the specific behavior in behavior models, and \text{Beh\_FSM} is to represent the states and transitions in the behavior.

Structure models describe the resources in the system and their statuses and compositional relationships, which are divided into two level abstractions. In system-level, node is the element. Total number of the nodes and the connection statuses need to be considered. And in node-level, task is
thought as the basic element. The execution statuses and the occurring sequences of them are used to represent the node.

To establish the FBS models, the intention will be stated explicitly first, and then the functions need to be listed from the control and communication aspects to support the intention. On the other side, the structure of the system is represented by the nodes and the tasks in node according to the realization, and the behaviors showing on the structures need to be listed in detail. At last, the behaviors and the functions need to be associated with the supporting relationships. The FBS models of NCSs are shown in Figure 3 in which only the system-level structure is considered.

4. FBS Model Based Hierarchical Transient Fault Detection

FBS models contain the functional, behavioral, and structural information of NCSs which are exactly used to deduce the fault effects for fault detection. When the information violates the specification, a fault can be identified. In this section
a fault detection approach consists of parameter matching and model-based detection is explained.

4.1. Hierarchical Transient Fault Detection Approach. The occurrence of transient faults in NCSs is unpredictable. The detection generally depends on their propagation effects. The effects can be presented as different features in FBS models. Some of them can be explored by simple parameter match, such as threshold and change rate. And the others need to be discovered through a complicated derivation (timing model, schedule, etc.). These features exist hierarchically in the FBS models, and a low-level fault could propagate into a high-level fault if not tolerated. The detection of transient faults in NCSs will be implemented by a hierarchical scheme in which the faults in the nodes and the system are fully covered.

A hierarchical scheme for the detection of transient faults in NCSs is shown in Figure 4. It consists of node-level detection and system-level detection. Parameter match is applied to inspect the characteristic features and the model-based detections are employed to excavate the undetected faults. System level models especially are employed to detect the undetected and propagable faults in node level.

4.2. Node-Level Transient Fault Detection. Transient faults on node level present as false states or state transitions of task behaviors, control data error, exceeding the threshold of communication load and CPU utilization, and false states of node resources (components). These faults emerge as deviation of the characteristic feature parameters of the system models on different abstraction level. Some of them can be identified by parameter match, but some of them need to be excavated by diverse models. The node-level detection consists of node-level parameter match and model based detection.

During the parameter match, the monitoring data from the probes are compared with the correlative data in a database which contains the symptoms of node-level transient faults that can be enumerated in advance from the experiential knowledge. If the mismatch is discovered, the fault will be reported to the fault manager.

The behaviors in each node are expressed by the models, that is, FSM models and timing models. The detection of this type of faults depends on the rules of the models. Meanwhile, some detection relies on not only the parameters but also the rules of the models. In model-based detection, the violations of the rules are the criterion for the identifications of faults. For example, when the monitoring data input, the output of the model, is out of the specified domain, then a violation appears.

4.3. System-Level Transient Fault Detection. Transient faults on system level are presented as the errors of external characteristics of the plant and the control system. System error of the plant is defined as the mismatching between the desired value and the observed value of the outputs of plant and the state variables of the observations. The error of the control system is defined as the degraded performance of control system, for example, some tasks cannot be scheduled because of the suddenly increasing network loads. Some parameters are used to represent the system errors, that is, the output of the plant, the communication time, the network bandwidth, and the utilization.

In system level parameter match, the monitoring system characteristic data from the probes, such as QoC and QoS, are used to compare with the correlative data in a database which reflects the symptoms of node-level transient faults that can be enumerated in advance from the experiential knowledge for transient fault detection. If mismatch is discovered, the fault will be reported to the fault manager. In another side, the undetected and propagable faults from the nodes will be excavated based on the system level models, that is,
components in the plant are expressed by the devices in a
prescribed granularity. Their statuses reflect the operations of
the physical system.

Behavior models contain communication models, schedul-
ing models, node task executing models, and plant oper-
ating models. These models are formally described based
on extended finite state machines (EFSM) [35]. An EFSM is a
6-element tuple \((S, s_0, E, f, O, V)\), where \(S\) is a finite set of
local states, \(s_0\) is an initial state, \(E\) is a set of events, \(f\) is a
state transition function, \(O\) is a set of output signals, and \(V\)
is a set of variables. The function \(f\) returns a next state, a set
of output signals, and an action list for each combination of a
current state and an input event. The communication models
consist of receive EFSM and send EFSM. And the scheduling
models are made up of a communication-schedule EFSM and
node-task-schedule EFSM. And the task executing models
are composed of multiple EFSMs according to the functions
of each node, such as measure EFSM and data filter EFSM
in a measuring node, compute EFSM and manage EFSM in
controller nodes, and actuate in EFSM in actuating nodes.
The plant operating models consist of the EFSMs which are
determined by its actions, and here those are not focused on.
An example of EFSM is shown in Algorithm 1. Each behavior
can be expressed in this manner.

Every function can be represented by a phrase, that is,
data acquisition and valve open calculation. They can be
characterized by functional flags, QoC parameters, and QoS
parameters. Generally, the functions in NCS are classified
according to the requirements such as "to acquire the mon-
itoring data and send them to the controller," "to collect the
monitoring data and make control strategies," and "to receive
the strategies and affect the control inputs of the plant." A
functional flag which is valued by correctness or fault is
to assess whether the function is met. QoS is to evaluate
the communication and QoC is to estimate the effect of
control strategies. QoS parameters contain collective latency,
collective packet loss, collective bandwidth, and information
throughput. If these parameters exceed the thresholds, a
fault may happen. QoC parameters contain the real-time
performance of the control strategy and the predicted and
measured control errors of the objects. And the functional
structure integrity of whole system is also a critical factor,
which is used to reveal that the system is under close-loop
control by a series of functional flags.

5. Implementations of Hierarchical Transient
Fault Detection in NCSs

The detection of transient faults is conducted based on the
observable parameters in the knowledge representation of
NCSs. In this section the FBS models are parameterized to
depict the measurable parameters in NCSs firstly, which
coverage the resources, FSMs, and flags. And then the fault
judgments in NCSs are provided to ascertain the criterion of
fault. Next, the implementation of the monitors and the
detectors is specified in detail. At last, we discuss the task
schedule for real-time assurance considering the increasing
tasks and loads due to the inserted fault detection.

5.1. Parameterized FBS Model Representations of NCSs. Structure
models consist of the resources, such as hardware and
software, and their statuses and relationships. On system
level, each node is identified by the only Node.ID. And a
tuple \{node(i), related_node(j), status\} \((i \neq j)\) is used to
formalize the node considering the connectability and the
adjacent nodes. In a master-slave network especially, each
slave node is expressed as \{slave_node.ID, master_node.ID, connectability\}. On node level, each task is identified by
the only task.ID. And the task is formalized as the tuple
{task.ID, sequence}. CPU utilization and memory usage are
employed to represent the execution platform hardware. The

\[
EFSM = (S, s_0, E, f, O, V)
\]

(1) States \((S) = \{\text{WAITED, TRANSMITTED, RESENT, FAULT-REPORT}\}.
(2) Initial State \((s_0) = \text{WAITED}
(3) Events \((E) \equiv \{\text{REQ, ACK, SEND, RST, TIMEOUT}\).
(4) Transition Function \((f) = \{\text{described in the state transition diagram in Figure 5}\}
(5) Output Signals \((O) = \{\text{Transmitted (message), Return Error (error message), Re-Transmitted (message)}\).
(6) Variables \((V)\)
\]

\begin{align*}
\text{REXMT: Retransmission Timer} \\
\text{TIMEWAIT: Time-wait Timer}
\end{align*}

Algorithm 1: The EFSM of the behavior "send."
5.2. Judgments of Transient Faults in NCSs. The transient fault judgments depend on the observable attributions of the faults. The transient faults in NCSs are characterized on system level and node level. System-level fault symptoms contain nonschedulability of the communication loads, incorrect QoS of network, non-real-time capability of the control strategies, imperfection of the functional structure, and uncontrollability of the system under the controller. And node level fault symptoms contain nonschedulability of the tasks in each node platform, output error of the plant (i.e., temperature, current, angle, etc.), incorrect state transitions (violations of the transmission rules), unexpected working states during operation (full CPU utilization, insufficient memory usage, etc.), and variables exceeding the predefined thresholds (collected data variables, such as outputs, observable transmitting state variables in nodes, and plant).

Actually, the detection of transient faults is to identify the violations of the state transitions and the unaccepted errors of the variables. Generally, the fault symptoms appear not only on node level but also on system level. Therefore, these feature parameters of faults can be obtained on different levels.

5.3. Two Level Monitors. The monitors in the system are categorized as system level monitor and node level monitor as shown in Figure 6. QoS monitor is deployed in master nodes, in which the time stamps, sequence and number of the messages are used to acquire the collective latency and the collective packet loss and a message sender in slave nodes and a collective bandwidth tester and an information throughput tester in master node are employed to measure the bandwidth and throughput. QoC monitors measure the whole time of the sampling, processing and actuating and communicating in a control period. Schedule monitor gets the flag of the communication scheduler, and functional integrity monitor supervises the connectivity of the nodes and the correctness of the node functions. And control performance monitors measure the output errors of the control objects by comparing the predicted value with the measured value.

On node level, there are four types of monitors: communication monitor records the states and the state transitions of the communication behaviors; task schedule monitor gets the flag of the task scheduler; resource monitor supervises the hardware platform running statuses (i.e., CPU utilization and memory usage); function monitor records the states and the state transitions of the task executing behaviors in which the function is defined as a correct execution process.

The QoS monitors are waked up if the QoC parameters are abnormal. And the schedule monitors are working when the task or load is changed. And the other monitors work once at every control period. Obviously, a part of the monitoring data can be processed at the slave node for fault detection, and the others can be processed at the master node. Therefore, hierarchical detections are wanted to carry out fast and systematic fault detection.

5.4. Hierarchical Fault Detector. Fault detectors consist of two types of detectors. Node-level detector is deployed in each single node, and system-level detector is arranged in master node. In the detectors, anomaly detection [36] is employed to identify the faults. Feature parameters and diverse models with constraints are specified in a database declared previously. Some of the monitoring variables are directly compared with the feature parameters to find the feature exceeding the threshold, and the others are inspected by the models to discover the nonconforming patterns as anomalies. The architecture of the detector is shown in Figure 7.

Three types of monitoring data are addressed at the node-level detectors, such as node resources, node behaviors, and node running statuses. On the other hand, the data from system level monitors are processed by the system level detector. All the detectors are driven by the events from the corresponding monitor.

5.5. Scheduling for Real-Time Fault Detection. Data monitoring and fault detection increase the node tasks and the communication loads, which may influence the real-time performance of the control system. In master-slave manner control networks, the partition of the time slots for each slave node is flexible according to the schedule mechanism, in which the increasing communication loads and tasks in node are considered. In a multitasking system, we divide the tasks in the system into four types: initial tasks for control Task_init, sub control tasks Task_ctrl, communication for control Task_com, and others Task_other. Meanwhile, a control period is divided into three stages as sensing, processing, and actuating. A global system task schedule scheme is shown in Figure 8. Different nodes are dispatched with different tasks for the implementation for the close loop control. For example, a sensor node may execute initialization, communication, and control tasks. The control tasks specially contain monitoring and fault detection. In each node, the other time slots for the other services are scheduled by some well-known algorithms, that is, RM [37], and EDF [38–40].

In each communication stage, all the communication tasks in each relevant node can be scheduled by the algorithms mentioned above. In the slave node, the fault report will be treated as a sole task. In such a manner, the node level faults are detected by the local detectors in the dispatched task stage instantly and the detection of the system level faults are achieved in a certain time, which ensures the real-time control of the system.

6. Case Study

In this section, a networked control liquid mixer system (NCLM) is described for the verification of the proposed
fault detection approach. FBS models of the NCLM are built up for the analysis of fault detection, and the simulation is conducted based on AADL suites.

6.1. Networked Control Liquid Mixer System. The networked control liquid mixer system is shown in Figure 9. A tank is used to hold liquid for a certain chemical reaction, and the temperature needs to be kept at 60°C within a margin of error (±3°C) and the liquid level needs to be maintained at 1.0 meter within a margin of error (±5%). The control period of temperature is 400 ms as quadruple as the control period of level. And the bottom area of the tank is 1 m², flow $A$ is heat and flow $B$ is cool, and the temperatures of them are 85°C and 25°C, respectively. Valve $V_A$ is used to adapt the heat flow which is supplied by the liquid source with a variable rate $fa$ as $0.001 + 0.0001 * t$ (t, ms, sawtooth function, step = 1, $t \in [0,100]$) m³/s, valve $V_B$ is regulated by the valve control node (VCN) to keep the temperature, and valve $V_C$ is regulated by the valve actuation node (VON) to keep the level. Flow measurements $A$, $B$, and $C$ (FMA, FMB, and FMC) are used to measure flow $A$ and flow $B$, level measurement LM is used to probe the liquid level, and temperature measurement TM is used to detect the liquid temperature. The realized control system consists of six nodes and a CAN (controller area network) communication network. TSN, FLMN, and FON are the control nodes for sensing temperature, flows $A$ and $B$, level and flow $C$. And the centralized control node (CCN) is the controller.

The flow rates $fa$, $fb$, and $fc$, which denote the flow of pipes $A$, $B$, and $C$, multiply 0.01, 0.01, and 0.02, respectively, to get the degrees of the valves. The control rule is made according to the first law of thermodynamics [41], and a simple controller is shown as the following:

\[
fa(k+1) = fa(k) + 0.001, \quad fa(0) = 0.01,
\]
\[
fb(k+1) = (85 - T(k)) * fa * \left( \frac{k}{T(k) - 25} \right),
\]
\[
fc(k+1) = fa(k) + fb(k) + L(k) - 1.0,
\]
\[
va(k+1) = \frac{fa(k+1)}{0.01},
\]
\[
vb(k+1) = \frac{fb(k+1)}{0.01},
\]
\[
vC(k) = \frac{fc(k)}{0.02},
\]
The structure models of NCLM contain global node resources, node tasks, CPU state and memory size, and QoS and QoC (real-time performance). The global node resources are denoted by \{CCN, all, connectivity\}, \{VCN, CCN, connectivity\}, \{VON, CCN, connectivity\}, \{FLMN, CCN, connectivity\}, \{TSN, CCN, connectivity\}, \{FON, CCN, connectivity\}. Node tasks are listed as \(T_{\text{initialize}}, T_{\text{schedule}}, T_{\text{sample}}, T_{\text{filter}}, T_{\text{calculate}}, T_{\text{send}}, T_{\text{receive}}, T_{\text{others}}\). Different tasks are dispatched in the nodes; that is, in node TSN, there are tasks including \(\{T_{\text{initialize}}, 1\}, \{T_{\text{schedule}}, 2\}, \{T_{\text{sample}}, 3\}, \{T_{\text{filter}}, 4\}, \{T_{\text{send}}, 5\}\), and other tasks for other applications.

The function and the behavior are associated with each other according to the supporting relationships, while different tasks in the nodes are linked to different behaviors. By detecting the violations of the specified FBS models, the transient faults will be exposed.

6.2 Simulation Based on AADL Tool Suites. The simulation platform of the NCLM system is described by AADL in Open Source AADL Tool Environment [42]. The architecture of the NCLM system is shown in Figure 10. Seven AADL processes are used to model the six nodes in the realized control system and the plant. The six nodes are connected by CAN network. The plant is linked with the sensor nodes and actuator nodes through the IO bus.

The tasks in TSN nodes are described by AADL threads as shown in Figure 11. Three application tasks (TSNcomm, TSNfilt, and TSNsample), one initialization task (TSNsyn), and three fault detectors (TSNFD3T, TSNFD2T, and TSNFDIT) are stated. Considering the causal relationships of the tasks and real-time fault detection requirement, the tasks are scheduled as the sequence as “TSNsyn, TSNsample, TSNFDIT, TSNfilt, TSNFD2T, TSNcomm, TSNFD3T”. By a reasonable assessment, we know that the total execution time of the tasks is under 20 ms which satisfies the global schedule requirement. Besides the plant node, the other nodes are scheduled similarly. Because of the different control period.

| Table 1: Function models of NCLM system. |
|----------------------------------------|
| **Intention of realized control system** | **Function** | **Subfunctions** | **Parameterization** | **Supporting behaviors** |
|----------------------------------------|--------------|------------------|----------------------|-------------------------|
| To keep the temperature at 60°C and maintain the level at 1.0 m | F1: control | SF1: sampling | flag(F_FON) = true & flag(F_TSN) = true & flag(F_FLMN) = true | B(FLMN, sample), B(FON, sample), and B(TSN, sample) |
| | F2: communication | SF2: sample data collecting | flag(F_FCN) = true & flag(F_TSN) = true & flag(F_FLMN) = true | B(FLMN, send), B(FON, send), and B(TSN, send) |
| | | SF3: control strategy making | flag(F_FCN) = true & flag(F_TSN) = true | B(TSN, receive) |
| | | SF4: control strategy releasing | flag(F_FCN) = true & flag(F_VCN) = true | B(TSN, receive) |
| | | SF5: control objects updating | flag(F_FCN) = true & flag(F_VCN) = true | B(TSN, receive) |

where \(fa, fb, \) and \(fc\) are the flow rates of pipes A, B, and C, and \(va, vb, \) and \(vc\) are the opening degree of \(V_A, V_B, \) and \(V_C, k\) is the time step which varies as sawtooth wave ranging from 0 to 100, \(L(k)\) denotes the measured level, and \(T\) is the measured temperature.

Each subfunction is supported by the behaviors of the nodes. The detailed specification of the function models of NCLM is shown in Table 1. Two functions and five subfunctions are used to portray the intention of the realized control system, and a series of node behaviors are listed. SF1, SF3, and SF5 are subject to F1, and SF2 and SF4 are subject to F2. Different behaviors may co-exist in one node, and each of them has different impact on the subfunctions. For example, \(B\) (TSN, sample) and \(B\) (TSN, send) are produced by the same node TSN, but one is to obtain the temperature of the plant and the other is to send this data to the controller. The behaviors of NCLM system are supported by the realized control system, which is rendered as the node behaviors, such as \(send, receive, schedule,\) and \(sample.\) And they are classified into three categories, which are shown in Table 2.

| Table 2: Behavior models of realized control system. |
|----------------------------------------|
| **Type** | **Expression** |
|----------------------------------------|----------------|
| Communicating | B(FLMN, send), B(FON, send), B(TSN, send), B(CCN, receive), B(CCN, send), B(VCN, receive), and B(TSN, receive) |
| Scheduling | B(FLMN, schedule), B(FON, schedule), B(TSN, schedule), B(CCN, schedule), B(VCN, schedule), B(TSN, schedule), and B(Globa, schedule) |
| Task executing | B(FLMN, sample), B(FLMN, filter), B(FON, sample), B(TSN, sample), B(FON, filter), B(TSN, filter), B(CCN, calculate), B(VCN, effect), and B(TSN, effect) |

The simulation platform of the NCLM system is described by AADL in Open Source AADL Tool Environment [42]. The architecture of the NCLM system is shown in Figure 10. Seven AADL processes are used to model the six nodes in the realized control system and the plant. The six nodes are connected by CAN network. The plant is linked with the sensor nodes and actuator nodes through the IO bus.
specially, the tasks of the temperature control are dispatched into the idle slots of the level control.

In this simulation system, the temperature, liquid level and flows, and the opening degrees of the valves are selected as the fault features for fault detection. A lot of monitors in different nodes are preset for probing these variables, which are described in the fault detection tasks. And the suspected monitoring data and fault messages are clustered in the last detector before reporting to the system-level detector. Eighteen detectors are designed for the fault detection. The periodic scheduling table is shown in Figure 12. During each liquid level control period, interactive tasks are dispatched in pairs. When the tasks for temperature control are joined, the corresponding tasks appear in node CCN which are
highlighted by the red circles. The other tasks for other services are not considered in this table, which will be arranged in the idle slots.

6.3. Experimental Results. To verify the detections of the transient faults in NCLM, a random transient liquid level fault at the sample task in FLMN node is injected once. The fault injection time is displayed in the corresponding window for monitoring the node as shown in Figure 13(a). At the fourth level control simulation period, a level fault occurred by which the level was changed to 1.152. The detector in this node immediately discovered this fault after the sample task by parameter match, and the change rate of the level is used to check the fault in the filter task, by which the fault will be exposed and the detection timing conformed to the sequence of the task schedule.

In the upper right of the CCN window in Figure 13(a), the opening degree of valve $V_C$ changes from 0.107871 to 0.867871, so a decision fault can be designated by the unreasonable variation. According to the schedule table, with the results in plant window and the level records in Figure 13(c), the fault reflected in the plant can be identified in the fifth simulation period obviously.

The flow for level control, $f_C$, has a large variation as well in the fifth simulation period as shown in Figure 13(d). And the effect persists for a few periods and causes the valve $V_C$ to close, which provides the condition for fault detection. Because the temperature is controlled in the fourth level control period, the temperature in normal operation or with fault injection is less different as shown in Figure 13(b).

In all, the results show that the injected transient fault can be detected hierarchically by the proposed fault detection approach. The transient faults can propagate into the sub-sequential behaviors that make the goal lost, and the detection through the FBS models can be carried out according to their effects in different hierarchies.

7. Conclusion and Future Works

The detection of transient faults in NCSs depends on their effects on the system. FBS models can structurally represent the entire knowledge of NCSs, which are suitable for the detection of the transient faults. The structure models of NCSs consist of the resources in the system and their statuses and compositional relationships; the behavior models describe the possible potential operations as a series of EFSMs; and the function models reveal the goals of the realized system by the functions presented in the nodes and the subsystems which are parameterized by a specified functional state sets. Their mapping relationships show the feasible paths the fault propagates. Hierarchical transient fault detection employs parameter match and model based detection to identify the faults as many as possible. In the former the features are compared with their predefined scopes and in the latter diverse models are utilized to analyze the anomalies. The monitors and the detector inserted in the system increase the number of tasks, so task schedule in advance is exhibited. After validating the schedulability, AADL suites are used to describe the simulation platform of NCLM for the verification of the proposal. The results show that the transient faults can be exposed hierarchically.

Currently, the scheduling analysis of the bursted tasks can only be conducted offline and the proposal is only applied for the analysis of a simple control system. In the future, the online schedule analysis will be enhanced and the applications will be expanded to more complex industrial control systems.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.
Figure 13: (a) Display of the nodes, (b) temperature comparison in normal operation and with fault injection, (c) level comparison in normal operation and with fault injection, and (d) flow comparison in normal operation and with fault injection.
Acknowledgment

This work was supported by the Natural Science Foundation of China (no. 61074145 and no. 61272204).

References

[1] F. L. Lain, J. Moyle, and D. Tilburg, “Network design consideration for distributed control systems,” IEEE Transactions on Control Systems Technology, vol. 10, no. 2, pp. 297–307, 2002.

[2] F. Yang, H. Zhang, Z. Liu, and R. Li, “Delay-dependent resilient-robust stabilisation of uncertain networked control systems with variable sampling intervals,” International Journal of Systems Science, vol. 45, no. 3, pp. 497–508, 2014.

[3] M. H. Mottaghi and H. R. Zarendi, “DFTS: a dynamic fault-tolerant scheduling for real-time tasks in multicore processors,” Microprocessors and Microsystems, vol. 38, no. 1, pp. 88–97, 2014.

[4] X. Yu and J. Jiang, “Hybrid fault-tolerant flight control system design against partial actuator failures,” IEEE Transactions on Control Systems Technology, vol. 20, no. 4, pp. 871–886, 2012.

[5] C. Zhou, H. Chen, N. Xiong, X. Huang, and A. V. Vasilakos, “Model-driven development of reconfigurable protocol stack for networked control systems,” IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews, vol. 42, no. 6, pp. 1439–1453, 2012.

[6] R. A. Gupta and M.-Y. Chow, “Networked control system: overview and research trends,” IEEE Transactions on Industrial Electronics, vol. 57, no. 7, pp. 2527–2555, 2010.

[7] H. Lee and Y. Kim, “Fault-tolerant control scheme for satellite attitude control system,” IET Control Theory and Applications, vol. 4, no. 8, pp. 1436–1450, 2010.

[8] C. Wilwert, N. Navet, Y. Q. Song, F. Simonot-Lion, and R. Zurawski, “Design of automotive X-by-wire systems,” in The Industrial Communication Technology Handbook, pp. 1–29, CRC, Boca Raton, Fla, USA, 2005.

[9] D. B. Júnior, F. Vargas, M. B. Santos, I. C. Teixeira, and J. P. Teixeira, “Modeling and simulation of time domain faults in digital systems,” in Proceedings of the 10th IEEE International On-Line Testing Symposium (IOLTS ’04), pp. 5–10, Madeira Island, Portugal, July 2004.

[10] S. Singh, A. Kodali, K. Choi et al., “Dynamic multiple fault diagnosis: mathematical formulations and solution techniques,” IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, vol. 39, no. 1, pp. 160–176, 2009.

[11] J. Sosnowski, “Transient fault tolerance in digital systems,” IEEE Micro, vol. 14, no. 1, pp. 24–35, 1994.

[12] G. K. Saha, “Software based fault tolerance: a survey,” Ubiquity, vol. 9, no. 23, p. 1, 2008.

[13] P. Richardson, L. Sieh, and A. M. Elkateeb, “Fault-tolerant adaptive scheduling for embedded real-time systems,” IEEE Micro, vol. 21, no. 5, pp. 41–51, 2001.

[14] A.-A. Ahmadi, F. R. Salmasi, M. Noori-Manzar, and T. A. Najafabadi, “Speed sensorless and sensor-fault tolerant optimal PI regulator for networked DC motor system with unknown time-delay and packet dropout,” IEEE Transactions on Industrial Electronics, vol. 61, no. 2, pp. 708–717, 2014.

[15] D. X. Xie, G. Chen, and D. F. Zhang, “Guaranteed cost fault-tolerant control for uncertain networked control system with packet dropouts,” in Proceedings of the IEEE International Conference on Automation and Logistics (ICAL ’12), pp. 35–40, Zhengzhou, China, August 2012.

[16] P. Zhang, S. X. Ding, P. M. Frank, and M. Sader, “Fault detection of networked control systems with missing measurements,” in Proceedings of the 5th Asian Control Conference, pp. 1258–1263, Melbourne, Australia, July 2004.

[17] C. Pinello, L. P. Carloni, and A. L. Sangiovanni-Vincentelli, “Fault-tolerant distributed deployment of embedded control software,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 27, no. 5, pp. 906–919, 2008.

[18] A. Li and B. Hong, “Software implemented transient fault detection in space computer,” Aerospace Science and Technology, vol. 11, no. 2–3, pp. 245–252, 2007.

[19] P. Bernardi, L. M. B. Poehls, M. Grosso, and M. S. Reorda, “A hybrid approach for detection and correction of transient faults in SoC,” IEEE Transactions on Dependable and Secure Computing, vol. 7, no. 4, pp. 439–445, 2010.

[20] L. Chittaro, G. Guida, C. Tasso, and E. Toppano, “Functional and teleological knowledge in the multimodeling approach for reasoning about physical systems: a case study in diagnosis,” IEEE Transactions on Systems, Man and Cybernetics, vol. 23, no. 6, pp. 1718–1751, 1993.

[21] N. Chatt, A.-L. Gehin, B. Ould-Bouamama, and R. Merzouki, “Functional and behavior models for the supervision of an intelligent and autonomous system;” IEEE Transactions on Automation Science and Engineering, vol. 10, no. 2, pp. 431–445, 2013.

[22] M. Lambert, B. Riera, and G. Martel, “Application of functional analysis techniques to supervisory systems,” Reliability Engineering & System Safety, vol. 64, no. 2, pp. 209–224, 1999.

[23] M. Bozanno, A. Cimatti, J.-P. Katoen, V. Y. Nguyen, T. Noll, and M. Roveri, “Safety, dependability and performance analysis of extended AADL models,” Computer Journal, vol. 54, no. 5, pp. 754–775, 2011.

[24] G. C. Walsh and H. Ye, “Scheduling of networked control systems,” IEEE Control Systems Magazine, vol. 21, no. 1, pp. 57–65, 2001.

[25] H. Cha, E. M. Rudnick, J. H. Patel, R. K. Iyer, and G. S. Choi, “A gate-level simulation environment for alpha-particle-induced transient faults,” IEEE Transactions on Computers, vol. 45, no. 11, pp. 1248–1256, 1996.

[26] A. A. Cárdenas, S. Amin, Z.-S. Lin, Y.-L. Huang, C.-Y. Huang, and S. Sastry, “Attacks against process control systems: risk assessment, detection, and response,” in Proceedings of the 6th International Symposium on Information, Computer and Communications Security (ASIACCS ’11), pp. 355–366, Hong Kong, March 2011.

[27] E. Normand, “Single-event effects in avionics,” IEEE Transactions on Nuclear Science, vol. 43, no. 2, pp. 461–474, 1996.

[28] K. Vaidyanathan and K. S. Trivedi, “A comprehensive model for software rejuvenation,” IEEE Transactions on Dependable and Secure Computing, vol. 2, no. 2, pp. 124–137, 2005.

[29] R. Chillarege, S. Biyani, and J. Rosenthal, “Measurement of failure rate in widely distributed software,” in Proceedings of the 25th International Symposium on Fault-Tolerant Computing, pp. 424–433, Pasadena, Calif, USA, June 1995.

[30] S. W. Kwak, B. J. Choi, and B. K. Kim, “An optimal checkpointing-strategy for real-time control systems under transient faults,” IEEE Transactions on Reliability, vol. 50, no. 3, pp. 293–301, 2001.

[31] A. Aviˇzienis, J.-C. Laprie, B. Randell, and C. Landwehr, “Basic concepts and taxonomy of dependable and secure computing,” IEEE Transactions on Dependable and Secure Computing, vol. 1, no. 1, pp. 11–33, 2004.
[32] M. P. E. Heimdahl and N. G. Leveson, "Completeness and consistency in hierarchical state-based requirements," IEEE Transactions on Software Engineering, vol. 22, no. 6, pp. 363–377, 1996.

[33] G. Barrett and S. LaFortune, "Bisimulation, the supervisory control problem and strong model matching for finite state machines," Discrete Event Dynamic Systems, vol. 8, no. 4, pp. 377–429, 1998.

[34] A. A. Andrews, J. Offutt, and R. T. Alexander, "Testing Web applications by modeling with FSMs," Software and Systems Modeling, vol. 4, no. 3, pp. 326–345, 2005.

[35] V. S. Alagar and K. Periyasamy, "Extended finite state machine," in Specification of Software Systems, pp. 105–128, Springer, London, UK, 2011.

[36] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: a survey," ACM Computing Surveys, vol. 41, no. 3, article 15, 2009.

[37] H. Wang, K. Bi, T. Luo, and H. Luo, "Periodic messages scheduling in control network," in Proceedings of the 2nd International Conference on Electric Information and Control Engineering, pp. 1113–1116, IEEE Computer Society, 2012.

[38] A. Shah and K. Kotecha, "ACO based dynamic scheduling algorithm for real-time multiprocessor systems," International Journal of Grid and High Performance Computing, vol. 3, no. 3, pp. 20–30, 2011.

[39] S. Y. Amdani and M. S. Ali, "An overview of real-time disk scheduling algorithms," International Journal on Emerging Technologies, vol. 2, no. 1, pp. 126–130, 2011.

[40] R. Sharma and Nitin, "Performance evaluation of new joint EDF-RM scheduling algorithm for real time distributed system," Journal of Engineering, vol. 2014, Article ID 485361, 13 pages, 2014.

[41] Y. A. Cengel, M. A. Boles, and M. Kanoglu, Thermodynamics: An Engineering Approach, vol. 5, McGraw-Hill, New York, NY, USA, 2011.

[42] J. Hugues, B. Zallia, L. Pautet, and F. Kordon, "From the prototype to the final embedded system using the Ocarina AADL tool suite," ACM Transactions on Embedded Computing Systems, vol. 7, no. 4, article 42, 2008.