Adaptive Message Scheduling for a Real-Time Distributed Control System

Tiberiu S. Letia, Radu Miron, Mihai Hulea

Technical University of Cluj-Napoca, 15, C. Daicoviciu, RO 400020 Romania (Tel:+40-264-401432; e-mail:tiberiu.letia@aut.utcluj.ro; radu.miron@aut.utcluj.ro; mihai.hulea@aut.utcluj.ro)

Abstract: The behavior of a real-time distributed control system is strongly influenced by the variable message delays when the communication system is highly loaded. The distributed control and information components need information from other components before some specified deadlines; otherwise their performances are seriously diminished. The changing of the transmission demands requires the adaptation of the communication system such that the software real-time requirements are fulfilled. On-line fuzzy logic schedulers for ordering the messages are used for this purpose. The tuning of fuzzy membership functions parameters is performed offline by a genetic algorithm.

Keywords: real-time control, adaptive real-time communication, Petri net, fuzzy logic scheduling, genetic algorithms, distributed control.

1. INTRODUCTION

Many real-time distributed control applications include network transmission resources that are variable and highly loaded. The study refers to the problem of a distributed control system with the hardware architecture presented in Figure 1. The application is conceived to solve the problem of urban vehicle traffic control (Letia et al., 2009). The system is composed of a blade server, a number of Local Processing Units (LPUs) assigned to each street intersection and communication nodes (denoted by CN) connecting them.

The communication links are heterogeneous (wire and wireless with different speeds) and are variably loaded. This leads to variable transmission durations between software components with different priority levels. The currently used commercial routers and access points have no facilities for message ordering. The variable length and variable number of messages collected in the CN buffers are sent in first in first out order. The LPUs have no possibility to influence the transmission order once a message is sent into the communication system. This part of the distributed control system has no facilities to control the message transmission taking into account the message urgency and importance. The messages have to be sent into the communication system in an order that improves the fulfilling of the global real-time requirements.

The software structure of a distributed control system is shown in Figure 2 on a task diagram. This consists of:

- Task 1 that implements the control functions that are periodically executed
- Task 2 that periodically reads the sensor devices
- Task 3 that informs the Vehicle Traffic Control Center (VTCC) and neighbors about the local state and events
- Receiver 1, ..., Receiver n, Receiver S playing the role of receivers from neighbor LPU or VTCC
- Message Scheduler orders the messages to be sent
- Sender that sends the messages to the addressee

The addressee of a message is one of the receiver tasks (Receiver 1, ..., Receiver S) of another LPU.

An operator using the VTCC can change the tasks parameters and ask different LPUs to periodically send different information. Task 1, Task 2 and Task 3 periodically send and receive information to and from other LPUs. The tasks with given priorities use information provided by tasks (with different priorities) implemented on different LPUs. The tasks, that have messages to be sent, load them into the OB buffer (Output Buffer). The message Scheduler uses the information provided by receivers and stored in the CB buffer (Control Buffer) and the information from the Message Parameter Pool to order the messages into the SB buffer (Send Buffer). The sending of the messages from one LPU to another is...
variably delayed and depends on the current communication system load. When the messages expected by a task from other LPU tasks are delayed and miss the deadlines, the system works with diminished of estimated performances.

2. RELATED WORK

In systems where several activities are executed on different nodes and cooperate via message passing, an increasingly important concept is the Quality-of-Service (QoS) that is a system performance metric from the application point-of-view (Pedreiras and Almeida, 2003). The QoS is a function of communication parameters such as the rates of the message streams. The flexible time-triggered method uses an asymmetric synchronous architecture that comprises one master and several slave nodes.

Ecker et al. (2003) present a model that is useful for the development of resource allocation algorithms for distributed real-time systems that operate in dynamic environments. This model, besides the dynamic environments, includes utility and service levels which provide a means of graceful degradation in resource constrained situations.

A measurement-analytic approach for estimating the overflow probability is an important measure of the quality of service (QoS) at a given multiplexing point in the network. The dominant time scale, which corresponds to the most probable time scale over which overflow occurs, can be used (Eung and Shroff, 2003). The complex systems are based on a multibuffered structure with general work-conserving disciplines.

Sensors embedded technologies often involve adaptable middleware. Such systems must adapt themselves in order to fulfill real-time requirements (de Freitas et al., 2008). The changing scenarios require variable durations of the services performed locally. These services implement the adaptability.

Many formal methods and analysis techniques have been proposed for specifying and verifying the real-time systems. Juan et al. (2001) use the delay time Petri nets for reduction and analysis of such systems. To describe the stochastic behavior and the preemption, Bucci et al. (2005) introduce the stochastic preemptive time Petri nets.

Metzner et al. (2005) approached the distributed real-time systems by checking the satisfiability of real-time requirements. For simplicity reasons, they use only one communication media for message changing.

Due to the fact that distributed real-time systems are concurrent and network dependent, new methodologies for testing and debugging under stress conditions such as heavy loaded and intense network traffic are searched (Garousi, 2008). An UML 2.0 model is used for distributed systems under test. Different types of arrival patterns for real-time events are considered (Garousi et al., 2008). Genetic algorithms are used to find the test requirements which lead to maximum possible traffic-aware stress.

Gonzales et al. (1998) handle the noise-affected applications maintaining the completeness and consistency conditions for learning rules. According to Cordon et al. (2004) fuzzy systems present interest due to learning and adaptation capabilities. Genetic fuzzy logic systems hybridise the approximate reasoning method of fuzzy systems with the learning features of evolutionary algorithms.

The standard fuzzy systems suffer the “curse of dimensionality” (Wang et al., 2006). Hierarchical fuzzy systems have emerged as an effective alternative to overcome this.

3. COMMUNICATION SYSTEM MODEL

Solving the message scheduling problem requires sound knowledge of the communication system behavior. The Delay Time Petri Net model presented in Figure 3 describes the communication between the local processing units LPU1,
LPU2 and LPU3. The brackets attached to transitions specify the delays of their executions. When only one parameter is included, the delay is fixed. The brackets with two parameters specify the lower and upper limits of the variable delays. The delays’ values for different communication environments are presented in Table 1, where \( er, lr, ea \) and \( la \) correspond to the earliest released, latest released, earliest acknowledged and latest acknowledged, respectively. The time unit used in the model is the millisecond. The legend specifies the transmission environment.

The transition \( T1 \) models the periodic sending actions of messages of LPU1. The message generated with a period \( x \) set by an traffic operator is introduced into the sender buffer represented by place \( P1 \). Some errors are possible to occur and this is represented by transition \( ERR1 \). The transition \( T3 \) corresponds to the sending of the message to the communication node with the destination LPU2 linked to the same CN as LPU1. The transition has a variable delay between \( er \) and \( lr \) milliseconds. When an error occurs, the transition \( ERR2 \) is executed. The message enters the LPU2 buffer. The transition \( T4 \) models the transfer of the message to a receiver buffer and the starting of the confirmation procedure. If an error occurs, the transition \( ERR3 \) is executed. Otherwise, the transition \( T6 \) sends a confirmation message to the CN buffer (place \( P6 \)). The confirmation message arrives finally to the sender LPU buffer (place transition \( P7 \)). It is taken (transition \( T8 \)) by the LPU1 receiver.

The upper part of the figure models the case when the message sent by LPU1 has to cross another CN before reaching the destination LPU3. The messages sent through the communication system have no priorities. The models of the relations LPU-CN-LPU and LPU-CN-CN-LPU were used to simulate the behavior of the entire communication system. The implementation of the model was used to verify and test the proposed message scheduling solution.

4. ADAPTIVE COMMUNICATION MECHANISM

Each message sent through the communication system contains the information: sender, addressee, message index and content.

The Message Scheduler uses the information from the Message Parameter Pool to order the messages from the buffer OB into buffer SB.

The sequence of the messages sent from an expeditor task to an addressee task composes a session. The Message Parameter Pool contains for each session the following information:

- addressee period and initial offset
- addressee priority (denoted by \( p \))
- transmission duration (denoted by \( d \)) of the last sent message from the source LPU to destination LPU
- deadline missing (denoted by \( d \)) during a moving specified time span

Each message sent to a destination receiver contains the deadline added by Message Scheduler. The destination receiver verifies if the message arrived before the deadline. It confirms to the corresponding scheduler from the source LPU the receiving of the message and informs it if the message arrives before or after deadline. The Scheduler updates the Message Parameter Pool with the session parameters referring to transmission durations and deadline missing.

Due to the operator’s actions and variable behavior of the environment, different parts of the communication system are variably loaded. This leads to variable transmission durations of messages. Even if this happens, the distributed real-time system meets the requirement that the messages are available to addressee tasks when they are reactivated.

The expeditor tasks have different periods than the addressee tasks, as it is represented in Figure 4.

For example, an expeditor Task \( i \) sends information messages \( \{ms_1, \ldots, ms_i\} \) to the addressee Task \( j \). The first one has the period \( T_i \), and the second one \( T_j \). As can be seen in Figure 4, the relative deadlines are variable. The transmission durations are variable, too. Message Scheduler can calculate the variable value of the laxity (denoted here by \( l \)) for each message. The messages have assigned the priorities of the addressee. The Message Scheduler has to order the messages from the SB buffer such that they arrive to destination in time. If not all the messages can arrive in time, the messages with lower priorities are those that miss the deadlines.

4.1 Fuzzification

In order to solve the real-time communication problem of the message scheduling, fuzzy logic rules are used. The fuzzy logic scheduler has as input variables the crisp values \( d \) and \( l \). They are fuzzified using the well known membership functions represented in Figure 5. The values \( d \) and \( l \) are converted into variables with the linguistic values denoted by \( d \) and \( l \) respectively. The fuzzified variables take values in the domain \( \{L, M, H\} \) (i. e., low, medium and high). They have the membership degrees \( \mu_d \) belonging to different domains. The values \( a, b, c \) have to be calculated offline.

4.2 Fuzzy Logic Rules

The scheduling is based on the inference using some rules having the following form:

\[
\text{IF } (p \text{ is } X) \text{ AND } (l \text{ is } Y) \text{ AND } (d \text{ is } Z) \text{ THEN } (v \text{ is } V)\]

(1)

where \( X, Y, Z \) and \( V \) take value from the set \( \{L, M, H\} \).
The tasks priorities (denoted by $p$) are considered to belong to the same set \{L, M, H\} also. The fuzzy logic rules described above are implemented using the Table 2, Table 3 and Table 4. They provide the fuzzy logic values for the output $v$.

<table>
<thead>
<tr>
<th>Table 2. Fuzzy logic rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p$ is $H$</td>
</tr>
<tr>
<td>L</td>
</tr>
<tr>
<td>M</td>
</tr>
<tr>
<td>L</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 3. Fuzzy logic rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p$ is $M$</td>
</tr>
<tr>
<td>L</td>
</tr>
<tr>
<td>M</td>
</tr>
<tr>
<td>L</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 4. Fuzzy logic rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p$ is $L$</td>
</tr>
<tr>
<td>L</td>
</tr>
<tr>
<td>M</td>
</tr>
<tr>
<td>L</td>
</tr>
</tbody>
</table>

### 4.3 Defuzzification

Due to the fact that more than one rule is activated for the same set of input values, the defuzzification process involves the use of the strength of each rule. The strength of the rules $r_j$ can be calculated with the following formula:

$$\text{Strength}(r_j) = \min \{\deg(p), \deg(d), \deg(l)\}$$

where $\deg(x)$ is the membership degree of the variable $x$ to the fuzzy logic domain.

Figure 6 depicts the defuzzification of a scheduling value $v$. For a set of crisp values ($l$, $d$) two fuzzy logic rules were activated. One provides that $v$ is $L$ with the strength $s_1$. The other provides that $v$ is $M$ with the strength $s_2$. The logical value $v$ has to be transformed into crisp scheduling value. The centre of gravity for singleton method is used for that purpose. The domains of the logical variables L, M and H are isosceles triangles. In this case, the abscissas under gravity centers $l_i$, $m_i$ and $h_i$, are respectively fixed. The crisp values provided by defuzzification are given by:

$$v = \frac{\sum_{j=1}^{n} v_i \cdot s_j}{\sum_{j=1}^{n} s_j}$$

where $s_j$ is the strength of the rule that gives the membership degree to the logical value $v_i$ and $n$ is the number of fuzzy logic rule activated for the same set ($l$, $d$).

Message Scheduler has the highest possible local priority and it is activated when a task creates a new message or a message is sent through the communication system. It performs online the following algorithm:

**Message Scheduler algorithm:**

**Input:** Output Buffer, MessageParameters;
**Output:** Send Buffer;
**Move and remove all the messages from Send Buffer to Output Buffer**

**FOR** all the messages of the OutBuffer **DO**
- Calculate the deadline and laxity;
- Fuzzify the $l$ and $d$ values;
- Use the inference to get the values $v$;
- Defuzzify to get the crisp value $v$;
- Order the messages from Output Buffer into Send Buffer using the $v$ values;

### 5. OPTIMIZATION OF THE MESSAGE SCHEDULING

#### 5.1 Performance Evaluation

The conversion of the crisp values $d$ and $l$ into fuzzy logic values requires the values $a$, $b$, $c$ and $\text{max}$ of the membership function to be set offline (see Figure 5). The tuning of membership functions is done by using a genetic algorithm. It has to minimize the fitness (performance) function:

$$J = \sum_{k=0}^{\text{TimeHorizon}} \sum_{i} \left( d_{i,k}^1 \cdot c_1 + d_{i,k}^2 \cdot c_2 + d_{i,k}^3 \cdot c_3 \right)$$

Fig. 5. Membership functions.

Fig. 6. Defuzzification.
where \( d_{jk} \) is the number of the messages with the priority \( j \) that missed their deadlines during a sampling step \( k \) sent by the tasks from LPU \( i \). \( J \) counts all the messages that missed their deadlines in all the LPUs during the time horizon interval. \( c_1, c_2 \) and \( c_3 \) are weighting coefficients of the system performance evaluation.

According to Michalewicz (1992), the utilization of GAs requires chromosomes coding the individuals’ features, genetic operators that act on chromosomes, and evaluation of the individuals.

5.2. Chromosome

Figure 7 represents the individual structure above and the chromosome below. A gene denoted by \( x_i \) \((i=1,\ldots,8)\) is coded by a real number.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>max</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
<td>max</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>max</td>
</tr>
<tr>
<td>( x_1 )</td>
<td>( x_2 )</td>
<td>( x_3 )</td>
<td>( x_4 )</td>
<td>( x_5 )</td>
<td>( x_6 )</td>
<td>( x_7 )</td>
<td>( x_8 )</td>
</tr>
</tbody>
</table>

Fig. 7. The structure of an individual and a chromosome.

The conversion of the chromosome values to the membership function parameters for the calculation of the \( d \) value of an individual is given by the following formulae:

\[
\begin{align*}
    a &= x_1 \\
    b &= x_1 + x_2 \\
    c &= x_1 + x_2 + x_3 \\
    \text{max} &= x_1 + x_2 + x_3 + x_4 \\
\end{align*}
\]

Similar relations are used for the laxity value \( l \).

5.3 Genetic Operators

Since the genes are not binary encoded, simple genetic operators cannot be used. Instead of these the following operators are used:

- **Mutations**
  - Randomly select a gene in the chromosome and randomly assign to it a real value.

- **Crossover**
  - Choose two chromosomes from the population. Randomly select a crossover point (between 1 and 7) to cut the (offsprings) by joining to each initial chromosome the opposite part of the other (previous chromosome).

The genetic algorithm using the mentioned chromosomes and genetic operators uses a model of distributed control system and the proposed communication system. The communication system is simulated on a model that implements a delay time Petri net (Figure 3) constructed for the entire system. The LPU task diagram is used as a model of the local control and information components.

6. TEST AND RESULTS

In order to test the system the following conditions have been considered. Each LPU sends messages with a period \( x \) that belongs to the domain \([300, 1000]\) ms.

Each task of a LPU (Task 1, Task 2 and Task 3) generates messages with the period \( T_{i,j} \) given by:

\[
T_{i,j} = k_{i,j} \cdot x
\]

where \( i \) denotes the LPU number \((1\text{--}9)\) and \( j \) is the corresponding task \((1, 2 \text{ or } 3)\). Task 1 periodically sends messages (with dimension between 1 and 5 KB) that contain control information to specified addresses. Task 2 periodically sends messages with the length of around 50 KB. Task 3 periodically captures images (with the lengths around 1MB) from the corresponding intersection and sends them to the Vehicle Traffic Control Center. The simulations were performed for \( 2\cdot10^6 \cdot x \) durations. The chosen values of \( k_{i,j} \) are presented in Table 5.

<table>
<thead>
<tr>
<th>( k_{i,j} ) matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPU</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>Task 1</td>
</tr>
<tr>
<td>Task 2</td>
</tr>
<tr>
<td>Task 3</td>
</tr>
</tbody>
</table>

The chart presented in Figure 8 shows that when the message rate is low, the scheduler does not contribute a lot to the missed deadline improvement. When the traffic is increased, the fuzzy logic scheduler obtains significantly better results as compared to first in first out (FIFO) message sending policy.

![Fig. 8. Simulation results](image)

The tests show that the number of missed deadlines for higher priority messages is close to those of lower priority messages.

6. CONCLUSIONS

The fuzzy logic rules were determined so that the fuzzy logic scheduler performance is improved. They transform the user’s preference priorities into the scheduler current priorities. More priority levels can be used when more tasks have to be scheduled. The behaviour of the fuzzy logic scheduler is not influenced by the operating system features.
More priority levels are required in order to increase the scheduler sensitivity.

REFERENCES


