Copyright © 2002 IFAC
15th Triennial World Congress, Barcelona, Spain

REAL-TIME CONTROL AND MONITORING IN A CONTAINER TERMINAL [1]

J.L. Poza[(1)], R. Simarro[(2)], M.A. de la Fuente[(3)], J. Simó[(1)], A. Crespo[(1)]

[(1)] Departamento de Informática de Sistemas y Computadores (D.I.S.C.A.)
[(2)] Departamento de Ingeniería de Sistemas y Automática (D.I.S.A.)
[(3)] Departamento de Informática de Sistemas y Computación (D.S.I.C.)
Universidad Politécnica de Valencia, Spain
[(1)] {jopolu , jsimo, alfons}@disca.upv.es
[(2)] rausifer@upvnet.upv.es
[(3)] mafuente@dsic.upv.es

Abstract: Management, control, monitoring and detecting errors in distributed industrial systems is strongly dependent on the physical and functional topology. This paper describes a solution to this problem by monitoring the event channel instead of monitoring the nodes. Multi-agent architecture is an appropriate framework to this kind of systems because it optimizes the complete yard management and uses an event-oriented channel to support the communication between agents. *Copyright© 2001 IFAC*

Keywords: Distributed computer control systems, transportation control, monitored control systems.

1. INTRODUCTION

The operations performed in a containers terminal involve one of the most complex environment within the transport industry. Its automatization makes it necessary to develop a series of independent systems, but decisions made in each one of the systems may directly affect to the operation of the others (fig 1.).

Due to the complexity of the individual systems, it is very difficult to create a single application able to integrate all requirements. Therefore, it is more advisable to approach each task independently. Distributing tasks makes the relationship among them an important issue to optimize the global operation of the terminal.

Multi-agent paradigm (Wooldridge and Jennings, 1995) is suited to approach the design and later development of a set of flexible, adaptable, versatile and robust applications which global goal is to effectively manage a container terminal.

Fig. 1. Functional division in a container terminal.

Communication between agents, and the knowledge about system states and the important events are a fundamental issue for solving errors and keeping the system control.

Monitoring is essential to obtain the required information about the operation of distributed systems in order to make management decisions and control the behavior.

[1] This work has been supported by grant 1FD97-2158-C04-03. CICYT Spanish Government.

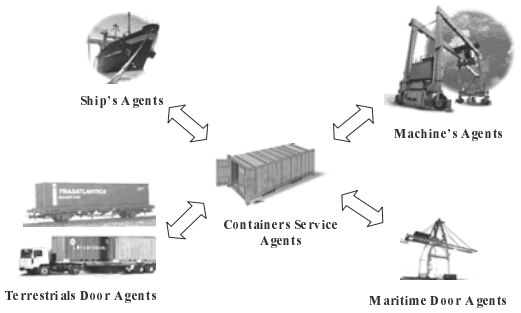


Fig. 2. System agents distribution.

The problem of monitoring a container terminal has not been dealt with depth until the moment. This paper describes a method for monitoring the system by watching the communication event channel and the messages prepared for the purpose. The section 2, introduces the agents system that must be monitoring, section 3 describes the monitoring model used, section 4 presents the procedure to generate the plans and to control the temporary impact in the global system. Conclusions are presented in section 5.

2. AGENT SYSTEM ARCHITECTURE.

2.1 Agents typology.

To design the architecture, the system has been divided according to its main tasks. In this way, a different kind of agent for each one of the main tasks has been developed (fig. 2.). The communication between agents is done by means of a reliable and secure communication channel (De La Fuente et al, 2001).

Containers Input/Output System. Containers input / output system consists of two kind of gate agents: marine and ground. The marine-gate agents have to manage containers load movements and containers unload movements corresponding to the ships that berth in the terminal. Ground-gate agents must inform the appropriate container service agent of the arrival of new containers and the arrival of trucks to remove containers from the terminal.

Containers Movement System. It consists of machinery agents and ship agents. Machinery agents obtain the most accurate sequence for the container movement to/from its correct position in the yard optimizing the use of the machinery resources within the terminal, and minimising empty movements and trucks delay time in the yard.

Ship agents face a scheduling problem where a set of resources (the cranes) must be assigned to the different operations (container load/unload) establishing a resource use time (container load/unload time). Their objective is to decrease the cranes inactivity time, to maximize the cranes using time, and to reduce the container load/unload time.

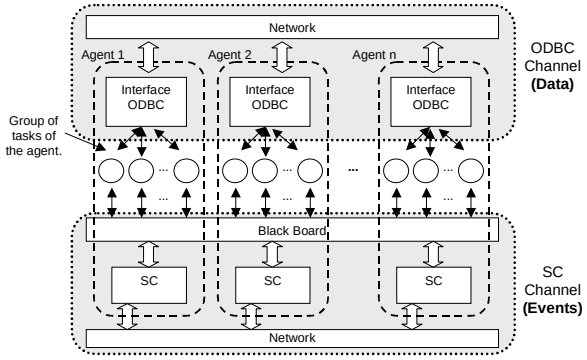


Fig. 3. The two system communications channels.

Containers Service Agents. The Terminal has been divided according to the different services (shipping routes involving a number of ships). Each service has assigned some specific stacking ranges. The main goal of this kind of agent is to determine the appropriate allocation for the arriving containers in the terminal from a specific service (allocation problem)

2.2 Communications system between agents.

The communication between agents and the corresponding data warehouse is made by means of a number of content-oriented communication channels. For this reason, two types of channels can be distinguished in the system: event-oriented and data-oriented channels. (fig.3).

The event-oriented channels includes several different communication system. For example, a serial port and a radio channel is used to communicate with the machinery agents, whereas a socket connection is used to communicate events from the database server. Using different heterogeneous systems makes it necessary to adapt the internal messages at the concrete communication system and to make use of different communication protocols. The three main components of the agent communication system are exposed next.

Event Channel. This level offers an event-oriented communication to broadcast the events produced in the terminal. This task is made through a communication server called “Servidor de Comunicaciones” (SC), based on sharing data through a distributed blackboard (Penny, 1989). The SC system has been developed and tested previously in the context of distributed systems and mobile robots (Posadas, et al., 2000). In order to communicate the SC with the systems that cannot be connected directly specific gateways to each different system are needed.

Gateways. In order to communicate several heterogeneous systems machines, gateways that adapt both communication channels are required. In normal system operation, the gateways must open, maintain and close the communication channels by means of TCP, UDP sockets or by the system serial COM ports that communicate with the radio systems.

Data Channel. This channel is the operating system support to the communication with the database, because the agents need to obtain data stored in different platforms. The communication between applications and databases is usually made by means of operating system ODBC drivers.

3. MONITORING MODEL

Monitoring can be defined as the dynamic collection process, interpretation and presentation of information concerning objects or software processes under scrutiny. It is needed for various purposes such as debugging, testing, program visualization and animation. It may also be used for general management activities which have a more permanent and continuous nature (performance management, configuration management, fault management, security management, etc.) (Sloman 87). In this case the behaviour of the system is observed and monitoring information is gathered. This information is used to make management decisions and perform the appropriate control actions on the system

3.1 Event control.

In order to know the system state and to handle the sequence of messages that are sent, the event channel must be controlled, because each message between agents is an event itself. Events captured can be made at three levels depending on its importance: state, errors and alarms. The system events that are not important can be considered as a fourth level with minimum priority in the monitoring.

In the capture state level, the monitor is connected to all the data shared by the applications and, after a previous filtrate, the activity in each node of the system is visualized.

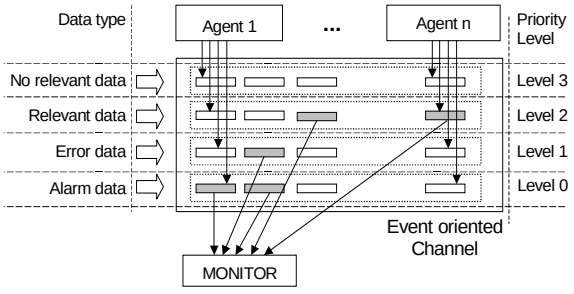


Fig. 4. Agents connection and the communication channel monitor.

The state visualization can be done by constant sampling or event-based. When visualization is made by sampling, a copy of the current situation in the communication channel can be obtained, which is not the habitual method. When visualization is event-based, a sequential registry of the channel activity is obtained, and thus the messages route through different nodes of the system can be followed. The state visualization is important to take a control of normal system activity, to verify the corresponding changes of state and similar tasks. Nevertheless, it adds a not desirable load level in the monitoring channel, although it has been stated that its effect is almost null on the normal operation.

At non-critical error verification level, error-oriented events in the communication channel are monitored. Distributed applications inform of errors, such as a message timeout, parse errors, message replication or similar. This level of monitoring is used to verify communication coherence.

Alarms are errors detected by applications and are significant for the global system operation. Errors as the absence of a basic node, a gateway breakout, or critical situations as a possible accident in the yard justifies this high-priority channel.

3.2 Message sequence control.

From any source, if an event happens, it follows a route of channels that communicate the applications. Each element in the system can be seen as an interconnected node. Connections among nodes are based on the route the message follows.

When the route followed by a message is being monitored it can be abstracted as a sub-graph of the system graph (fig. 5.). To assure coherence in the sequence of events produced by an action when processing the message a graph analysis tool is used.

Time spent by an event travelling trough the system is parameterized with the equation (1), where T_{event} is the total time that an event remains in the system, taking into account differences in the time measurements depending on the transmission media used. For each event, different machines (p), different communications channels (q) and different applications (r) can be involved in the calculation of the total time.

$$T_{event} = \sum_{i=1}^{p} TM_i + \sum_{i=1}^{q} TC_i + \sum_{i=1}^{r} TA_i \qquad (1)$$

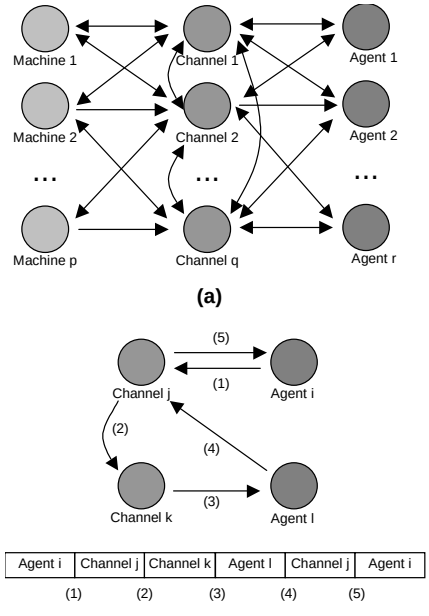


Fig. 5. Generic node graph of the system (a) and node sub-graph of a specific message and his message sequence (b).

Mathematical methods using patterns of event sequences can be used to calculate the time of any sequence of messages. Another method is to accumulate the individual times of each message in the sequence.

4. TRACE GENERATION.

A trace is an actual description of the system temporary evolution. The trace generation has generally been executed by means of libraries or dedicated units, just as it is presented in (Ohlenroth, 1996). These units should be combined with the program code or application to monitor and can be made by means of a connection to the communication channel or the system hardware. Anyway, the trace will maintain the corresponding coherence and homogeneity in its creation.

4.1 Generation of monitoring information.

Monitoring a distributed system implies generating the state log system in a concrete moment (snapshot) or the events log or a system event sequence (Mansouri-Samari and Sloman, 1992). System state monitoring implies that each node sends information to the monitoring agent (Magee, et al. 1989), which can be executed periodically or under demand. Event monitoring is a more complex task which consists of three phases: location of event detection, time of event detection and event report format (Mansouri-Samari and Sloman, 1992). The previous levels are detailed following.

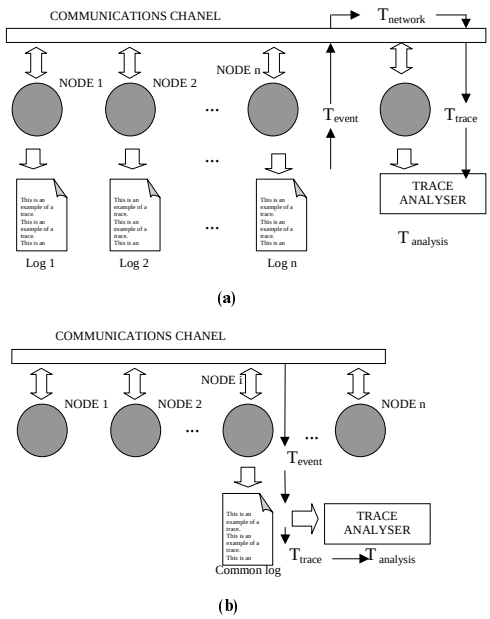


Fig. 6. Distributed node monitoring (a) and communications channel monitoring (b).

Location of event detection. There are two ways of detecting events in the distributed system. One is to detect individually on the part of each node the important events, to store this information and to send it to the node that executed the analysis (fig. 6.a). This approach has several problems. The main one is the temporary overload of the nodes due to the necessity of generating the traces. The other big problem is the overload of communication channel due to the necessity of transporting tracing among nodes through the communication channel. The time consumed to generate a simple event is obtained by the equation (2). T_{trace} is obtained with the control time method described in the section 4.2. $T_{network}$ is the time of crossing a TCP channel, which is very studied and depends on the amount of data and the network components. $T_{analysis}$ is the time consumed in the trace analysis and depends on the algorithms used and it is not the proposal of this study, although currently this algorithms are being developed.

$$T = T_{event} + T_{trace} + T_{network} + T_{analysis} \qquad (2)$$

As the total trace is distributed in n nodes, the total time to monitor the system is obtained by the equation (3).

$$T = \sum_{i=1}^{n} (T_{trace} + T_{network}) + T_{analysis} \qquad (3)$$

The alternative to event monitoring in the nodes is to locate events in the same communication channel (fig. 6.b). Accordingly, some advantages on the nodes monitoring method are obtained. Nodes overload is avoided, because these should execute

additional work, only in the marked of messages or the time control. Also, the communications channel overload is decreased since the traces does not need to go through the channel. The time consumed by this method is obtained from the equation (4). For each value of *n*, the time is lower that the one obtained by monitoring the distributed method.

$$T = T_{event} + T_{trace} + T_{analysis} \quad (4)$$

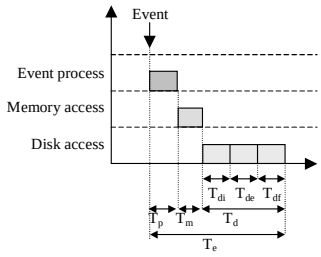
Time Of Event Detection. In order to characterize temporarily the time consumed in the route from a message through the system, the Time Firewall method (Kopetz, 1998) is used by each application maintaining a registry of the time consumed by a message while it remains in the application.

When the message arrives, the application compares its Time Field with the corresponding internal table where temporary characteristics of the message are specified. Based on this comparison, the message will be dealt accordingly. When the message leaves the application, the time that has remained in the application is added to the Time Field. A table to maintain the time intervals exists for every node. By means of this table, the communications state can be temporarily characterized at any moment.

This temporary characterization is very interesting because it allows to know whether the system is tuned and works in a synchronous way any moment. In the event of arriving messages with a high value time field, it is possible to make an inspection in the route of nodes towards the origin of the bottleneck and take measures to solve it (Poza, et al., 2000).

Event Report Format. The information a trace file contains makes it important or not. Standard XML has been used as a trace file in the developed system. XML documents are also known as self-describing. That is, each document contains the set of rules which data must be conformed to. Because any set of rules can be reused in another document, other authors can easily create the same class of document, if necessary. XML as trace format language has many advantages: it is not necessary to parse the message when the application does not take part in the monitoring, only the applications taking part in the monitoring will treat the corresponding tag. Another advantage is that applications are backwards and forewards compatible with the message, because a change in the message format does not make it necessary to change the applications.

The internal format of the message has the “trace” tag to store the trace information about this event. The TTP attribute is used to maintain the time control of the message with the TTP protocol and the “path” attribute is used to store progressively the sequence of nodes that are crossed by the message.



T_e Total time of process of the event.
T_p Time of processing of the event.
T_m Time of storage in memory of the trace.
T_d Time of storage in disc of the trace
T_{di} Time of initial disc (opening and position in the file).
T_{de} Access time to the stored disc for of trace.
T_{df} Time of final disc (it closes of the file).

Fig. 7. Tasks envolved in the trace generation for a simple event.

4.2 Trace Generation.

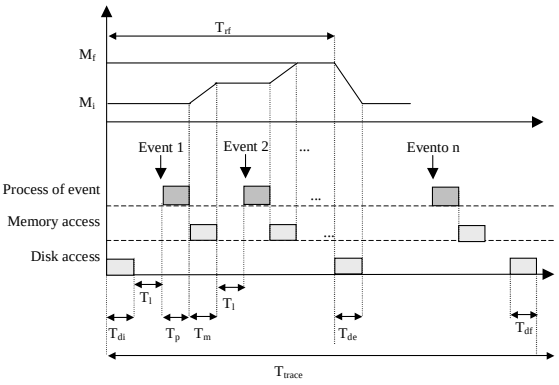
The trace generation is made in the monitoring node. Tasks implied in an event that must be observed are described following (fig. 7.). Event management: it is composed by the associated computes and the trace characters chain creation. Memory access: it consists of processor memory storage of chain trace generated previously. Access to disk: in this task, the characters chain corresponding to the monitoring are stored in a disk file (trace). The time control is handled by monitoring software connected to the corresponding node. Currently, a dedicated class implemented in MFC 4.0 is being used.

Times involved on the trace generation of a efficient model are present in (5). Considering a number *n* of events are required to obtain an efficient trace, calculating the time of a trace of a path message implies to consider the number of times that it is executed. The total time of a trace is on the equation (6).

$$T_{trace} = t_e + t_m + t_{di} + t_{de} + t_{df} \quad (5)$$

$$T_{trace} = n(t_e + t_m + t_{di} + t_{de} + t_{df}) \quad (6)$$

The hybrid monitoring is an interesting model that can implemented. This system joins the philosophies described in the last two sections, that is, it must avoid the excessive memory use for the log chain and it must not slow down the normal execution of the system caused by writing the file in disk (fig. 8.). This diagram implies creating a messages buffer that it is stored in the file when it reaches a certain size. In this way we make sure that maximum log chain is not oversize and time in opening up, consenting or closing is not spent. Besides, the log file is updated. The times involved on the trace generation of a efficient model are present in (7).



T_{trace} Total time of process all the event.
T_p Time of processing a simple event.
T_m Time of storage in memory of the trace.
T_d Time of storage in disc of the trace
T_{di} Time of initial disc (opening and position in the file).
T_{de} Access time to the stored disc for of trace.
T_{df} Time of final disc (it closes of the file).
T_{rf} Total time of process n events.
T_l Free time to another task.
M_l Minimal memory consumption.
M_l Maximal memory consumption

Fig. 8. Efficient model to generate traces.

$$T_{trace} = \sum_{i=1}^{n} (t_{event} + t_p + t_m) + n(t_d) \qquad (7)$$

In this model, two parameters determine the monitoring efficiency: the first is the size of log chain buffer, and the second is the file update frequency. The balance of each parameter can determine the optimum way to generate a log, without the conditions of real time system being interfered.

5. CONCLUSIONS

If the method used to monitor the system is the distributed node monitoring method (fig 6.a), the analysis consists of several phases as merging traces, validation of monitoring information, database updating, combination of monitoring information, filtering of monitoring information and finally the analysis of monitoring information (Mansouri-Samari and Sloman, 1992). As our system uses the communication channel monitoring method (fig 6.b) it is only necessary to do the filtering phase and the analysis phase.

Table 1. The four different modes of monitoring.

Mode	Number of SC data.	Interval of events.
Critical events	1	1
Snapshots	n	1
Message Path	1	n
System Evolution	n	n

The system allows a flexible events monitoring, because it is possible to watch out for different control levels (table 1). With this method, a way to monitor a distributed system only by means of an event-oriented communication channel has been obtained.

The system overload that monitoring introduces is minimum, because the connection of the monitoring agents is carried out in a separate machine. This implies that the rest of the agents do not perceive the effect of the monitoring node. Any agent can switch from a normal work mode to a monitoring work mode, by reading or writing the trace field in the XML message.

Nowadays, this monitoring system is being used to control the sequences of messages generated in a multi-agent system implemented to automatise the container terminal of the Port of Valencia, Spain.

REFERENCES

De La Fuente Anuarbe, M. A., J.M. Figueroa García, V.J. Botti Navarro and C. Ricolfe Viala. (2001). Arquitectura Multi-Agente Para La Gestión De Operaciones Terrestres En Terminales De Contendores. In: *XXII Jornadas de Automática*. Barcelona, Spain.

Magee, J., J. Kramer, and M. Sloman (1989). Constructing Distributed Systems in Conic. In: *IEEE Transactions on Software Engineering*, 15(**6**):663--675.

Kopetz H. (1998). The Time-Triggered Model of Computation. 0-8186-9212-X/98 *IEEE*.

Mansouri-Samari, M. and M. Sloman (1992). "Monitoring Distributed Systems (A Survey)", In: *Imperial College Research Report No. DOC92/23*..

Ohlenroth, M. (1996). Application Oriented Monitoring. In: *TUCZ / RA-TR-96-09*.

Penny, H. (1989). Blackboard Architectures and Applications. Edited by V. Jagannathan, Rajendra Dodhiawala, Lawrence S. Baum.

Posadas, J.L., P. Pérez, J.E. Simó, G. Benet and F. Blanes (2000). Communications Structure for Sensor Fusion in Distributed Real Time Systems. In: *6th IFAC Workshop on Algorithms and Architectures for Real-Time Control*. *AARTC'2000*. Mallorca.

Poza, J.L., J.L. Posadas, J.E. Simó and A. Crespo, (2001). Data And Event Management In A Maritime Terminal Of Containers. In: *IFAC Conference on New Technologies for Computer Control*. *NTCC'2001*. Hong Kong.

Sloman, M. (1987). Distributed Systems Management. In: *Imperial College Research Report, DOC 87/6*.

Wooldridge, M. and N.R. Jennings (1995). Intelligent Agents --- Theories, Architectures, and Languages. In: *Lecture Notes in Artificial Intelligence* vol. 890.