MODEL-BASED ANALYSIS OF A WINDMILL COMMUNICATION SYSTEM

Simon Tjell
Department of Computer Science
University of Aarhus
Denmark

Abstract This paper presents the experiences obtained from modeling and analyzing a real-world application of distributed embedded computing. The modeling language Coloured Petri Nets (CPN) has been applied to analyze the properties of a communication system in a windmill, which enables a group of embedded computers to share a group of variables. A CPN-based model of the system is used to analyze certain real-time properties of the system.

Keywords: Real-time systems, communication protocol, Coloured Petri Nets, performance analysis

1. INTRODUCTION

This paper is based on a real life product development problem from the Danish windmill manufacturer Vestas Wind Systems [1] and proposes a solution to the problem. The proposed solution is analyzed by use of a CPN-based model of the system. The paper summarizes parts of a project [7], which has been running for a period of one year. A modern windmill from Vestas is controlled and monitored by a complex application consisting of a group of inter-connected control components. Each control component implements parts of the control algorithms, which cooperate to control and monitor the operation of the windmill through a collection of physical sensors and actuators. The components are executed in a distributed system of embedded computers connected by a network bus. The connection between the components is established by shared access to a group of variables. Basically, a variable represents one of three values: A measurement from a sensor, the output value for an actuator or an intermediate calculation between two components. The correctness of the output-variables from a component is highly dependent of the freshness and consistency of the input-variables of that component. Some of the output-variables are used to control actuators that adjust the wings and other physical parts of the mill. In that way, the correctness of the output-

Please use the following format when citing this chapter:

Tjell, S., 2006, in IFIP International Federation for Information Processing, Volume 225, From Model-Driven Design to Resource Management for Distributed Embedded Systems, eds. B. Kleinjohann, Kleinjohann L., Machado R., Pereira C., Thiagarajan P.S., (Boston: Springer), pp. 245–254.

variables has a direct influence of the degree of wear and tear of the machinery over time. This makes it important to optimize the method for sharing the variables in order to maximize the lifetime of the mill - and this task is the main aim of this paper and the project it describes. A design for the communication system and a model-based analysis of the design are provided. The following sections of the paper are structured like this: Firstly the existing software environment is introduced. This is followed by an identification of problems in the existing design of the communication system. The identification is followed by a proposal for an alternative design. The proposed redesign has been modeled, simulated and analyzed. This process is described in the last section of the paper.

2. THE DAO APPLICATION FRAMEWORK

Distributed and Active Objects (DAO) is a proprietary framework for developing windmill control applications. The framework is based on the Active Object design pattern [5] with the addition of distributed capabilities and dynamic attachment mechanisms. The following list gives a description of the key elements of the control application based on the DAO framework. The description defines the terminology for the remaining sections of this paper. Please refer to Figure 1 when reading the list.

1 Node: A node is an embedded computer. Each node has a kernel, which executes a number of components. A typical system consists of up to five nodes.

2 Kernel: The kernel has the responsibility of periodically activating the components on its own node. The activation period for each component is determined by a scheduling registration. The kernel continously runs through a cycle of operations.

3 Component: A component is a software object, which complies with a specified interface. The interface contains an activation method, through which the component is periodically activated by the kernel. The components contain all control algorithms used in the control application.

4 Variable: A variable has a type and a value. The value can be altered and monitored by the components. Each variable is represented in one original instance and a number of copies. The copies for a variable exist on nodes with components attached to that given variable. A number of components can establish dynamic attachments to each variable for reading and/or writing (one component at a time) the value of that variable. Variables exist in two forms on a node: Kernel variables (the originals) and component variables (copies on which the components

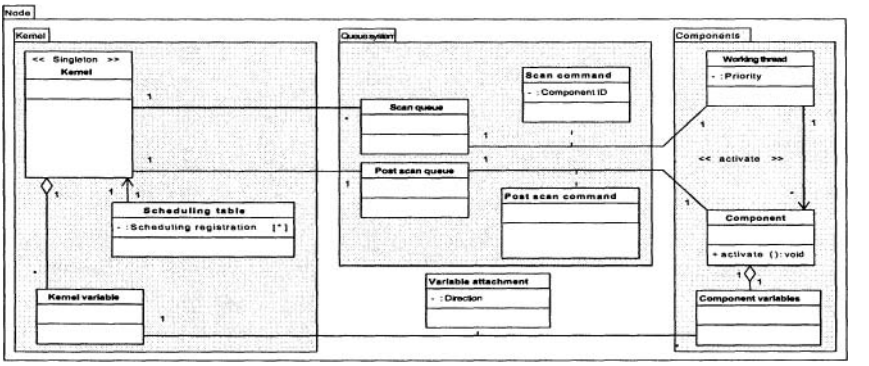

Figure 1. The static structure of the DAO-framework.

operate). The kernel has the responsibility of copying values between the two representations of variables.

5 Communication component: The communication component (CC) is a specialized component that exists in one instance in every node. It differs from the regular components by being able to communicate with its peer CCs through a network interface. The communication is performed in order to maintain variable consistency through-out the nodes. When the CC is activated, it performs two main tasks: It generates and sends update messages containing values of the kernel variables to which components on other nodes are attached and it receives the update messages from other nodes. One message holds values for a number of variables.

6 Queues: The scan queue is used when the kernel activates a component. The kernel does this by placing a scan command containing the ID of the component to be activated in one of the scan queues. The choice of scan queue determines the priority of the activation since commands from each scan queue are consumed by a specific threads. Each thread in the thread pool has an individual priority. When an activated component terminates its activation, this is signaled to the kernel with a post scan command that is placed in the post scan queue. This queue is shared by all working threads. When the kernel receives a post scan command

3. THE PROBLEM

This section gives a description of the existing communication protocol used for communication between the nodes. The description is supplemented by a characterization of a number of problems with this protocol. This descrip-

tion is followed by a proposal for a replacement communication protocol. The communication between the nodes is performed by the communication components in order to keep shared variable values updated. These components have access to a communication interface through which they are able to exchange UDP-datagrams [2] across the network. The CCs are periodically activated by the kernel. During one activation cycle are number of messages are sent and received.

The proposal of a new communication protocol for the DAO-framework is relevant, because problems are being experienced with the existing CC. In its existing design, the communication component uses a simple protocol, in which the CCs send out messages with variable values when variable values are changed - i.e. triggered by the event of changing values. The messages are sent unreliably using multicast through the UDP-protocol with no detection of lost messages. The UDP-protocol features a CRC-mechanism [2], which makes the receiver of a message able to discard the message, if its contents have been altered during the transmission due to electrical noise. This is the only sort of error detection. The nature of the existing design of the CC causes two critical problems:

1 If a message is lost during transmission this is not detected by neither the sender nor the receiver. Messages are sent when a state changes and when this information is lost, it results in the view of the state becoming inconsistent between the sender and receiver(s) of the message. The period of inconsistency continues until a new message is successfully exchanged. This happens the next time the state is changed.

2 If a large number of variables are altered within a short period of time, this will cause a large number of messages to be sent and received in that period. This can cause the CCs in the DAO-nodes to consume too much time within a given activation cycle. This can cause a skew in the time of activation for the other components on a node, which can result in erroneous output from the control algorithms implemented in those components.

4. PROPOSAL FOR AN ALTERNATIVE DESIGN

It has been decided to base the proposal of an alternative design of the communication protocol on the principle of Soft State signaling protocols [2]. The discussion leading to this decision can be found in [7]. The proposed communication protocol is based on a generic Soft State principle [2]. Soft State messaging is a variant of protocols within the family of signaling protocols. This family of protocols spans within two generically defined poles; Soft State and Hard State protocols. Signaling protocols are applied in a wide range of applications. Common to those applications and the specific application in

relation to DAO is the need to maintain a consistent view of a shared state - where the state could be a routing table, a list of shared files or a set of variables. Soft State protocols differ from Hard State protocols by the fact that they rely on the exchange of messages using best-effort-semantics as opposed to reliably exchanging messages. The difference is observable in the way, in which the distributed views of the state are updated across the network. In both cases messages are sent from the sender, which is where the changing of the state occurs to a receiver (or a group of receivers), where the state is observed. Also, in both cases, the sender is the initiator of the message exchange. The difference here lies in which event triggers the sending of the message with the changed state from the sender to the receiver. In the case of the Hard State protocols, the triggering event is the change itself - i.e. when the sender detects a change in its local view of the state, this causes a message with the updated state to be sent to the receiver. On the other hand, in the case of the Soft State protocols, the triggering event is a temporal event caused by a timer running out. Each state is associated with an update timer that causes a periodic event trigger.
Based on the properties of the proposed communication protocol, it is considered a sane choice for replacing the existing protocol, because the proposal is suited for addressing the two main problems with the existing protocol (the two issues refer to the two problems described earlier):

1 All messages are periodically retransmitted. This resolves the problem of potentially long periods of inconsistency in the case of lost messages.

2 The periods for sending messages are planned offline. This makes it possible to predict the maximum amount of messages being exchanged within an activation cycle and thereby predict processing overloads.

The original attachment mechanism with which attachments between components and remote or local variables are registered is adopted and applied in combination with the new protocol design. Each variable is assigned with an update timer that triggers the generation of periodic update messages for that variable. Based on knowledge of the properties of the variables, it is possible to define a method for grouping the variables based on their frequency of modification. In this way, each variable belongs to a certain group in which all variables share the same update rate, which is used for initializing the update timers and thereby determine when to send update messages. Five such groups are identified. Futhermore, the probabilistic distribution of variables in these groups is known based on experience from the existing system. This means that it can be estimated that each group contains a certain percentage of the variables. This knowledge is applied when modeling the application and the modification of the variables. While the modification rate differs between the

variables the swiftness of communicating changes to variable values is equally important for all variables in order to optimize consistency.

5. MODELING AND ANALYZING THE PROPOSAL

After having chosen an appropriate communication protocol, time comes to developing the model of the DAO-system with the proposed protocol integrated. This section describes the simulation process. An example of the analyzed data is given. For the full set of analysis results, please refer to [7]. The model is developed using Coloured Petri Nets (CPN) [3], which is a formally specified modeling language with a graphical representation. CPN-models specify the structural and behavioral aspects of (distributed) systems in which the state is represented by places holding tokens. Places are connected by transitions that are able to move tokens between the places of a net and thereby modify the state of the places and the model in general. In the case of modeling DAO and the communication protocol, the tokens represent entities such as messages and variables. In the same way, the transitions represent actions such as sending messages and reading variables. Tokens are also used for representing more abstract aspects of the system; delays, timers, counters etc. Analogously, places are used to represent buffers containing message, tables of variables and so on.

CPN-models differ from traditional Petri Nets by the fact that the tokens have types and values - much like in standard programming languages. This makes it possible to distinguish the tokens and simulate tokens moving around between the places. Figure 2 shows a part of the model. Places are visualized by ellipses and transitions as rectangles. The model consists of a hierarchi of modules like this. This particular part of the model specifies the dynamic behavior of a DAO-node.

The dynamic behavior of each nodes is modeled by a sequence of repeated events, including (ordered for understandability):

1 Generation of update messages: the table of local variables is evaluated in order to find variables with expired update timers. The current values of these variables are collected in a new update messages and the timers are initialized. In each case, the table of attachments from external components is investigated - if no attachments are found, it is not necessary to include the variable in the update message. If no variables comply with the constraints, no update message will be generated.

2 Reception of incoming update messages: The node receives messages through the communication channel.

3 Handling of incoming update messages: When the messages have been received they are processed, which means that all local variables are

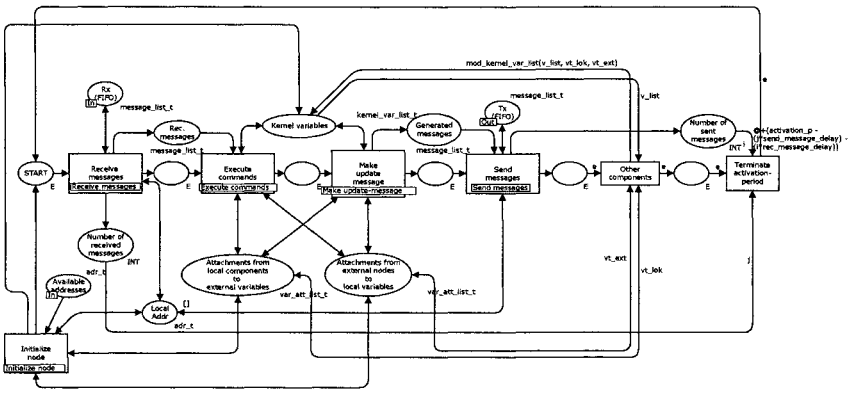

Figure 2. An example of a part of the CPN-model specifying the dynamics of a node.

updated with the values from the update messages. The values in the update messages are not necessarily different from the current values of the variables.

4 Transmission of outgoing update messages: If an update message has been generated, it is transfered to the communication channel.

5 Simulation of the effect of other components: The values of the local variables are changed periodically based on the knowledge of the modification rates for each group of variables. This is done to simulate the effect of the local components modifying the variables.

These events reflect the cycle of events that is handled by the kernel on each node.

One of the main advantages of models expressed in CPN is the fact that they are executable. This makes it possible to perform automated simulations while collecting measurements from the model. In the specific case, simulation has been applied to analyze a group of properties:

1 Consistency of the variables: The most basic purpose of the communication protocol is to maintain consistency within the group of shared variables. The level of consistency is measured by continously performing simple counting operations that lead to a number for the percentage of consistent variables. A variable is considered being consistent when all its instances in all nodes share the same value. Figure 3 shows the result of measuring the average consistency while varying the activation period of the CC and the number of nodes. As expected, the level of

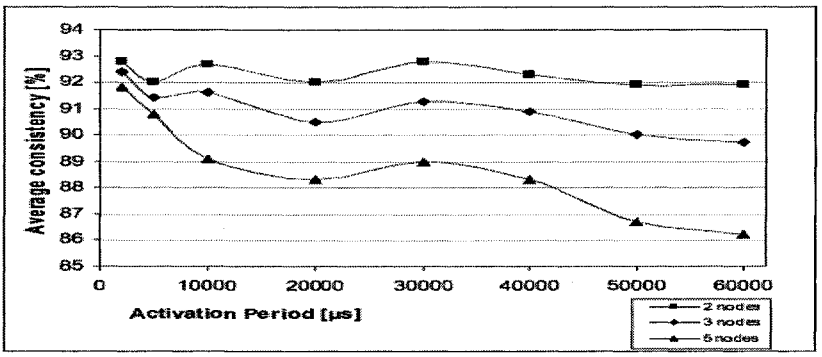

Figure 3. Measuring the average consistency as a function of activation period.

consistency decreases when the activation period is increased, since this will increase the magnitude of the delay that occurs between modification of a variable values and the next-coming transmission of a periodic update message. Furthermore, the consistency level decreases when the number of nodes is increased. This is caused by the fact that more nodes means more components, which results in more writing attachments for each variable.

2 Update delay: Another measure for the quality of the CC is the delay that occurs before all instances of a variable are updated when the value of one of the instances is modified. This is important because the quality of the output from the control algorithms depends directly on this property. The measurement of the delay is performed by supplying the tokens that represent update messages with a timestamp. The timestamp contains the time of modification for the involved variables and is used to calculate the delay when the update message is received by a node.

3 Resource consumption: Analyzing the amount of processing time being consumed by the CC on a node is relevant, because it directly defines the amount of time left for activating the other components on the same node. Measurements from the target platform have shown, that sending and receiving messages consumes significantly more time than any other task of the communication component. This - in combination with the fact that the CC is continously activated with a fixed period - makes it reasonable to estimate the amount of time being spent during each activation of the CC based on the number of messages being handled within that activation cycle. The amount of resource consumption is measured

in each activation cycle by counting the number of incoming and outgoing messages. The maximum consumption will occur when a CC on one node receives messages from all other nodes and at the same time has to send one multicast message itself. This means that the maximum can be calculated based on a collection of parameters: the variable activation cycle, the fixed amount of time used for handling messages and the variable number of nodes. For low activation frequencies, the measured level of resource consumption is similar to the calculated maximum since it will be necessary to send and receive messages in all activation cycles. This is not the case, when the activation frequency is increased. This increases the fraction of activation cycles where less or no messages are handled. On the other hand; when the activation period becomes shorter, each message being sent and received represents a larger fraction of the time in the activation cycle. Measurements on the actual embedded computers have shown that the action of sending and receiving messages is the major source of CPU-time consumption on the nodes. This makes it possible to calculate the percentage wise consumption of CPU-time of the CC by observing the number of messages being processed within a given activation cycle. This approximation is only feasible as long task of processing messages is significantly costlier in time than any other task being handled by the communication component.

Apart from these properties, the tolerance to packet loss is analyzed by varying the probability of losing packets in the communication channel connecting the nodes while observing the influence on the properties described above. The level of consistency and the magnitude of the update delays increase when more packets are lost. At the same time, the resource consumption decreases because the lost packets wont result in consumption of processing time on the receiving nodes.

The analysis illustrates how the configuration of the system is a tradeoff between consistency and resource consumption. With the Soft State protocol, the level of consistency depends on the frequency of sending the periodic update messages - and this frequency directly affects the amount of processing resources being consumed by the CC. In that context, this paper and [7] are ment to illustrate how the model-based analysis of the communication system is applied to gain knowledge of the parameters of this tradeoff. The protocol does not intend to guarantee full reliability at all times but rather a probabilistic reliability of adjustable quality.

6. RELATED WORK

This paper is focused around two main topics: the use of Soft State protocols for sharing states and the use of CPN-models for analyzing an industrial real-

time system by simulation. Related work exists within both topics. Firstly, [6] introduces a framework for understanding and discussing the properties of Soft State signaling protocols. The paper presents a Soft State transport protocol accompanied by a formal model, with which the protocol is analyzed by use of queuing theory. In [4] a Markov-model is specified for comparing the performance properties of generic Soft and Hard State protocols in combination with hybrid variants. Both of these works are focused on the performance and consistency issues of the protocols alone. This paper combines the protocol technicalities with the aspects of the application and the hardware platform in the simulation-based analysis. [8] develops CPN-models for analysing the RSVP-protocol, which is partly based on Soft State signaling.

7. SUMMARY

This paper has described the process of identifying the problem in question and has presented a proposal for its solution. This proposal has been modeled and the model has been used for simulation with the purpose of analyzing its real-time properties. CPN-models have proven to be a strong tool in this work and their executability have eased the process of understanding the model and thereby the existing system and the protocol that has been applied. The main weakness of this type of analysis is the lack of a strong method for validating the correctness of the model itself compared to what is being modeled.

REFERENCES

[1] Vestas wind systems. http://www.vestas.com.

[2] David D. Clark. The design philosophy of the DARPA internet protocols. In *SIGCOMM*, pages 106–114, Stanford, CA, August 1988. ACM.

[3] Kurt Jensen. Coloured petri nets: A high level language for system design and analysis. *Lecture Notes in Computer Science; Advances in Petri Nets 1990*, 483:342–416, 1991. NewsletterInfo: 39.

[4] Ping Ji, Zihui Ge, Jim Kurose, and Don Towsley. A comparison of hard-state and soft-state signaling protocols. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 251–262, New York, NY, USA, 2003. ACM Press.

[5] R. Greg Lavender and Douglas C. Schmidt. Active object: an object behavioral pattern for concurrent programming. pages 483–499, 1996.

[6] Suchitra Raman and Steven McCanne. A model, analysis, and protocol framework for soft state-based communication. In *SIGCOMM*, pages 15–25, 1999.

[7] Simon Tjell. Modeling and analysis of a communication protocol for windmills (danish only). Master's thesis, University of Aarhus, 2005. (http://daimi.au.dk/~tjell/thesis.pdf).

[8] M.E. Villapol and J. Billington. Modelling and initial analysis of the resource reservation protocol using coloured petri nets. In *Proc. Of the Workshop on Practical Use of High-Level Petri Nets, within the 21st International Conference on Applications and Theory of Petri Nets*, pages 91–110, 2000.