An Application of a Formal Approach for Distribution of Real-time Control

Thanikesavan Sivanthi[1], Ulrich Killat[1], Kishore Angrishi[2]

[1]Department of Communication Networks,
Hamburg University of Technology, Hamburg, Germany
{thanikesavan.sivanthi,killat}@tuhh.de

[2]Operating Facility Hamburg,
luftfahrtgeräte gauting gmbh, Hamburg, Germany
kishore.angrishi@lgg-gauting.de

Abstract — *A system which provides a set of control functions on time is referred as a real-time control system (RTCS). In a traditional RTCS, a set of tasks running in a central controller provide the required system control. The recent technological advances have made it possible to embed controllers with sufficient computing power directly in the end control devices. These devices also have a communication interface by means of which they can communicate with the other devices via a broadcast bus. These devices are referred to as intelligent nodes. A network of such intelligent nodes can be used to perform the same control functions as that of a centralized RTCS in a distributed manner. This paper discusses an application of a formal approach to distribute the control of a centralized RTCS over a set of intelligent nodes with an example.*

1 Introduction

Traditionally, a RTCS uses a centralized control paradigm. In such a system, a controller runs a set of tasks to control the dumb end control devices (e.g.) sensors, actuators. This system has many disadvantages like poor scalability and less robustness to failures. The end control devices today have an embedded microcontroller and a communication bus interface unit. These devices are referred to as intelligent nodes. A network of such intelligent nodes, which could perform the same control functions of a centralized RTCS is referred to as a decentralized real-time control system (DRTCS). In a DRTCS, each node performs local controls and when needed communicates with the other nodes via a broadcast bus to provide global controls. There are many advantages in such a system (e.g.) good scalability, good modularity and better robustness to failures. This led to a paradigm shift in the field of RTCS from centralized to decentralized control.

The interesting question is how one can distribute the control tasks of a centralized RTCS to a set of nodes. There are different approaches for distributing the control, many

of them are based on heuristics [1], [2], [3] and [4]. In this paper we describe a formal approach, which models the control software using task graphs and with the help of an integer linear program [5] distributes the tasks over the nodes of a DRTCS. The rest of the paper is organized as follows. Section 2 describes the formal approach for distributing the real-time control. Section 3 applies the formal approach to a potable water control system of an aircraft. The paper concludes with Section 4.

2 Formal approach for decentralization of real-time control

The control software of a RTCS consists of several tasks. The tasks may be periodic or aperiodic. A periodic task executes once every time period T, also called as the task period. An aperiodic task executes on occurrence of an event. The interarrival time between any two consecutive arrivals of an event varies widely. Each task is a sequential execution of elementary computing functions or modules (e.g.) a simple control task consists of sense, process and actuate modules which perform the sensing, processing and actuating functions and which are executed in that order. There can be precedence constraints between two modules, if a module of a task depends on the data from another module of the same or different tasks. The decentralization of real-time control means devolving the control from a central controller to a set of intelligent nodes. This involves the distribution of the modules of the tasks over the nodes of a DRTCS. The distribution should be done in a manner such that all the tasks finish before their deadlines. The distribution should also consider the resource constraints namely, the maximum allowed utilization of each node and the bus. Hence the distribution problem could be seen as an allocation and scheduling problem with the following constraints.

Constraints for allocation:

1. Utilization of each node is less than or equal to the maximum allowed utilization of that node.
2. Utilization of the bus is less than or equal to the maximum allowed utilization of the bus.
3. A module which performs functions that are local to a node should be assigned to that node.
4. Incase of fault-tolerance, the redundant modules must not be assigned to the same node.

Constraints for scheduling:

1. The first module of a task can never be released before the task's release time and the last module of a task must finish before the task's deadline.
2. A precedence constrained module is scheduled only after the execution of all of its predecessor modules and their communications to this module.
3. A non-precedence constrained module can begin its execution anytime in a node but, its execution can never overlap with the execution of the other modules assigned to the same node.

In RTCS there can be many tasks, hence the distribution problem is inherently complex. But the complexity can be handled by considering the distribution for each operation mode of the system. This is because, in each operation mode there are only certain tasks which are activated and running. In our approach at first we identify the different modes of operation of the system. Then for each mode of operation we built an acyclic task graph [5], which consists of all tasks that are executed in that operation mode. In [5], we considered that all tasks are periodic, but in RTCS there are also aperiodic tasks. If the minimum interarrival time between two consecutive event arrivals T_i which trigger an aperiodic task is known then, the aperiodic task can be viewed as a periodic task with task period equal to T_i. Thus, all aperiodic tasks of a RTCS can be considered as periodic tasks with their deadlines equal to their periods. The vertices of the task graph represent the modules of all tasks in one hyperperiod. A hyperperiod is defined as the least common multiple of all task periods. The directed edges of the graph have a weight equal to the number of messages communicated between the modules. An edge represents the data dependency of a module on another module. No edge exists between modules, if there is no dependency between them. The task graph also details the attributes of all tasks. The attributes are the worst case execution times of the modules of the tasks, the tasks release times and their deadlines.

The integer linear program (ILP) discussed in [5] is used to allocate and schedule the tasks modules and their communications in the nodes and the bus respectively. The ILP has an objective function which minimizes the maximum of all differences between the completion times and the deadlines of all tasks. The allocation and resource constraints are formulated as constraints to the objective function. A feasible schedule exists only if the resulting objective function has a value less than or equal to zero (i.e.) only when all tasks finish before their deadlines, while satisfying all the allocation and scheduling constraints. The result of the ILP is a feasible global schedule, which is an allocation and scheduling plan for the modules of all tasks. The global schedule is derived only for one hyperperiod. This is because the schedule derived for one hyperperiod repeats for every subsequent intervals of length equal to one hyperperiod.

3 An application: Potable water control system

The existing potable water control system in an aircraft is a complex centralized RTCS, with more than fifteen devices controlled by a controller. The system performs various tasks mostly related to water quantity monitoring and management, system pressure monitoring and management. We have applied the approach described in Section 2 to distribute the control over the nodes of a prototype decentralized potable water control system. The nodes of the decentralized system are connected by means of a CAN bus. The prototype decentralized potable water control system shown in “Figure 1” has the following intelligent nodes.

1. Two level sensors
2. One panel indicator
3. One fill/drain valve
4. One drain valve

5. One depress valve
6. One pressure sensor
7. One air compressor

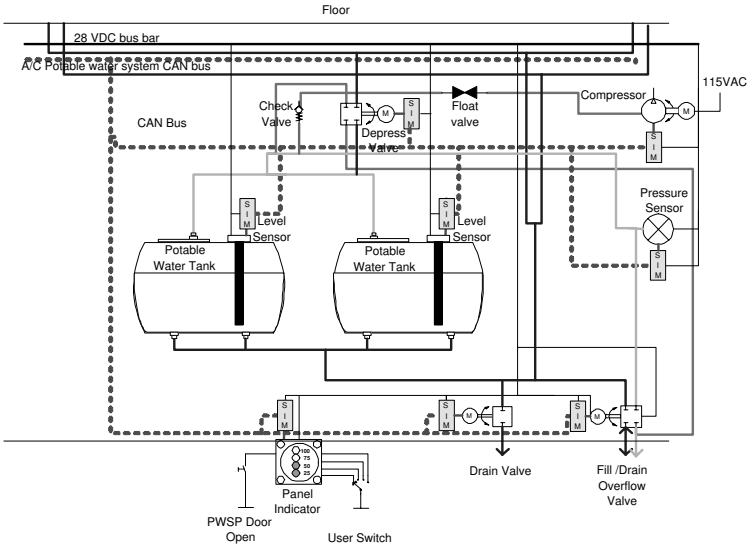

Figure 1: The prototype decentralized potable water control system

Each node has a maximum allowed utilization of 50% and the maximum allowed utilization of the bus is 60%. The system performs various control tasks, which can be grouped under five different operation modes. The tasks executed in each operation mode are identified, this includes the redundant tasks for fault-tolerance. The tasks and their communications are then represented in a task graph. The task graph for one such operation mode is shown in “Figure 2”.

The task graph consists of five tasks with a hyperperiod of 60 ms. There are three instances of Task1, Task2, Task 4 and Task 5 in one hyperperiod and Task 3 has a single instance. M2 and M3 are redundant modules for module M1. M5 and M6 are redundant modules for M4. M11 is a redundant module for module M9 and M10. M13 is a redundant module for M12. M15 is a redundant module for M14. The modules M7 and M8, M9, M10, M16, M17 perform functions that are local to Node 3, Node 1, Node 2, Node 4 and Node 5 respectively. These modules must be allocated to their corresponding nodes. The global schedule for the task graph derived using the ILP is shown in “Figure 3”. The global schedule defines the allocation and scheduling plan for all the task modules and their communications in the nodes and the bus respectively.

The ILP derives a TDMA based schedule for the communications in the bus. The time-triggered communication in CAN is achieved by implementing a time-triggered layer on the top of CAN. The node with the highest priority, also called the “monitor”, sends a clock synchronization message to synchronize all the application clock of the nodes. If this node fails, the node with the next highest priority will send the clock synchronization message. The operation mode determination is performed by the redundant modules, which are executed in three different nodes. The nodes which execute the operation mode

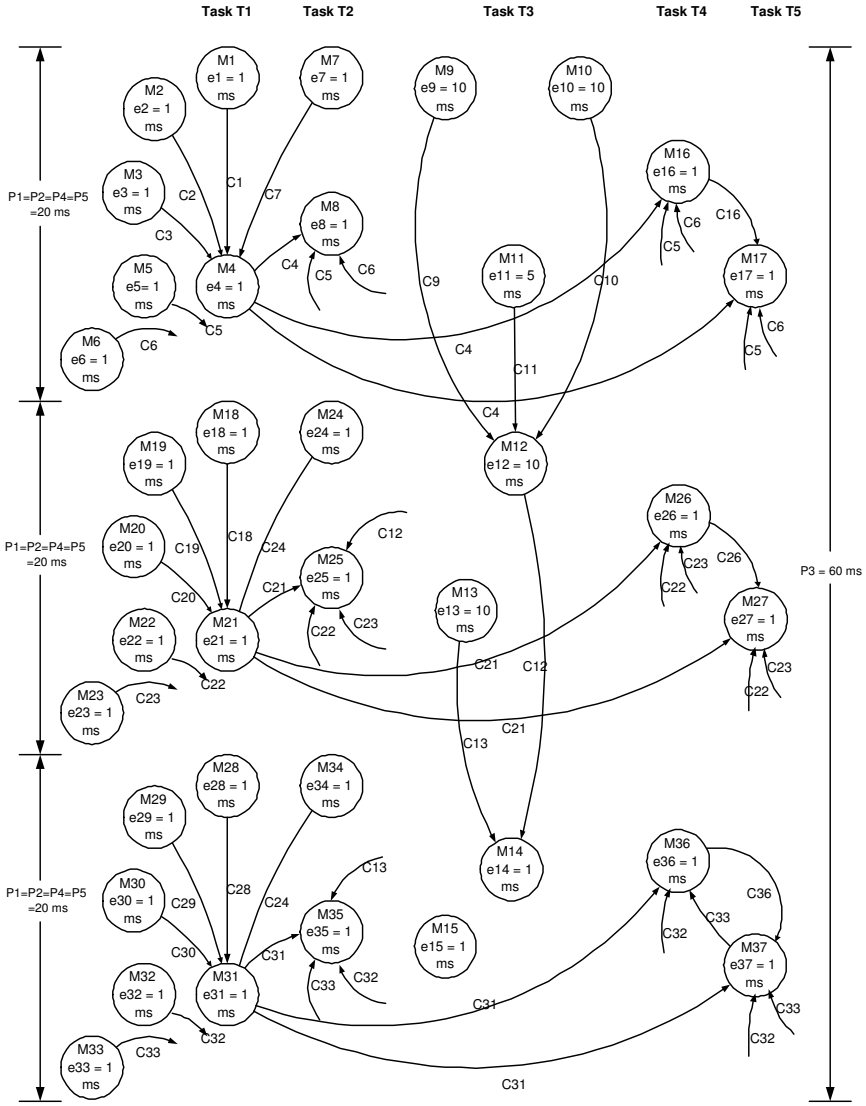

Figure 2: Task graph

determination modules, will send the operation mode change messages. The mode change latency of this system is greater than the hyperperiods of all operation modes hence, the nodes follow the global schedule for the new mode in the subsequent hyperperiod.

When building a decentralized control system, one need to ensure that all the system run-time control functionalities are fulfilled before the system is implemented. This can be achieved using a system simulation setup. We have built such a simulation setup for the prototype decentralized potable water system using Vector CANoe [6]. Vector CANoe is a powerful tool, which supports the entire development process for networked systems from planning to implementation. It offers special functions for all phases of a product cycle, (e.g.) model creation, simulation, functional testing, diagnostics, and analysis.

The global schedule derived using the ILP is used to distribute all system control tasks over the nodes of the prototype decentralized potable water system. This helps to refine the design to the level of the network nodes, which involves specifying the behavior of the

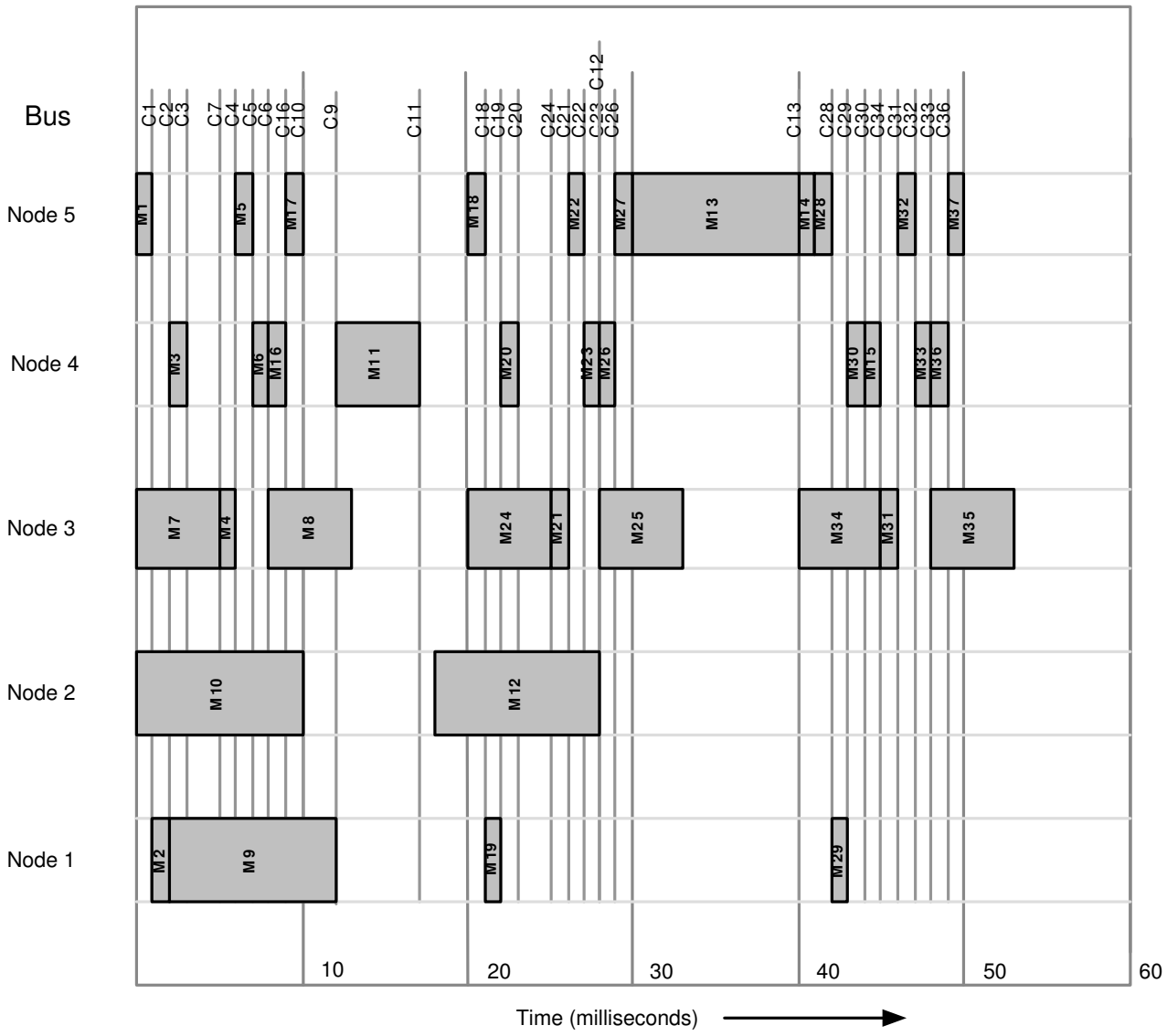

Figure 3: Global schedule

network nodes with regard to input and output variables and the messages to be received and transmitted. The physical process to be controlled is modeled using the environment input and output variables of Vector CANoe. The behavior of each network node is specified using the functions available in a C-like programming language called CAPL [6]. The test cases of a centralized potable water system derived from the requirement specification, is used to verify the compliance of the decentralized potable water system using simulation.

4 Conclusion

In this paper, we discussed a formal approach for distributing the control of a RTCS to a set of nodes. The control software is abstracted using task graphs and the distribution is carried out using an ILP. The complexity of the distribution is handled by finding an

allocation and scheduling plan for each operation mode of the system. The derived plan confirms to the task and resource constraints. The approach is applied to decentralize a potable water control system in an aircraft. A prototype of the decentralized potable water control system is built in Vector CANoe and the system control functionalities are verified by simulation.

References

[1] H. Barada, S.M. Sait, and N. Baig. Task matching and scheduling in heterogeneous systems using simulated evolution. *10th IEEE Heterogeneous Computing Workshop (HCW 2001)*, April 2001.

[2] C. Hsueh and K. Lin. Scheduling real-time systems with end-to-end timing constraints using the distributed pinwheel model. *IEEE Trans. Comput.*, January 2001.

[3] M. Lin, L. Karlsson, and L.T. Yang. Heuristic techniques: Scheduling partially ordered tasks in a multi-processor environment with tabu search and genetic algorithms. *ICPADS*, July 2000.

[4] S. Faucou, A.M. Deplanche, and J.P. Beauvais. Heuristic techniques for allocating and scheduling communicating periodic tasks in distributed real-time systems. *IEEE WFCS*, September 2000.

[5] T. Sivanthi and U. Killat. Global scheduling of periodic tasks in a decentralized real-time control system. *IEEE WFCS*, September 2004.

[6] *CAN open environment, Overview and Project Description*. Vector Informatik GmbH, Germany, 1996.