Chapter 2
Resource Allocation in Distributed Control and Embedded Systems

Traditionally, control design problems are decoupled from software design and implementation considerations. Such a separation allowed the control and computer science communities to focus on specific problems independently, and led to the development that we are familiar with nowadays. However, as explained in *Årzèn et al.* [13], this separation relies on the fact that these two fields use very simplified models of their interface with each other. In fact, control designers disregard the characteristics of the implementation and the available computational and communication resources. On the other hand, real-time designers see the control loop as a periodic task with a hard deadline, that have sometimes to fulfill data dependency constraints (especially when the sensing and actuation are distant). Recently, researchers from these two communities have shown that if more elaborate models are used, significant improvements in terms of implementation efficiency and quality of control may be achieved. This *integrative co-design approach* is central to this book.

This chapter is organized into two parts. The first part presents the state of the art of the real-time scheduling theory, focusing on the most used results in distributed control and embedded system (*DCES*) applications. We start by presenting real-time single-processor scheduling problems. Next, we focus on the problem of ensuring real-time networked communications. We emphasize different methods for managing the access concurrency having a determinant impact on the guarantee of deterministic real-time communications. Finally, an overview of the problem of guaranteeing end-to-end real-time constraints in distributed systems is given. In the second part, we present a state of the art of the new approaches, that are based on more elaborate models, and that take into account both the dynamic nature of the controlled systems as well as some characteristics of their implementation. Various problems and models were discussed in the literature. We propose a classification of these different approaches and illustrate the different problems and models that were addressed.

2.1 Real-Time Scheduling Theory

As mentioned above, the objectives of this section concern different tools and mechanisms which handle the concurrency in real-time systems at the processor level as

A. Çela et al., *Optimal Design of Distributed Control and Embedded Systems*, Communications and Control Engineering, DOI 10.1007/978-3-319-02729-6_2,

well as at the distributed communication architecture. We focus on the models and tools related to control applications without claiming to be exhaustive. Throughout this section, we give some references that may help for a deeper and better understanding of the models to be used in the particular applications we may be faced with.

2.1.1 Real-Time Single-Processor Scheduling

This subsection is devoted to real-time scheduling of multiple task on a single processor. The objective is to obtain a calculation model related to concurrent execution of a given number of tasks. For more details, please refer to two excellent books of *Buttazzo* [48] and *Cottet* [67]. In real-time processing systems [48, 67], the processor is a resource that is shared between various concurrent *tasks*. A *task* represents a sequence of instructions that are intended to be executed by the processor. The service that is delivered by a task may be performed several times during the lifetime of the application. For such reasons, a task may be *"instantiated"* several times in the form of *jobs* or *task instances*. Jobs or task instances represent the execution flow that corresponds to the effective execution of the task code.

2.1.1.1 Events Characterizing the Lifetime of a Job

A job or task instance is characterized by the following temporal parameters:

- *release time*: the time instant at which the scheduler is requested to execute the job, which have just become ready to run;
- *start time*: the time instant at which the job starts its execution;
- *preemption times*: time instants when the scheduler suspends the execution of the job on behalf of other jobs having a more important priority;
- *resumption times*: time instants at which the execution of the job is resumed after a preceding preemption;
- *completion time*: time instant at which the job finishes its execution;
- its *absolute deadline*: time instant before which the job should have terminated.

Figure 2.1 illustrates the single processor scheduling of three jobs j_1, j_2 and j_3. More precisely, in this figure:

- instants t_1, t_2 and t_3 represent the respective release time instants (depicted by up arrows) of jobs j_1, j_2 and j_3,
- instants t_1, t_2 and t_6 represent the respective start times of jobs j_1, j_2 and j_3,
- instants t_2 and t_4 represent respectively preemption and resumption times of job j_1,
- instants t_6, t_4 and t_7 represent the respective completion time instants of jobs j_1, j_2 and j_3,
- instants t_5 and t_8 represent the respective absolute deadlines (depicted by down arrows) of jobs j_1 and of j_2 and j_3.

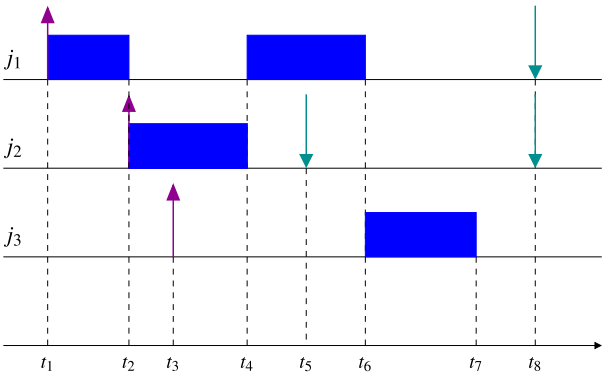

Fig. 2.1 Single-processor scheduling of three tasks

2.1.1.2 Task Model

A real-time task $\tau^{(i)}$ is characterized by:

- worst case execution time (*WCET*) $c^{(i)}$;
- activation law of its jobs: the jobs of a given task may be activated periodically with a period $T^{(i)}$, sporadically with a minimum inter-arrival time or aperiodically if no temporal constraints are imposed on the activation of its jobs;
- relative deadline $D^{(i)}$: the time interval between the release time of the job and its absolute deadline.

A real-time task is called *periodic*, *sporadic* or *aperiodic* according to the activation law of its jobs [48, 67].

2.1.1.3 Scheduling Algorithms Classification

This paragraph describes the commonly used terminology for classifying the existing scheduling algorithms [48, 67].

- *Preemptive/non-preemptive*: A scheduler is called *preemptive* if it is able to suspend a running task on behalf of other tasks that have more important priorities. It is called *non-preemptive* in the opposite case. Preemption is supported by the majority of real-time operating systems. In the opposite, the scheduling of packets in networks is always non-preemptive.
- *Off-line/on-line*: In *off-line* scheduling algorithms, the sequencing of the tasks to be executed is described at design time, as a *schedule* or *execution plan*. Consequently, the scheduler is simply a sequencer, that executes the different tasks according to the schedule that was pre-computed off-line. The execution order of the different tasks is then identical to that specified in the execution plan. In practice, the schedule is executed in a repetitive way. The period of repetition is called *major cycle* or *hyperperiod*. In general, the schedule describes the start

time instants of the different tasks instances, and possibly, their preemption and resumption time instants, which are expressed as a function of an elementary time unit, called *minor cycle* of the schedule. In the opposite, in *on-line* scheduling algorithms, the choice of what task to execute is determined at run-time by the scheduler. When activated, the scheduler performs a given processing in order to determine the next tasks to execute. In most cases, this processing amounts to the comparison of the priorities of the ready tasks. These priorities may be fixed in the case of *fixed-priority* scheduling algorithms or dynamic (i.e., adjustable at run-time) in the case of *dynamic scheduling algorithms*.

2.1.1.4 Schedulability Analysis

Real-time scheduling theory [48, 67] aims at providing the sufficient conditions (and preferably the necessary and sufficient conditions) guaranteeing that a task set (which is defined by a given model) will respect its real-time constraints (which are defined by the assigned deadlines). An important theoretical tool that it provides is the *schedulability analysis*. Schedulability analysis is performed off-line, in order to ensure that the scheduling of a task set (which satisfies a given model), using a given scheduling algorithm, ensures the respect of the tasks deadlines. In the sequel, we present the fundamental results that are related to the preemptive real-time scheduling of periodic tasks whose relative deadlines are equal to their periods, by fixed-priority and dynamic-priority scheduling algorithms.

Consider a task set containing $\mathcal{N}$ independent tasks $\tau = (\tau^{(1)}, \ldots, \tau^{(\mathcal{N})})$. Each task $\tau^{(i)}$ is characterized by its *worst-case execution time* (*WCET*) $c^{(i)}$, its period $T^{(i)}$ and its relative deadline $D^{(i)} = T^{(i)}$. The task set τ is said to be *schedulable* by a given scheduling algorithm if all the jobs of all the tasks forming τ finish their execution before their absolute deadlines.

- *Fixed-priority scheduling*: In this scheduling policy, a fixed priority $p^{(i)}$ is assigned to each task $\tau^{(i)}$. At each instant, the scheduler executes the ready task whose priority is the most important. Since preemption is authorized, if during the execution of a given task $\tau^{(i)}$, another task $\tau^{(j)}$ that has a more important priority becomes ready, then task $\tau^{(i)}$ is preempted and the processor is allocated to task $\tau^{(j)}$. In 1973, *Liu and Layland* [156] have shown that the optimal priority assignment is given by the *rate monotonic* (*RM*) rule (i.e., the smaller the period is, the higher is the assigned priority). Thus, the rate monotonic scheduling algorithm is a fixed-priority scheduling algorithm where priorities are assigned according to the rate monotonic rule. The optimality of a real-time scheduling algorithm was defined by *Liu and Layland* by its ability to schedule a given task set, which verifies a given task model, such that the predefined real-time constraints are met. A fixed-priority (resp. dynamic-priority) scheduling algorithm is optimal (for a given task model) in the sense that *any task set* (*satisfying the task model*) *that is not schedulable under this algorithm will not be schedulable under any other fixed-priority* (*resp. dynamic-priority*) *scheduling algorithm*. A sufficient

schedulability condition using *RM* is

$$\mathscr{U} = \sum_{i=1}^{\mathscr{N}} \frac{c^{(i)}}{T^{(i)}} \leq \mathscr{N}\left(2^{1/\mathscr{N}} - 1\right).$$

The necessary and sufficient schedulability condition by *RM* requires the analysis of the maximum response time $R^{(i)}$ of each task (i.e., the maximum among the response times of its jobs) (see, for instance, *Joseph and Pandya* [135]). The response time of a job is defined by the duration between its release time and its completion time. $R^{(i)}$ is given by:

$$R^{(i)} = c^{(i)} + \sum_{j \in hp(i)} \left\lceil \frac{R^{(i)}}{T^{(j)}} \right\rceil c^{(j)}.$$

where $hp(i)$ is the set of tasks which have priority over $\tau^{(i)}$. The task set τ is schedulable by *RM* if and only if $R^{(i)} \leq D^{(i)}$, for all $i \in \{1, \ldots, \mathscr{N}\}$.

- *Dynamic-priority scheduling*: In this scheduling policy, the priority $p^{(i)}$ that is assigned to the task $\tau^{(i)}$ may vary over time. *Liu and Layland* [156] proved that the optimal dynamic priority assignment policy consists in assigning the most important priority to the task that is the closest to its deadline. This priority assignment rule is called *Earliest Deadline First* (*EDF*). The necessary and sufficient schedulability condition under *EDF* is simpler than that established for *RM* and is given by:

$$\mathscr{U} = \sum_{i=1}^{\mathscr{N}} \frac{c^{(i)}}{T^{(i)}} \leq 1.$$

Real-time scheduling theory progressed substantially since the fundamental article of *Liu and Layland* [156], to take into account the problems involving the scheduling of sporadic and aperiodic tasks, the scheduling of tasks whose deadlines are lower or higher than their periods, the protected access to shared resources, the precedence constraints and the non-preemptive scheduling. A detailed description of some fundamental results concerning these problems can be found in *Shin* [142], *Buttazzo* [48], *Liu* [157], *Burns* [46] and *Decotigny* [71].

2.1.2 Real-Time Medium Access Control in Communication Networks

Ensuring real-time communications is the responsibility of the entire communication stack. Nevertheless, as explained by *Zimmermann* in [267], the crucial role falls on the *MAC* (medium access control) sub-layer of the layer 2 of the *open systems interconnection* (*OSI*) model. The *MAC* sub-layer has the responsibility of managing the access to the communication medium, which may be shared between several

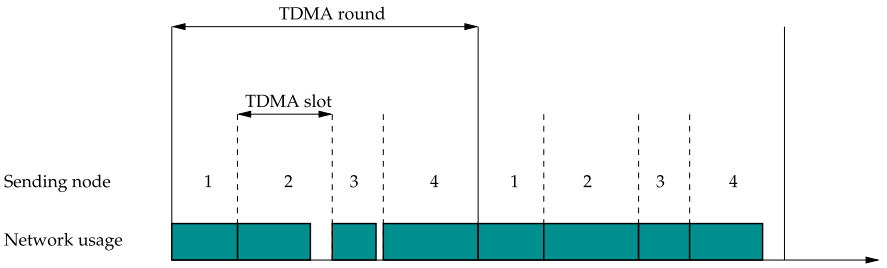

Fig. 2.2 Bandwidth sharing using the *TDMA* protocol and used terminology

nodes of the network. In real-time networks, there are several access protocols. The most deployed ones are described in the sequel:

2.1.2.1 Time Division Multiple Accesses (*TDMA*)

The *TDMA* [153, 183] protocol makes it possible to statically divide, the available bandwidth, in the temporal domain, between several competing nodes. In this protocol, each node knows exactly the moments when it is allowed to transmit over the network. As a consequence, each node has to transmit during the *time slot* which is allocated to it, called *TDMA slot*. In this way, collisions are avoided. The time slots are predetermined off-line. Their sequencing has a periodic structure. The minimal sequence of time slots allowing to describe the sequencing of the time slots of the various nodes is called *TDMA round* (Fig. 2.2).

The *TDMA* protocol may be implemented in a centralized or a distributed way. In centralized implementations, a master node has the responsibility of triggering the communications of the other slave nodes by transmitting a synchronization signal. The major inconvenient of this approach is that a breakdown of the master node leads to a total breakdown of the network. The distributed implementations require the establishment of a sufficiently precise global time in all the nodes of the network, requiring thus the use of some appropriate clock synchronization algorithms. The *TDMA* protocol is the cornerstone of the mobile communications *GSM* protocol. The *TTP/C* communication protocol [238] manages the concurrent access to the communication medium using a distributed implementation of the *TDMA* access protocol. In order to ensure a global clock, the *FTA* (*Fault-Tolerant Average*) algorithm proposed by *Kopetz and Ochsenreiter* in [141] is used to ensure the clock synchronization of the different network nodes.

2.1.2.2 Token-Bus

The *Token-Bus* access protocol was specified by the *IEEE* (*IEEE standard 802.4*) and by the *ISO* (*ISO standard 8802.4*). It represents the cornerstone of many

communications protocols that are employed in industrial field busses [231], like *Profibus* [18], *ControlNet* [66], *MAP* [164] and *ProfiNet* [39].

In this protocol, the network nodes are logically organized in a *ring topology*: each node knows its logical predecessor and its logical successor. The access arbitration is performed by the circulation of the *token* between the nodes. At any given moment, only one node has the token. The possession of the token gives to the possessing node the right to transmit over the network. The node that takes possession of the token can start transmitting over the network. It must pass the token to its successor if the time it has held the token reaches a limit or if it finishes transmitting before the expiry of this duration. If a node that does not have any information to transmit, receives the token, then it transmits it directly to its successor node. Concerning the real-time properties of this protocol, the analysis of the worst-case response times of *Profibus* and *ControlNet* messages are respectively given by *Tovar and Vasques* in [236] and by *Lian et al.* in [153]. The worst-case response time of a message is defined as the duration between the moment the transmission is requested and the moment when the message is received by the destination process.

2.1.2.3 Carrier Sense Multiple Access with Collision Avoidance (*CSMA/CA*)

The carrier sense multiple access with collision avoidance access protocol (*CSMA/CA* is used in many networks such as *CAN*, *DeviceNet* or IEEE 802.11 wireless networks. In this access model, each message is characterized by a unique priority. Since the shared communication medium can transmit only one message at a time, each node that wishes to transmit a message must initially check whether the network is free (by sensing the network to find whether or not a carrier signal is being transmitted). If the network is free, then the node can start transmitting. However, it is possible that other nodes start transmitting at the same time, because they have detected at the same time that the communication medium has become free. In this situation, the transmission of the highest priority message is continued; the other messages with lowest priorities are discarded. By this way, collisions are explicitly avoided.

CAN networks [132, 196] use the *CSMA/CA* protocol in order to manage the concurrent access to the shared bus. Their implementation of *CSMA/CA* relies on a bit synchronization mechanism at the bus level. In *CAN* networks, each message is characterized by a unique identifier. Furthermore, no node is particularly addressed: all the sent messages are broadcasted to all the other network nodes. When several nodes are emitting and if at least one node sends one a '0' (called dominant level in *CAN* terminology), then all listening network nodes will detect a '0' at the same time, even if there are other nodes which have transmitted a *'1'*. Reciprocally, when all the transmitting nodes send a *'1'* (called recessive level), all the listening nodes will detect a *'1'*. The CAN bus behaves like a logical AND gate. Since the identifier field is located at the beginning of the frame (the most significant bit is first coded, the highest priority is *0*), and the binary synchronization is implemented on the bus, when a collision occurs, the different nodes directly compare the identifiers of

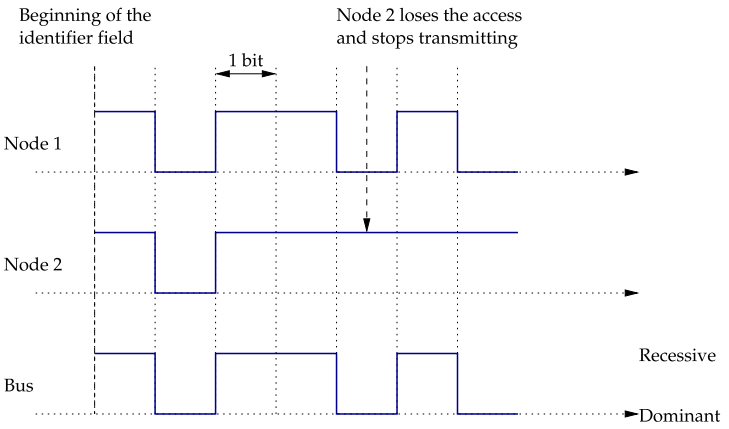

Fig. 2.3 Illustration of the bus arbitration mechanism in CAN networks

their messages to the resulting logical level at the bus. A node that detects that the resulting level is dominant (*'0'*) whereas it has sent a *'1'*, knows that it has tried to send a message whose priority is lower than another message and must consequently stop transmitting. Figure 2.3 describes a collision between two messages, which were sent at the same moment by node 1 and node 2. Node 2 stops transmitting when it detects that it has a lower priority than node 1 (because it has sent a *'1'* and the resulting level on the bus was a *'0'*).

CAN protocol is a deterministic protocol. Consequently, it is possible to compute an upper bound over messages response times. A *CAN* bus may be seen as a non-preemptive scheduler. The computation of the worst-case response time of fixed-priority messages have been tackled in *Tindell and Burns et al.* [232, 233]. The evaluation of a *RM* priority assignment policy was also addressed in these references. The use of *EDF* scheduling in *CAN* networks was studied by *Di Natale* in [74].

The binary synchronization over the bus, which is necessary to collisions arbitration, introduces a relationship between the maximum data-rate and the length of the network cable. In fact, to use a data-rate of 1 Mbps, the maximum length of the cable must be less than 40 m. If it is necessary to use a cable whose length is greater than 620 m, then the maximum data-rate falls down to 100 kbps.

Note that in data networks, like Ethernet or the Internet, the *CSMA/CD* (carrier sense multiple access with collision detection) protocol is the most used medium access protocol. The fundamental difference between *CSMA/CD* and *CSMA/CA* resides in the collision arbitration mechanism. In fact, in the *CSMA/CD* protocol, the nodes that generate a collision are able to detect this collision and to stop their transmission during a random duration. For that reason, it is impossible to bound messages response times in networks employing the *CSMA/CD* access protocol. In wireless networks, such as *IEEE 802.11* networks, the *CSMA/CA* protocol is employed because it is not possible to detect the collisions (and thus to deploy the *CSMA/CD* protocol).

2.1.3 Real-Time Scheduling of Distributed Systems

Real-time multiprocessor scheduling problems are still not as well understood as real-time single-processor scheduling problems; the most obscure points, as underlined by *Sha et al.* in [211], are related to the schedulability analysis. In this field, the impact of the *Dhall and Liu's* paper [73] on real-time multiprocessor scheduling theory was equivalent to the impact of the *Liu and Layland's* paper on real-time single-processor scheduling theory [156].

Multiprocessor scheduling algorithms may be classified into two categories [211]:

- *partitioned scheduling*, where each task is assigned to only one processor,
- *global scheduling*, where all the tasks compete for the use of all the processors.

The problem of the optimal partitioning of tasks among processors, as underlined by *Garey and Johnson* [93], is *NP-complete complexity.*[1] For that reason, simulated annealing or branch and bound-based heuristics were proposed to tackle this problem. The use of these heuristics relies on the modeling the scheduling problem as an optimization problem. A detailed presentation of these approaches is given in [16]. An outline of the most important results concerning multiprocessor schedulability analysis may be found in *Sha et al.* [211].

The tasks, that may be located on the different processors, may have data-dependencies and thus exchange messages via a communication medium. The communication medium may be seen as "a processor" that only supports the non-preemptive scheduling. Among the tools for off-line partitioned scheduling generation for tasks with precedence, on distributed architectures, one may cite [139, 217], which is based on the so-called *Adéquation Algorithme Architecture* (for efficient matching of the algorithm on the architecture) approach [102]. In *Syndex*, the architecture and the algorithm are described by two graphs. The algorithm graph is a direct acyclic graph, where the vertices represent the operations to be performed and where the edges represent the data-dependencies between the operations. The architecture graph describes the available parallelism as well as the communication possibilities between the various processors. Based on these two graphs, and possibly on placement constraints which may be specified by the user, *Syndex* uses the greedy list scheduling algorithm given by *Yang and Gerasoulis* and *Kwok and Ahmad* [144, 257] to synthesize the scheduling of the operations on the different architecture elements.

2.2 Integrated Approaches for Control and Resource Allocation

In the previous subsection, we have described some important results of the real-time scheduling theory. Disregarding the nature of the considered applications, these

[1] NP-complete is the set of all decision problems whose solutions can be verified in polynomial time.

results mainly addressed the problem of communication or computation resource allocation, in order to respect strict temporal constraints. In this subsection, we outline other approaches, which jointly consider the problems of control and communication or computation resources allocation. First, we review some problems and methods for the adaptive sampling. Second, we give an outline of the methods allowing to jointly considering the problems of control and communication resource allocation. Finally, we review the state of the art of the methods for the joint control and computational resource allocation in embedded systems.

2.2.1 Adaptive Sampling of Control Systems

The analysis of asynchronously sampled systems was undertaken at the end of the fifties. The research works, which were performed during the sixties and at the beginning of the seventies mainly focused on *single-input single-output* (*SISO*) systems. To the best of our knowledge, the first adaptive sampling method was proposed by *Dorf et al.* [77]. The proposed adaptive sampler changes the sampling frequency according to the absolute value of the first derivative of the error signal. Using analog simulations, the authors have shown that this method reduces between 25 % and 50 % of the number of required samples, with respect to the periodic sampling, given the same response characteristics.

Other approaches were proposed thereafter, in particular those of *Gupta* [108, 109], *Tomovic and Bekey* [234, 235] and *Mitchell and McDaniel* [176]. The evaluation of these approaches and their comparison to the periodic sampling was performed by *Smith* in [216]. Simulations have shown that these adaptive sampling methods are not always better than periodic sampling, especially where the input is subject to unknown disturbances. Remarking that the methods of *Dorf et al.* [77] and [176] are closely related to [125], *Hsia* proposed a generic approach allowing to derive adaptive sampling laws [126].

More recently, an important direction of research on the *DCES* performance enhancement through adaptive sampling is that of *Event Driven Controllers* (*EDC*). The aim of this class of controller is to reduce the calculation and communication resource utilization in order to provide in priority those tasks which need more. This results in a non periodic sampling depending on the state of all the subsystems composing a *DCES* and the "*policy*" implemented to handle their resources. Naturally, the hight level system specification such as stability needs to be assured which means that each new control signal has to be calculated, at least, with a minimum sampling frequency. In Årzèn [11], the *EDC* adapts its task period with respect to required system performances expressed as an event condition on the system state. These events are usually generated when the system state crosses an hyper-surface in state space. It is clear that the difficulties reside in the detection of this crossing and the event generator structure.

A second class of event based approaches called self-triggered one, consists in emulating event-triggered control by computing for each sampling instants a lower bound of the next sampling interval or the next candidate sampling instant.

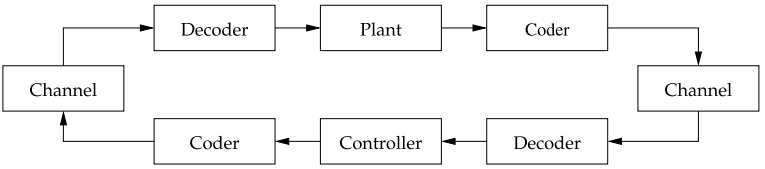

Fig. 2.4 General model of the information flow in a control system whose control loop is closed through finite bandwidth communication channels

As stated in *Årzèn* [11], event-driven control is closer in nature to the way a human behaves as a controller. Indeed, when a human performs manual control his behavior is event-driven rather than time-driven. This fact conjugated with the need to optimally handle calculation and communication resources partially explains the rationale behind the important research activity in designing event-based controllers.

The *EDC* controller are triggered by *external* events or they are self-triggered. The works of *Heemels et al.* [110], *Åström and Bernhardsson* [15], *Tabuada and Wang* [226], *Suh et al.* [223], *Heemels et al.* [111], *Henningsson et al.* [113], *Lunze* [163], *Wang et al.* [246], *Marchand et al.* [165], and many others, fall in the event-triggered class whereas the works of *Velasco et al.* [242], *Anta and Tabuada* [6–9], *Wang and Lemmon* [247–250], *Mazo and Tabbuada* [170, 172], *Araujo* [10] are some of the contributions in the latter class of self-triggered controllers.

In a general distributed control architecture related to *DCES*, reduction of sampling frequency is not always sufficient to enhance system performances. It is also needed to synchronize the decisions between subsystems sharing given calculation or communication resources. The recent works of *Tabuada* [225], *Seyboth* [210], *Donkers and Heemels* [75], *Mazo and Cao* [171]. *Wang and Lemmon* [251], *de Persis* [70] go in this direction with the objective to coordinate the subsystems' local decision.

2.2.2 Allocation of Communication Resources: The "Per Symbol" Paradigm

In this paradigm, the information exchange is modeled at the symbol level. The quantization of measurements and control commands is thus implicitly taken into account. The general model is given in Fig. 2.4.

In this model, the communication channel can transmit at most R bits per time unit. Because of these resource limitations, measurements and control commands must be encoded (as a flow of symbols) before their transmission and decoded at their reception. Various coding techniques may be employed. A fundamental question is to determine the necessary and/or sufficient data-rate allowing the existence of a coder, a decoder and a controller that achieve the stabilization of the system. Many contributions have tried to bring more insight into this fundamental question, by treating various models of resource limitations.

For instance, in [72], *Delchamps* has shown that it is impossible to asymptotically stabilize a discrete-time unstable *linear time invariant* (*LTI*) system, whose output passes through a quantizer having a finite number of quantization levels. In this setting, it is necessary to introduce and use other stability concepts, like, for example, practical stability.

The problem of state estimation, in the presence of state and measurement noise, was studied in *Wong and Brockett* [253]. In the considered model, the state observer is situated at the same location as the plant. However, the controller is located at a distant place. Consequently, the observations must be sent to the controller through a finite bandwidth communication channel. It was shown that this problem is quite different from the classic estimation and vector quantization problems. The concept of *finitely recursive coder-estimator sequence* was then introduced. Necessary conditions as well as sufficient conditions, which are related to the stability and the convergence of various coding and estimation algorithms, were stated. These conditions relate the network data-rate to the dynamical characteristics of the plant.

Next, in [254], *Wong and Brockett* introduced the concept of *containability*, as a weaker stability condition, to tackle the problem of the stabilization of networked systems through limited capacity communication networks, where the values of the measurements (which are received by the controller) and the controls (which are sent to the plant) belong to a finite set of values $\mathscr{J}$ (because of the quantization which is induced by the limited data-rate communication channel). Considering continuous-time *LTI* systems, which are impulsively controlled, they proved that a necessary condition to ensure the containability is $e^{\frac{2}{R}\mathrm{tr}(A_c)} \leq |\mathscr{J}|$ where $|\mathscr{J}|$ is the size of the alphabet, $\frac{1}{R}$ is the transmission duration of one bit and A_c is the state matrix. They also proved that if the initial condition of the system lies in a bounded set, then a memoryless coding and control is sufficient to ensure the containability, if some conditions affecting the data-rate are met.

In [182], *Nair and Evens* considered a class of discrete-time, linear, time-varying and infinite-dimensional plants. No process or measurement noise affects the considered plants. The initial state is the realization of a random variable. Communications constraints only affect the sensors-to-controller link. The controller is directly connected to the actuators. The considered problem is the synthesis of a coder (on the sensors-to-controller link) and of a controller that minimize a cost function of the state over finite and infinite horizons. The cost function over the finite horizon is the mth output moment. A coder/controller scheme was proposed. Under some technical assumptions, which are related to the probability density function of the initial state, to some conditions depending on the size of the alphabet, to the data-rate and to the plant dynamics, a necessary and sufficient optimality condition of the proposed coder/controller was established. A necessary and sufficient condition for the existence of a coder/controller that asymptotically stabilizes the system (in the sense that the mth output moment converges to zero over an infinite horizon) was stated. In the special case where the plant is invariant, unstable and finite-dimensional, this last condition simplifies to $R > \log_2 |\lambda|$ where λ is the unstable open-loop pole with the largest magnitude.

Next, in [42], *Brockett and Liberzon* proposed the idea of "zooming" as a means for ensuring the asymptotic stability of continuous-time and discrete time system whose control loops are closed through a finite bandwidth communication network. The zooming technique consists on changing the sensitivity of the quantizer over the time, based on the available quantized measurements. The relationship between performance and complexity of the quantized stabilization using the zooming technique was further studied by *Fagnani* in [82].

In [80], *Elia and Mitter* addressed the problem of the stabilization of single-input linear systems whose measurements and control commands are quantized. By first considering quantizers with a countable number of levels, and supposing the exact knowledge of the state, they proved that the coarsest quantizer allowing the quadratic stabilization of a discrete-time single-input *LTI* system, is logarithmic, and may be computed by solving a special *linear quadratic regulator* (*LQR*) problem. The state-feedback control problem as well as the state observation problem were solved within this theoretical framework. These results were thereafter extended to the continuous-time single-input and periodically sampled linear systems. The expression of the optimal sampling period (for the suggested quantizers) was established. It only depends on the sum of the unstable eigenvalues of the continuous system. This approach was finally extended to address quantizers with a finite number of levels.

Next, in [227], *Tatikonda and Mitter* considered discrete-time *LTI* systems. The control loop is closed through a limited capacity communication channel. Consequently, before their transmission, the measurements are quantized and encoded in symbols by a coder. At their reception, a decoder reconstructs a state estimate that will be used by the controller, which is directly connected to the plant. Two types of coders were studied:

- class 1 coders, which know past measurements, past controls and past transmitted symbols that were sent over the channel,
- class 2 coders, which only know past measurements.

Stabilization and asymptotic observability properties were particularly studied. It was shown that a necessary conditions for the existence of coders and decoders making it possible to guarantee these two properties is given by $R > \sum_{\lambda(A)} \max\{0, \log|\lambda(A)|\}$, the sum is over the eigenvalues of the state matrix A. This necessary condition is independent from coder classes and becomes sufficient if the whole state is measured and if class 1 coders are used.

2.2.3 Allocation of Communication Resources: The "Per Message" Paradigm

The different approaches that may be related to the "per message" paradigm are motivated by the fact that in all communication networks, protocol frames contain fields with fixed and incompressible length. These fields include, for example, the

identifier field, the *CRC* (Cyclic Redundancy Check) field, which is used by error detection algorithms. For example, in *CAN* networks, the minimal length of the fixed protocol fields is 47 bits (the length can be more important because of the bit-stuffing mechanism as given in *Rachid and Collet* [196]). A measure that is encoded in 12 bits and sent in a *CAN* message only represents 20 % of the size of the message. In *Bluetooth* networks, the minimal size of the data field is 368 bits. If an information whose size is less than 368 bits is to be transmitted, then padding by '0' bits must be carried out.

The various approaches, which are related to the "per message" paradigm, may be classified into two categories:

- The decentralized minimization of the network bandwidth usage. The basic idea is that each node tries to locally minimize its bandwidth consumption.
- The scheduling of the concurrent access to the network. In these approaches, a more global view of the application, of its distribution and of the network access mechanism is considered. The information transmission over the network is managed by taking into account static characteristics (the model) or instantaneous information (the state) of the controlled dynamic system.

2.2.3.1 Minimization of the Network Bandwidth Usage

A research direction related to the "per message" paradigm is the *model based control*, which was studied in *Yook et al.* [259], *Montestruque and Antsaklis* [177, 178], *Hespanha and Xu* [117, 118], *Li et al.* in [149, 150]. The basic idea of this approach is to use local open-loop observers in order to reduce the required communications between the sensors and the controller.

In [259], a method allowing the minimization of the required communications in some particular distributed control applications was proposed. This method addresses multivariable discrete-time *LTI* systems that are implemented according to a specific distributed architecture and controlled by using output-feedback. The global distributed system has n states, m inputs and m outputs, such that the ith input $u_i(k)$ and the ith output $y_i(k)$ are co-located at the same network node, which is also equipped with a computer (Fig. 2.5). The measurements, which are provided by the m sensors, are corrupted by a measurement noise $\nu(k)$. The m nodes are connected through a perfect network (without delays, nor losses of information). Each node contains an estimator that estimates at the same time its own output as well as the outputs of the other nodes. The estimated outputs $\hat{y}_j(k)$, $j \neq i$ are then used by the ith node, for the computation of the control $u_i(k)$, instead of the measured outputs $y_j(k)+\nu_j(k)$, $j \neq i$, whose use would require a transmission of their values over the network. By this way, an important communication saving is achieved, at the price of a more important computational load at the computer nodes. This approach relies on the fact that all the estimators compute identical values of the estimated outputs $\hat{y}_j(k)$. If the estimator of the ith node realizes that $|y_i(k) + \nu_i(k) - \hat{y}_i(k)| \geq g_i$, where g_i is a fixed threshold, then it broadcasts to the other nodes the value of its measurement $y_i(k) + \nu_i(k)$, enabling them to update the state of their estimators.

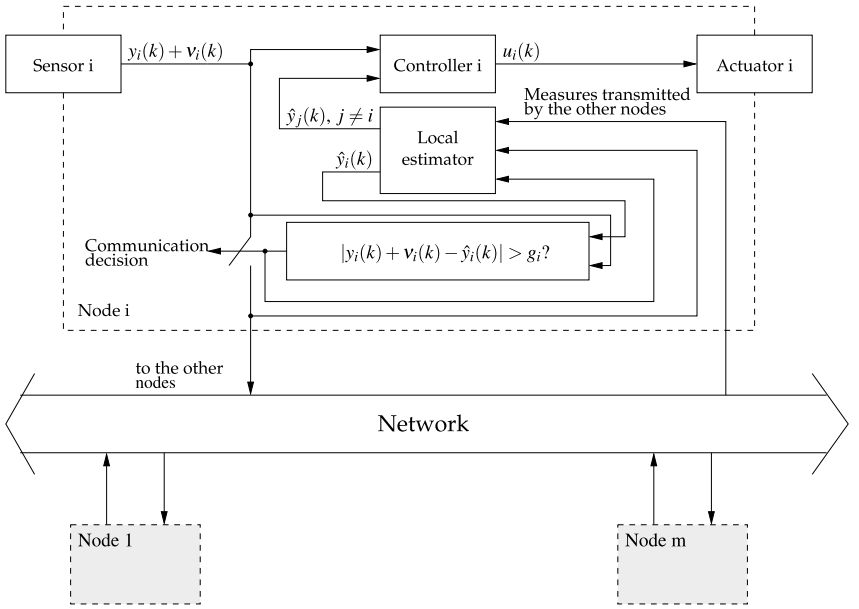

Fig. 2.5 Global architecture and model of the ith node according to the approach of [259]

By this way, $|y(k) + v(k) - \hat{y}(k)|$ is limited by $g = [g_1, \ldots, g_m]^T$. Finally, a result allowing the choice of H to guarantee a maximum degradation of ε % compared to the ideal system (without networked communication) was stated. The experimental validation of the method was carried out on a two axis contouring system. The experimental results have shown that only 12 % of the communication is necessary in order to guarantee a maximum degradation of 1 %, assuming a model uncertainty of 20 %.

A similar approach was studied in *Montestruque and Antsaklis* [177]. In the considered architecture, the controller is directly connected to the plant. A perfect network connects the sensors to the controller. The sensors periodically transmit (each T_s time instants) the measurements to the controller, which is provided with an open-loop state-observer. The estimated state is then used for the computation of the control commands. At the reception of a message from the sensors, the state of the open-loop observer is updated. The necessary and sufficient stability conditions of this particular model of networked control systems were stated, and generalized to take into account output feedback. Sufficient stability conditions when the sampling period, T_s, is time-varying but bounded were presented in *Montestruque and Antsaklis* [178] and *Li et al.* in [149, 150].

Next, in *Hespanha and Xu* [117], a similar architecture was studied. In the considered model, the plant is disturbed by a zero-mean Gaussian white noise. Instead of sending the measurements (of the full state) when the prediction error exceeds a threshold [259] or periodically [177], the transmission of measurements is performed using predefined communication logics. The study and the evaluation of

different communications logics (stochastic and deterministic) was also performed. By modeling the problem as an appropriate jump-diffusion process, sufficient conditions for the boundedness of the finite moments of the estimation error were established for the considered logics. Considering a long term average cost, penalizing at the same time the estimation error and the transmission rate, the expression of the optimal communication logics was explicitly given in *Hespanha and Xu* [118].

In *Gommans et al.* [97, 98] the main rationale behind the novel dropout compensators remains the same. They act as model-based, closed-loop observers if information is received and as open-loop predictors if a dropout occurs. These compensators were considered for two dropout models, using either worst-case bounds on the number of subsequent dropouts or stochastic information on the dropout probabilities. For the worst-case bound dropout model, sufficient conditions for global asymptotic stability of the closed-loop networked control systems (*NCS*) with the compensation based strategy are derived. For the stochastic dropout models, necessary and sufficient conditions for (exponential) mean square stability of the closed-loop *NCS* are given. In addition, for both dropout models they developed linear matrix inequality (*LMI*) based conditions for the synthesis of the compensator gains.

In Chap. 12 of this book, we propose a design methodology combining zero and hold strategies in order to optimize the system performance as well as to increase its stability domain in presence of packets dropouts. This static switching strategy may be adapted to operate state dependent one.

2.2.3.2 Medium Access Scheduling

The experimental study of communication networks characteristics was performed in *Nilsson* [187] and *Lian* in [153]. Studying the main characteristic of *ControlNet*, *DeviceNet* and *EtherNet* networks, *Lian et al.* [153] have shown that the transmission time of a message (i.e., the time the message spends on the physical link from the source to the destination) in the most used networks may be neglected. The delays occurring in networked control loops are mainly due to the *contention* between the different messages which are sent by the nodes of the network. The most efficient way of reduction of these delays is the design and use of appropriate message scheduling strategies.

These results show the practical importance of the study of the medium access control as well as scheduling algorithms. The problems of the concurrent access to shared communication resources were studied these last years within various theoretical frameworks and with various modeling assumptions. We present thereafter a brief summary of the approaches taking into account explicitly the concurrent access to the communication network. These contributions were classified into three categories, according to the class of the used scheduling algorithms: off-line scheduling, on-line scheduling of the sensors-to-controller link and the on-line scheduling of the controller-to-actuators link.

- *Off-line scheduling*: The problem of optimal control and off-line scheduling of the controller-to-actuators link was studied in *Rehbinder and Sanfridson* [200].

In the proposed model, the control commands are sent to the actuators through a shared *TDMA* bus. At each slot, only one control command can be sent, the remaining commands for the other actuators are held constant. The choice of which actuator to update at each slot was handled using the notion of communication sequence introduced by *Brockett* [43]. Only periodic communication sequences were considered. A quadratic cost function is associated to each communication sequence, corresponding to the worst-case initial condition and worst-case sequence permutation. Control commands and periodic communication sequences are obtained through the solving of a complex combinatorial optimization problem. The problem of the optimal control and scheduling in the sense of *LQG* was introduced and developed in *Lincoln and Bernhardsson* [155]. The relaxed dynamic programming method was applied for its resolution leading to a more efficient search heuristics. A heuristic approach for the problem of the optimal control and off-line scheduling of the sensors-to-controller link in the sense of the $\mathscr{H}_\infty$ performance index was proposed in *Lu* [161]. The application of branch and bound algorithm to this problem was performed in *Ben Gaid et al.* in [22]. The application of genetic algorithms and particle swarm optimization to this problem was undertaken by *Longo et al.* in [159]. A generalization of this approach to tackle uncertainties of the plant model was undertaken by *Al-Areqi et al.* in [2], and to cope with nonlinearities in *Su et al.* [221].

- *On-line scheduling of the sensors-to-controller link*: The scheduling of sensor measures was studied in *Walsh and Ye* [244]. The addressed configuration consists in a continuous-time plant where the controller is directly connected to the actuators. The network only connects the sensors to the controller. The notion of *MATI* (maximum allowable transfer interval) was introduced, and represents the upper bound on the time between two consecutive sensor messages transmissions that guaranties the stability of the plant. The *MATI* is defined for a given scheduling algorithm. A new on-line scheduling algorithm, called *MEF-TOD* (maximum error first—try once discard), was introduced. In this dynamic priority on-line scheduling algorithm, the priority of a sensor message depends on the error of the measure that it carries; smaller the error is, lower is the assigned priority. The error is defined as the weighted absolute value of the difference between the value of the current measure and the value of the last transmitted measure. If a node fails to send a message, then this message is discarded (dropped from the queue). The authors stated sufficient stability conditions, involving the value of the *MATI*, which ensure the stability of the system, when the *MEF-TOD* algorithm and a round robin like static scheduling algorithm are used. These results are based on the perturbation theory and are very conservative. This approach was generalized to nonlinear systems in *Walsh et al.* [245]. The practical implementation of the *MEF-TOD* algorithm was considered in [243]. This implementation, which was performed on *CAN* networks, is mainly based on the nondestructive bitwise arbitration of *CAN* technology to dynamically encode the dynamic priorities. The effects of the quantization of priorities were experimentally studied. Sufficient input/output $\mathscr{L}_p$-stability results for a class of network scheduling protocols, including *MEF-TOD* and static scheduling algorithms were stated and illustrated

in *Nesic and Teel* [185]. These results considerably reduces the conservativeness of the results that were initially stated in *Walsh et al.* [244]. The application of the *Rate Monotonic* scheduling algorithm to the networked control systems was investigated in *Branicky et al.* [41].

- *On-line scheduling of the controller-to-actuators link*: On-line scheduling of control commands to the actuators was studied in *Palopoli et al.* [191]. In the proposed model, it is assumed that every slot, only one command vector can be sent to an actuator group, the other control vectors are set to zero. The stabilization is achieved using a model predictive controller, which calculates on-line the appropriate control law and the allocation of the shared bus. The cost function used by the *Model Predictive Control* (*MPC*) calculates a weighted sum of the infinity norms of the states and the control commands over a specified horizon. The optimization problem solved at each step by the *MPC* algorithm was proven to be equivalent to the generalized linear complementarity problem (*GLCP*) [258]. The same architecture is considered in *Goodwin et al.* [99]. The considered model assumes that it is possible to send only one message during one sampling period. The actuators, which do not receive their control inputs, maintain constant the last received ones. The control commands are quantized with a fixed precision. The expression of the optimal model predictive controller, in the sense of a quadratic cost function, was established. The optimal solution as well as computationally efficient approach (*OPP*) to the problem of joint control and network scheduling was proposed in *Ben Gaid et al.* [27]. A solution expressed as piecewise linear feedback law was proposed in *Görges et al.* [101]. Robustness issues were investigated in *Al-Areqi et al.* [2], using the *OPP* algorithm for complexity reduction. The impact of network induced delays and packet dropouts to this problem were investigated in *Guo and Jin* [107].

2.2.4 Allocation of Computational Resources

2.2.4.1 Optimal Control and Mono-Processor Scheduling

The problem of the optimal selection of control tasks periods subject to schedulability constraints was addressed in *Seto et al.* [208]. Assuming that the discrete-time control laws are designed in the continuous-time domain and then discretized, the notion of performance index was introduced. The performance index quantifies the performance of the digitalized control law at a given sampling frequency. In most control applications, the performance index is minimal when the continuous-time control law is used and increases (i.e., degrades) as the sampling frequency is decreased (note that for some control systems this relationship is more complicated, as illustrated in *Eker* [79]). Considering this important class of control applications, the problem of the optimal sampling frequency assignment for a set of control tasks consists on minimizing a weighted sum of the performance indices of the considered control tasks subject to schedulability constraints. In *Rehbinder et al.* [199],

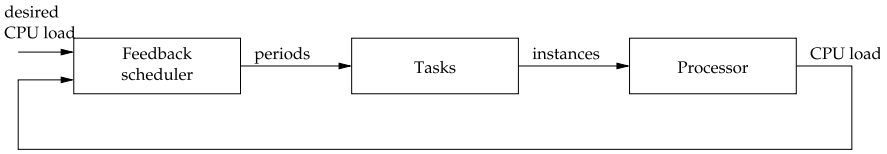

Fig. 2.6 General model of a feedback scheduler

the optimal off-line scheduling of control tasks in the sense of *LQG* was considered, assuming that all the control tasks have the same constant execution time. The resolution of this problem was performed using the exhaustive search method, which limits the application of this approach to applications with a limited number of tasks. Similarly, the problem of the optimal mono processor scheduling of control tasks in order to optimize a robustness metric (i.e., the stability radius) was treated in *Palopoli et al.* [192, 193].

2.2.4.2 Scheduling of Control Tasks in Environments with Variable Computing Workload

It is well known that worst-case analysis techniques given in *Sha et al.* [211] may be used in order to guarantee the deadlines of tasks with variable but bounded execution times. However, when the average execution time is smaller than the worst-case execution time (*WCET*), these techniques lead to an oversized design. Recently, new approaches were proposed in order to handle variations in tasks execution times and system overload more efficiently than worst-case analysis techniques, among them the feedback scheduling given in *Lu et al.* [160], *Cervin et al.* [55], *Robert et al.* [203], *Xia and Sun* [255] and the elastic task model given in *Buttazzo et al.* [49].

Feedback scheduling (*FSB*) is a control theoretical approach to real-time scheduling of systems with variable workload. The feedback scheduler whose general model is schematically given in Fig. 2.6, may be seen as a *"scheduling controller"* that receives filtered measures of tasks execution times and acts on tasks periods in order to minimize deadline misses. The application of feedback scheduling to robot control was experimentally evaluated in *Simon* [212].

In the elastic task model given in *Buttazzo et al.* [49], a periodic task set containing $\mathcal{N}$ tasks may be seen as a sequence of $\mathcal{N}$ linear springs. In this model, the utilization factor of a task is analogous to the spring's length. Tasks may change their utilization rate in order to handle overload conditions, which may occur, for example, if a new task is admitted to the computing system. In order to ensure the schedulability of the task set, tasks are compressed or decompressed. In *Liu et al.* [158], the elastic task model was applied to the scheduling of control tasks with variable execution times. The use of this method allows the application of the approach of *Seto et al.* [208] in order to find the optimal tasks periods based on tasks average execution times (instead of their worst-case execution times), leading to an improvement of the control performance. *Buttazzo et al.* [47] generalized

this approach to take into account the degradations that may occur to the control system if its control task that was designed to work at a given rate runs at another rate. The analytical expressions of the performance degradations were given and a compensation method was proposed. This method allows to trade-off the performance degradations and the required memory space (which is needed to store the parameters of the pre-computed control laws that will be used by the compensation algorithm).

However, all these described approaches are mainly based on the assumption that control performance is a *convex function* of the sampling period. In reality, as illustrated in *Martì* [167] (and further in the forthcoming Chap. 4), the quality of control is also dependent on the dynamical state of the controlled system.

The work of *Bini and Cervin* [34], *Samii et al.* [205, 206], although different in the formulation of the optimization problem in terms of objective functions and constraints, can be included in a subset of works that share in common an off-line approach where sampling periods are derived before run-time and kept constant during execution.

Inside the class of *FBS* methods, in our opinion, there are two main trends. The first one concerns those methods relaying on the instantaneous plant state information and metric in order to adapt the sampling period. In this class, we may classify the work of *Martì et al.* [169]. The second class includes those methods calculating the future sampling periods based on finite or infinite horizon metric as, for example, the methods developed in *Henriksson and Cervin* [114], *Castane et al.* [52], *Cervin et al.* [57]. This type of approach was also adopted in *Ben Gaid et al.* [28, 29] where the sampling period is calculated as a function of the states of the controlled plants, a given static scheduling and a quadratic metric over an periodic infinite horizon.

2.3 Notes and Comments

This chapter introduced the basic concepts, the terminology as well as the state of the art of communication and computation resource allocation approaches in *Distributed Control and Embedded Systems* (*DCES*). To this end, the basic concepts of the real-time scheduling theory are outlined, focusing primarily on the hard real-time scheduling of tasks on processors and messages on deterministic networks. An outline of the state of the art of the approaches for the integrated control and communication/computation resource allocation was given, providing an overview of the tackled problems and the provided solutions.

This particular interest on the hard real-time systems is motivated by their industrial realism and utility. Proposing simple and relevant integrated model of *DCESs* are of prime importance for their design as well as for their wide spread in industrial applications. We have clearly seen that different problems posed and treated in literature are related to their stability and performance robustness which are, practically, of different nature. The stability analysis of *DCES* is related to methods and tools used in the field of control and information sciences (*Mitter et al.* [80, 227])

whereas their performance optimization calls for methods and approaches related traditionally to control system domain and in parallel to those of computer science. The necessity of double-breasted view of the *DCES* design problem not only facilitates its analysis but allows optimizing its performances. Application of simple and consistent scheduling policy given respectively in *Yook et al.* [259], *Hristu* [123], *Longo et al.* [159], *Ben Gaid et al.* [27, 28] allows reducing their model complexity by inducing structure properties such as periodicity of control and/or sensing signals/messages. This fact, as it will be seen in the forthcoming chapters of Part II, simplify substantially the design of *DCESs* projecting it in a class of well-known problems in the area of control systems.

Naturally, the *DCES* area encompasses a larger class of applications including those for which a predefined or fixed scheduling policy is impossible. This is mainly due to non-determinism induced by the computation model related to the network nodes as well as to the model of communication related to the network. In this case, we observe variable induced delays on signals/messages sending and reception as well as modification of their reception order. This fact faced us with the problem of system information reduction in general and particularly the way to handle related variable induced delays. As it will be seen in the forthcoming Chaps. 10, 11, 12, 13 and from the literature review done in Sect. 2.2.1, two main trends are observed. The first one concerns the techniques related to system state information enhancement based on the system model as a complementary information source. The second one proposes methods and approaches whose objectives are to use the communication and calculation resources offered by *DCES* with respect to the system state or system performances. These two main trends have the same objective that is injecting complementary information in the system in order to overcome the lack of resources and/or to reduce the resources reserving them in priority to subsystems needing more. The topological structure and the size of *DCES* are different and function of the considered application. The decisions taken at the level of each subsystem composing it are effective if they are coordinated between them. The different approaches and techniques originally borrowed from the computer science such as synchronization help to increase the system performances. In the same time, they induce new models of communication between sub-systems that have to be considered and merged with the dynamic model of *DCES*. In fine, the main problem to handle is related to the structure and the quantity of information communicated between sub-systems in order to reduce the information delay of the critical system. Mastering this delay represents one of the main challenges in the design of *DCES*.

http://www.springer.com/978-3-319-02728-9

Optimal Design of Distributed Control and Embedded Systems

Çela, A.; Ben Gaid, M.; Li, X.-G.; Niculescu, S.-I.

2014, XXIV, 288 p. 94 illus., 60 illus. in color., Hardcover
ISBN: 978-3-319-02728-9