Chapter 2
Model Predictive Control Tools for Evolutionary Plants

Andrea Cataldo, Ivan Cibrario Bertolotti and Riccardo Scattolini

Abstract The analysis and design of control system configurations for automated production systems is generally a challenging problem, in particular given the increasing number of automation devices and the amount of information to be managed. This problem becomes even more complex when the production system is characterized by a fast evolutionary behaviour in terms of tasks to be executed, production volumes, changing priorities, and available resources. Thus, the control solution needs to be optimized on the basis of key performance indicators like flow production, service level, job tardiness, peak of the absorbed electrical power and the total energy consumed by the plant. This paper proposes a prototype control platform based on Model Predictive Control (MPC) that is able to impress to the production system the desired functional behaviour. The platform is structured according to a two-level control architecture. At the lower layer, distributed MPC algorithms control the pieces of equipment in the production system. At the higher layer an MPC coordinator manages the lower level controllers, by taking full advantage of the most recent advances in hybrid control theory, dynamic programming, mixed-integer optimization, and game theory. The MPC-based control platform will be presented and then applied to the case of a pilot production plant.

A. Cataldo (✉)
CNR-STIIMA, Istituto di Sistemi e Tecnologie Industriali Intelligenti per il Manifatturiero Avanzato, Milan, Italy
e-mail: andrea.cataldo@stiima.cnr.it

I. C. Bertolotti
CNR-IEIIT, Istituto di Elettronica e di Ingegneria dell'Informazione e delle Telecomunicazioni, Turin, Italy

R. Scattolini
Dipartimento di Elettronica Informazione e Bioingegneria, Politecnico di Milano, Milan, Italy

T. Tolio et al. (eds.), *Factories of the Future,*
https://doi.org/10.1007/978-3-319-94358-9_2

2.1 Scientific and Industrial Motivations

The configuration of the control system represents a key phase during the design and management of large scale complex production systems characterized by strongly interacting and possibly spatially distributed subsystems (such as power networks, transport networks, data traffic networks, and irrigation networks). The increasing amount of information to be managed and the need of flexibility and reconfigurability in industrial production contexts have a direct impact on control systems to deal with typical manufacturing problems such as routing, scheduling, and planning. Therefore, also the control systems must be flexible and scalable to cope with the selection of different control policies, plug-and-play operations, changes in the demand, modified conditions in the factory environment (e.g. addition or removal of sensors and actuators), self-reconfiguration after the malfunctioning of parts of the system.

Model Predictive Control (MPC) [1, 2] is widely used to control continuous industrial processes, such as chemical and petrochemical plants or pulp industry. However, its application in the discrete manufacturing industry is still in its infancy, although great advantages could be achieved in the design of the overall production system architecture in terms of scalability, adaptivity to changing environments, robustness to communication limitations and faults, asynchronous communication, reconfigurability with respect to the addition, replacement or removal of subsystems. These properties bring to the analysis and identification of most suitable distributed and hierarchical Model Predictive Control methodologies to be applied to flexible and time variant production systems, characterized by a very fast evolution concerning the whole product and process life cycle, from the ramp-up to the end-of-life.

This work proposes a two-layer control architecture based on MPC solutions to support the design and management of control system configurations for manufacturing plants. The lower layer of the architecture is dedicated to control the single piece of equipment and to consider the possible dynamic mutual influences and overall constraints, for example on the maximum admissible energy consumption. The higher layer is in charge of coordinating the whole system guaranteeing the fulfilment of operational and production constrains. The proposed control architecture has been implemented into a prototype control platform.

This chapter is organized as follows. The state of the art related to MPC is discussed in Sect. 2.2. The proposed solution is introduced in Sect. 2.3 and then further detailed in Sect. 2.4. The industrial case and the experiments can be found in Sect. 2.5, whereas the conclusions are drawn in Sect. 2.6.

2.2 State of the Art

Model Predictive Control (MPC) is the most popular and widely used advanced control technique in view of its ability to cope with linear and nonlinear models and to consider state, input, and output constraints directly in the design phase. In

addition, the parameters that characterize the controller working function can be easily tuned, starting from empirical models of the system to be controlled, models obtained with simple experiments on the plant like step or impulse responses. For these reasons, there are nowadays thousands of applications of MPC (e.g. [3]) and strong theoretical results have been developed [4].

MPC algorithms are usually designed and implemented according to a centralized approach, where all the information collected by the sensors are sent to a unique central station that computes the values of the commands to be transmitted to the plant actuators. This centralized structure guarantees the optimality of the control action and is suitable for many industrial plants. However, the implementation of a centralized structure becomes too demanding in terms of computational burden and transmission of information because of the growing complexity of the systems and of the increasing amount of information to be managed. For these reasons, and also to deal with large scale complex systems, many distributed versions of MPC, also called DMPC (Distributed MPC), have been proposed in recent years. The survey papers [5, 6] and the book [7] thoroughly describe the most recent methods. In DMPC, local MPC regulators are used to control parts of the system and exchange information according to a flat structure (i.e. single-level structure), in order to coordinate their actions.

Another way to handle the control of large scale complex systems is to resort to multilayer hierarchical control structures, instead of a flat one. At the lowest layer, a local MPC controller is designed for each subsystem, while at the higher layer an MPC controller (also named supervisor) coordinates and assigns high-level tasks to groups of subsystems [5]. The hierarchical control structure can be effective and provides a common framework to solve routing, scheduling, and planning problems.

Standard real-time protocols (e.g. Modbus [8]) are typically chosen to implement inter-module communication, as well as communication with the plant environment, in industrial applications. As outlined in [9] and further asserted in [10, 11], choosing the Modbus protocol offers significant advantages. For instance, it guarantees the resiliency of traditional fieldbus solutions. At the same time, it enables the improved bandwidth, open connectivity, and standardization of Ethernet-based networks.

About platforms and tools available for the MPC design and implementation, there are different industrial solutions implementing proprietary MPC algorithms, so they do not allow modifying the structure of the control algorithm but only set the related parameters to characterize the control behaviour. On the contrary, some tools (e.g. HYSDEL [12], YALMIP [13]) allow to design and implement MPC algorithms, starting from the system modelling through logic propositions, translating them into linear expressions Mixed Logical Dynamical (MLD) model [14], and then running the control algorithm.

2.3 Problem Statement and Proposed Approach

The reference production system consists of production resources (e.g. machine tools), storage spaces (e.g. buffer slots), transport systems (e.g. conveyors), and pallets that are used to move the work-in-progress parts. Two relevant industrial problems related to system control are investigated: *Production scheduling* and *Pallet routing*.

The *Production scheduling problem* is related to the design of a control algorithm for the production scheduling and buffer management of a multiple-line production plant. It is assumed that the machines can operate at different speeds corresponding to different energy demands. The controller must be tuned to optimally move the pallets from a source node to one of the available machines and to decide the processing speed through the minimization of a suitable cost function.

The *Pallet routing problem* deals with controlling the movement of the pallets along the network of machines and transportation system to optimize an objective function, while coping with problems such as traffic, starvation, bottleneck and deadlock.

The proposed approach exploits distributed and hierarchical MPC algorithms (see Sect. 2.2) to design a flexible and scalable *genomic MPC-based Control Platform*. This platform can assist and support industrial production system designers to distribute and integrate specific advanced model predictive control solutions into the production system architecture, in order to impress to the whole automated production system an adaptable behaviour. The term *genomic control* comes from the analogy with the DNA model, in which the combination and the specific integration of *control kernels* (nucleotides) into the production system architecture (helical structure) defines the automated production system as a whole. Control kernels will be hosted within a virtual execution environment to guarantee their isolation and platform independency, while preserving their real-time execution characteristics.

Among open-source virtualization products, Xen [15] can be mentioned as a relevant Virtual Machine Monitor (VMM) for the x86, x86_64, and IA64 architectures. Even though Xen has not been designed with real-time execution in mind, the research community is actively working to overcome this limit [16, 17]. More specifically, the RT-Xen project [18] aims at developing a fully real-time, Xen-based hypervisor. However, the existing open-source virtualization products are characterized by two main shortcomings:

- Due to their focus on data centres and server farms, most existing virtualization products were not specifically designed for real-time execution. For this reason, detailed real-time performance data are usually not publicly available and it is unclear if, and to what extent, any real-time software hosted in a Virtual Machine (VM) instead of a physical processing node will still satisfy its timing requirements.
- The scientific literature (e.g. [19]) has shown that the isolation of different applications hosted on the same processing node might be less than perfect. This is also true for VMs and especially for what concerns undue timing interferences, which are of paramount interest for real-time execution.

For these reasons, in the design of the execution environment, special attention is paid to analyse the configuration parameters of the virtualization software, as well as their effect on the real-time characteristics of the system. Secondly, it is necessary to carry out a preliminary evaluation of the degree of isolation provided by the virtualization software with respect to concurrent execution of multiple VMs on the same processing node.

Each control kernel is equipped with a specific standardized software communication interface to exchange data with the external environment (i.e. hardware devices in a production system). The same interface also takes care of the internal communication between control kernels. Following the analysis reported in Sect. 2.2, Modbus has been chosen for internal communication between MPC modules.

The careful deployment of control kernels in a virtual execution and communication environment further improves flexibility because it makes control kernels mostly independent on the physical characteristics of the system without sacrificing their real-time performance.

The obtained control solutions are optimized on the basis of key performance indicators like flow production, peak of absorbed electrical power and total energy consumed by the plant, so that such control algorithms are able to imprint to the production system the desired functional properties. This is especially useful for contemporary automated production systems that are often characterized by a fast and evolutionary behaviour.

2.4 MPC-Based Control Platform

As anticipated, the proposed MPC-based control platform is structured in a two-layer software control structure.

The *lower layer* manages a set of atomic control kernels (i.e. MPC modules) that are deputed to implement specific MPC functionalities. Such control kernels are selectively distributed and then integrated into the industrial production system hardware devices (e.g. hardware controllers, sensors and actuator drivers) by the automation plant designer to control individual pieces of equipment in the production system. In particular, an innovative MPC algorithm for nonlinear systems has been developed. The common approach to deal with nonlinear systems is to consider simpler models obtained through linearization procedures, so that the resulting optimization problems to be solved on-line are quite simple and compatible with computational limitations, although the neglected nonlinearities can lead to poor performance. Alternatively, purely nonlinear MPC algorithms require a heavy computational effort and pose difficult optimization problems. An MPC algorithm has been developed to overcome these limitations, by considering the model of the system linearized along the planned trajectories. This method is an evolution of the approach presented in [20] with enhanced properties with respect to the original one.

The *higher layer* supports the design, evaluation and implementation of the production system automation architecture and the related control functionalities. In

particular, this software layer implements an MPC plant coordinator taking full advantage of the most recent advances in hybrid control theory. The software implementation is based on a workbench suitable to support the automation engineer to formalize and implement the flexibility and reconfigurability concepts of an evolutionary production system by converting automatically them into a specific selection and combination of control kernels. Furthermore, innovative solutions based on MPC for manufacturing systems described by Mixed Logical Dynamical (MLD) models have been developed. MLD enables the description of the dynamics characterizing event-driven systems and gives a mathematical formulation of the logical and operational constraints to be considered by using Boolean variables.

The control platform has been implemented in C++ programming language and named Dynamic Control Platform for Industrial Plants (DCPIP). The Object-Oriented programming paradigm [21] has been chosen to support modularity and guarantee easy maintainability. Moreover, the internal data architecture is dynamic, i.e. the platform dynamically builds the necessary data structure, according to the production system structure, to cope with the specific plant to be controlled. Indeed, the platform contains a block library where a certain number of specific and predefined operating machine control modules (the so-called *kernels*) are stored.

DCPIP offers specific structural and functional characteristics to deal with the control kernels and with the higher software level pertaining to the coordinator control algorithms.

The MPC controller was developed in MATLAB, making use also of the YALMIP, HYSDEL and CPLEX software tools. The MPC algorithm is managed by the DCPIP control environment by launching the following computation phases:

- Building of the optimization problem (model of system and cost function).
- The state of the system to be controlled is acquired (Input).
- Solving of the optimization problem.
- Application of the calculated control action.

The DCPIP can be interfaced with either a real plant or a virtual plant that can be modelled as discrete event simulator. No change is required in the DCPIP if the simulator accurately represents the real plant from the point of view of the Input/Output variables.

The control kernels represent software modules that are dynamically generated during the DCPIP start-up to provide specific control and communication functionalities. Since the Object-Oriented paradigm is used for the DCPIP implementation, then the generic kernel can be described in terms of a class object that are instantiated according to the specific plant to be managed. The control kernels needed to design the DCPIP are:

- *Task_Manager*
- *Line_Supervisor / Plant_j_Line_Supervisor*
- *Machine / Machine_j*
- *Controller / Controller_j*
- *Interface_vs_ext / Interface_vs_Txt_file / TCP_IP*

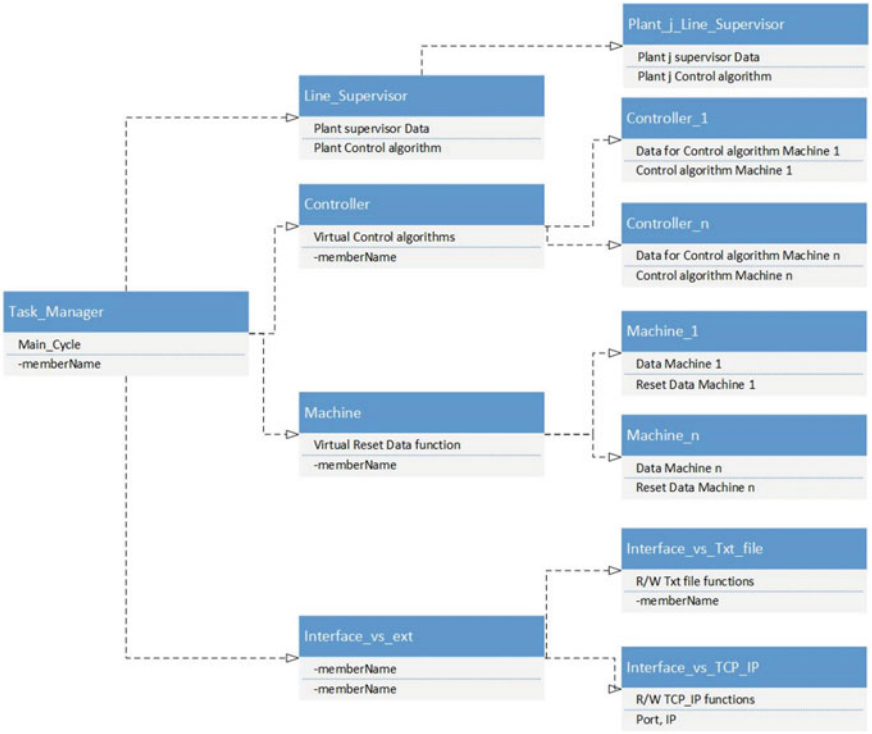

Fig. 2.1 Kernels architecture

A hierarchy of the different kernel classes is shown in Fig. 2.1 and further details are presented in the following subsections.

The control software platform has to communicate the Input/Output process and monitoring variables not only via text files but also via fieldbus (i.e. TCP). This is very important in order to distribute the control platform on different devices (controllers and PCs) connected to each other in a network. This allows to allocate the control platform, the optimization process, the simulator of the plant and the plant controllers all on different devices and to use different software technologies to run such software components (i.e. Virtual Machines).

A kernel hierarchic architecture (including plant supervisor, machine controllers, and communication software modules) requires the implementation of a cross-communication between all the software components, in order to be able both to exchange data with the plant simulator/real plant and to coordinate the different control kernels according to specific supervisor algorithms run into specific control kernel themselves.

2.4.1 Main_program

This kernel contains the software instructions needed to dynamically start the whole CP data structure configuration:

- Instantiation of the interface between the control architecture and the external process environment.
- Instantiation of the pointer to the kernel *Line_Supervisor*. This software module is responsible for the management of the data structure of the configured resources in the Plant or Simulator.
- Start of the *Task_Manager* main cycle. This software program schedules the activation of the different control algorithms for each configured resource in the Plant or Simulation model.

2.4.2 Task_Manager

The *Task_Manager* kernel contains all the variable and methods for the execution of each kernel. In particular, it starts the dynamic kernel generation, performs the control platform Main Cycle that scans the Input variable coming from the PLC/Simulator, runs the control algorithms contained in the kernels and updates the Output variables to be sent to the PLC/Simulator. It must be highlighted that the Input acquisition and the Output updating are carried out by means of the communication functionalities implemented in dedicated kernels.

2.4.3 Line_Supervisor / Plant_j_Line_Supervisor

The *Line_Supervisor* kernel implements the control algorithm of the plant supervisor controller. In particular, it communicates with each machine control kernel (horizontal or cross communication) besides with the PLC/Simulator by means of the I/O data exchanged via text files or TCP protocol.

This class takes advantage of the object-oriented paradigm because the classes derived via inheritance from *Line_Supervisor* (*Plant_j_Line_Supervisor*) can implement specific control algorithms for the considered industry sector (Petro-chemical, Pulp&Paper, Wood, Steel, Automotive, etc.) by means of on virtual functions.

2.4.4 Machine / Machine_j

The *Machine* kernel contains variables declaration and relative methods to perform the data structure management of the generic machine that must be controlled. The

Machine_j kernels represents the specific machine control algorithm implementation. This means that the machine control strategies must be implemented in such kernels.

2.4.5 Controller / Controller_j

The *Controller* kernel contains the control algorithm of the corresponding machine to be controlled. Each machine is characterized by its own control functionalities, therefore the *Controller* class has some virtual functions that are specialized in the derived class controllers *Controller_j*.

2.4.6 Interface_vs_ext / Interface_vs_Txt_file/TCPIP

The *Interface_vs_ext* kernel implements the communication of the *Machine_j* controller and *Line_Supervisor* controller kernels with the external environment, e.g. all the Input/Output data exchanged between the kernels and the PLC/Simulator. This class has methods to execute the Read/Write functions on the Input/Output data, according to the different communication methods (e.g. text file and TCP protocol). Furthermore, this class is virtual because there are different communication channels and further communication protocols could be defined and added to the control platform structure. The *Interface_vs_Txt_file* and *Interface_vs_TCPIP* kernels implement the specific communication methods, being classes derived from the parent class *Interface_vs_ext*. The Read/Write functions are specialized according to the relative communication channel (Text files or TCPIP protocol).

2.5 Testing and Validation of Results

This section presents how the control platform (Sect. 2.5.2) was customized for a reference industrial case (Sect. 2.5.1) to run a set of experiments (Sect. 2.5.3).

2.5.1 Industrial Case

The reference industrial case is a de-manufacturing pilot plant (Fig. 2.2) implemented in the lab of CNR-STIIMA (ex CNR-ITIA) [22–24]. The system was designed for testing and repairing of printed circuit boards (PCBs) and it consists of four cells and a transport line based on transport modules, in particular:

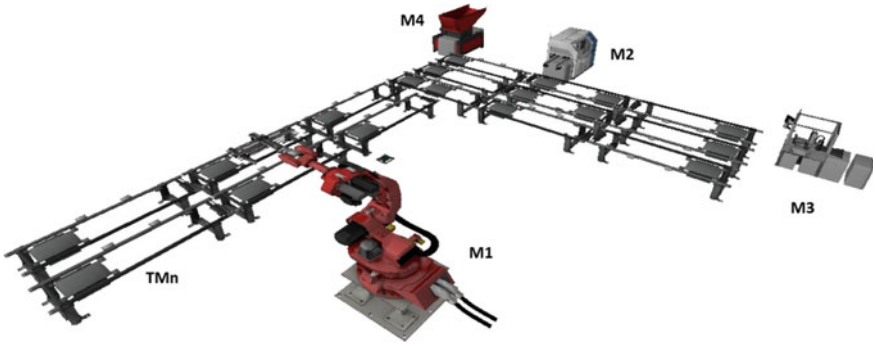

Fig. 2.2 De-manufacturing pilot plant

- *Cell M1* consists of a robot cell able to load/unload the PCBs on a pallet that is then placed on the transport line to be moved towards other machines.
- *Cell M2* consists of a testing machine where PCBs are tested in order to identify the specific faults.
- *Cell M3* consists of a reworking machine where PCBs are repaired by replacing failed components.
- *Cell M4* that is able to unload the PCB from the pallet. Then the PCB is sent to the recycling area of the plant.
- 15 *Transport Modules TMi* (with i ranging from 1 to 15), connected together to compose a modular and flexible transport line.

The typical sequence of operations performed by the de-manufacturing plant consists of the following steps:

- The board is loaded on a pallet by M1.
- The transport line moves the pallet to M2 where the board is tested and possible failures are identified to decide whether the board is sent to M3 or M4.
- If a repair is needed, then the pallet with the board is sent first to M3 and back to M2 to test it again.
- If the board is properly working, then it is sent back to M1 where it is unloaded from the pallet and stored in the warehouse. Otherwise the board is sent to M4.
- After the board is removed, the pallet is ready to load a new board and start a new cycle.

The Transport Modules enable to move the pallet and stop it in specific positions based on the transport line topology, on the configuration of each module in terms of mechanical structure, on the automation system instrumentation (i.e. actuators and sensors), and on specific low level control system functionalities. Thus, the pallet can be moved from a specific position to another one called *Buffer Zone* ($BZ_{i,j}$), or in general *Node* N_k, where $BZ_{i,j}$ represents the j-th stop position on the i-th transport module [25]. If these Buffer Zones are considered from a mathematical point of view, then the plant can be seen as a graph whose nodes represents all the possible

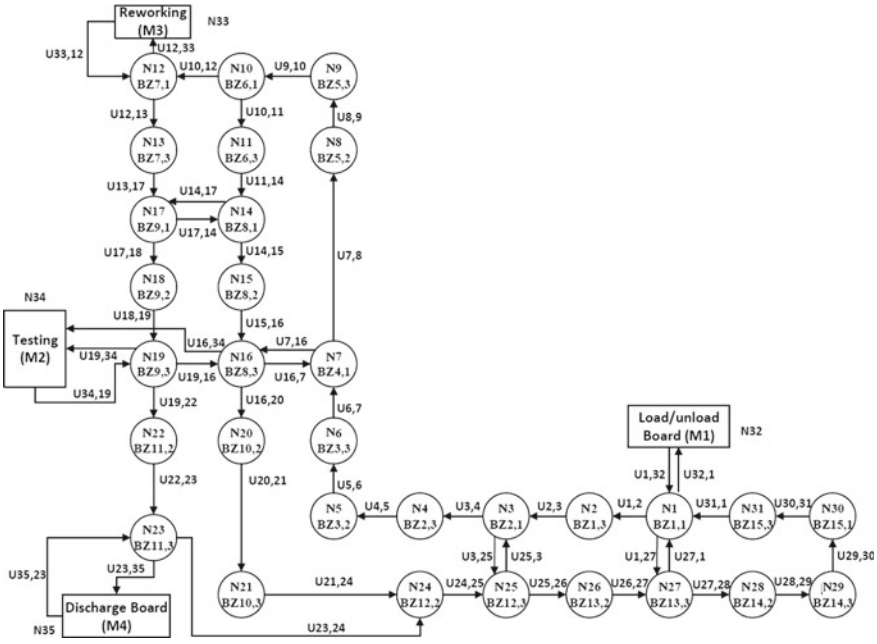

Fig. 2.3 Pilot plant graph representation

positions the pallets can take and the arcs define the possible movements between those positions. Each machine can be seen as a special node that must be visited by the pallet for a minimum amount of time (i.e. the processing time). The resulting reachability graph is depicted in Fig. 2.3. This kind of graph can also be automatically generated thanks to artificial intelligence (AI) techniques as discussed in [26].

2.5.2 *De-manufacturing Pilot Plant Control Platform Implementation*

The extended automation system architecture involving the control platform (Sect. 2.4) and the Plant/Simulator of the industrial case were designed according to the functional structure depicted in Fig. 2.4. A specific MPC algorithm has been developed [27] for the de-manufacturing pallet transport line [28].

The following software components of the control platform are hosted on a PC:

- *DCPIP* (see Sect. 2.4). Such control system acquires via text file the Input from the Hard Disk of the PC, elaborates it and writes the corresponding Output via text file on the same Hard Disk.

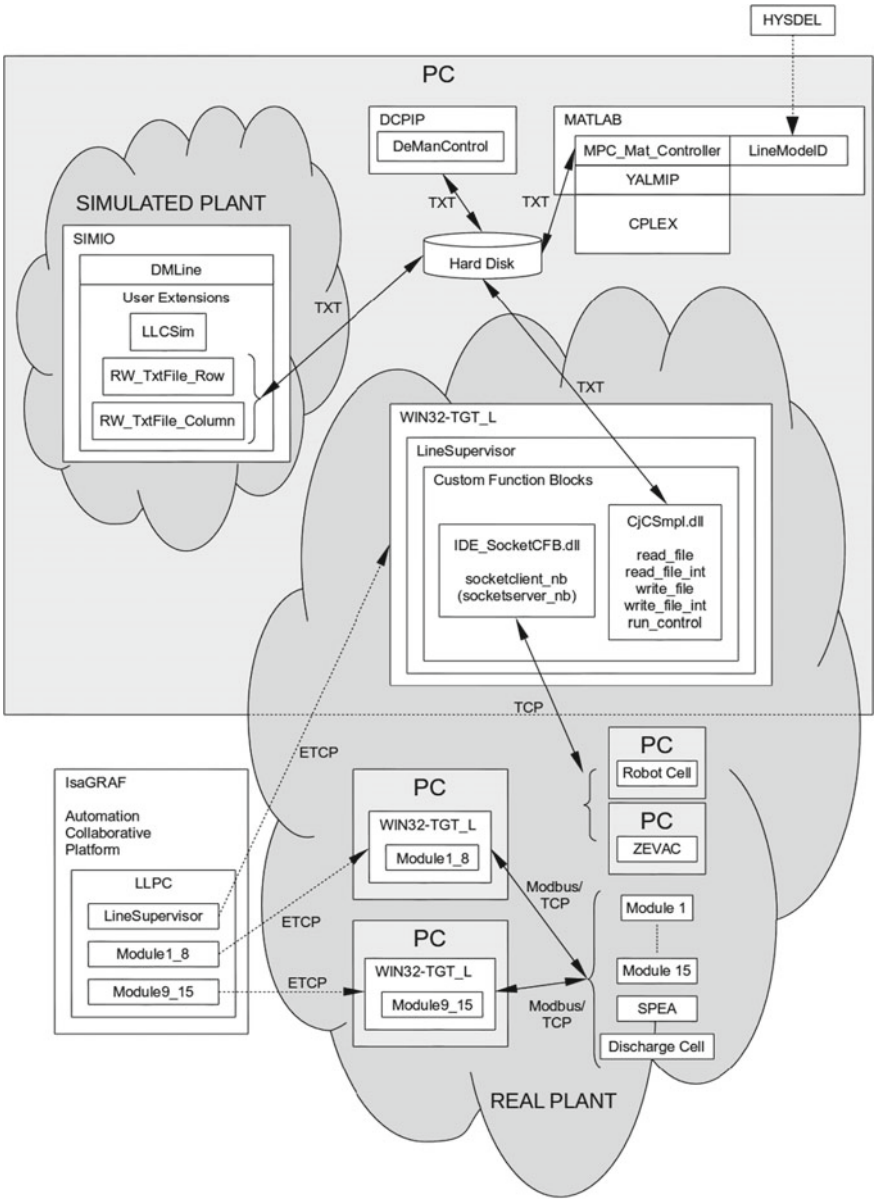

Fig. 2.4 Extended automation system architecture

- *MATLAB* where the Model Predictive Controller is implemented and run (see Sect. 2.4).
- *Simulated Plant*, implemented using SIMIO commercial tool. The simulator represents the real plant from the point of view of the Input/Output variables and performs the plant dynamics, by reading via text file the control actions that are sent by the control environment and calculates the relative process variables.
- *Plant Target WIN32-TGT-L* consists of the low-level controller of the plant that communicates with the software control environment the same way as the *Simulated Plant*.

The software configuration of the real plant consists of:

- Soft PLC (i.e. PLC algorithms running on PC and implemented in ISaGRAF platform) used to implement the plant *low level* control functionalities.
- PC, used as controllers of specific plant devices (Robot cell M1 and reworking machine M3).

For what concerns communication, a CAN-based Modbus adaptation layer (called Modbus CAN) has been designed and implemented. The protocol was successfully analysed with the help of a model checker, to prove several properties of interest related to correctness and communication error tolerance.

In order to evaluate and improve VM-based execution in a realistic scenario, close to typical industrial automation applications, the measurements have been carried out on a setup based on Modbus TCP communications between a master node and a slave node. In this setup, the master node coincides with the execution environment under test and is based upon a Tecnint Leonardo PC BOX industrial embedded PC. The slave node has been built using a standard personal PC that mimics the behaviour of a typical Modbus slave. Performance evaluation was based on round-trip time measurements of Modbus TCP frame exchanges. As a result, the Xen credit scheduler was modified to improve the real-time behaviour of virtual machines dedicated to real-time execution.

2.5.3 Experiments

Several experiments have been carried out to assess the performance of the proposed approach. The obtained experimental results show a satisfactory behaviour in terms of the number of pallets that can be loaded on the transportation line and of the average time required to move the pallets from the initial node to their final destination. Notably, the computational load required to solve on line the optimization problem required by the solution of MPC is fully acceptable for the considered case.

The results obtained addressing the *Production scheduling* and *Pallet routing* problems (Sect. 2.3), extensively described in [27, 29], show that the developed algorithms, based on the MLD representation of the systems considered in the two cases, are highly flexible, so that they can be easily adapted to different problems.

Table 2.1 Task priority assignment for Modbus TCP protocol stacks

	Master	Slave
Ethernet receive task	+4	+3
LWIP main task	+3	+2
Modbus TCP receive helper	+2	n/a
Application	+1	+1
Idle RTOS task	0	0

Moreover, it is easy to obtain different behaviours of the controlled manufacturing system by properly tuning easy-to-understand parameters of the algorithm, such as the cost function and the constraints defining the on-line optimization problem to be solved in MPC. The MPC-MLD approach also enables to cope with dynamic changes of the production environment, such as minimum production and maximum absorbed power requirements in the *Production scheduling problem*, or local malfunctioning of the transportation line in the *Pallet routing problem*. Finally, the important issue related to the computational load required to solve on-line MILP problems has not resulted to be a real bottleneck both in the considered simulation environment and in the real application of these methodologies to a complex pilot plant (see [27]).

For what concerns virtual execution and communication, the experiments have been focused on developing and analysing an appropriate priority assignment scheme for the various tasks involved in the Modbus master and slave protocol stacks. The main goal of the analysis has been to determine a priority assignment scheme that minimizes communication jitter, based on previous experience on jitter analysis gained on similar applications [30]. As it can be seen in Table 2.1, jitter reduction has been achieved by favouring application-level message processing with respect to the lower layers of the protocol stacks.

Afterwards, the performance of Modbus TCP communication has been evaluated and compared with a plain data exchange through a direct TCP connection. The experimental results presented in [30] show that the real-time characteristics of Modbus TCP communication are affected by the underlying TCP segment acknowledge mechanism. More specifically, this mechanism induces a significant communication jitter that, as a result, may hinder the applicability of this method in demanding real-time systems. It is also worth noting that the dependency between segment acknowledgment and jitter is not straightforward and depends on multiple factors, e.g. the above-mentioned priority assignment, the protocol stack architecture, and the traffic initiator (resulting in an asymmetry between the timing behaviour of master requests and slave responses). Further experiments carried out on the same testbed have also provided additional insights on the overhead of the Modbus protocol stack layer with respect to a streamlined, custom TCP-based protocol. This information can be profitably used, for instance, in the design of other similarly distributed embedded systems.

The implemented MPC takes into account the graph-based representation of the plant (Fig. 2.3). Figure 2.5 shows how the control platform works if two pallets are concurrently moved along the plant that is managed as if it consists of multiple feeder

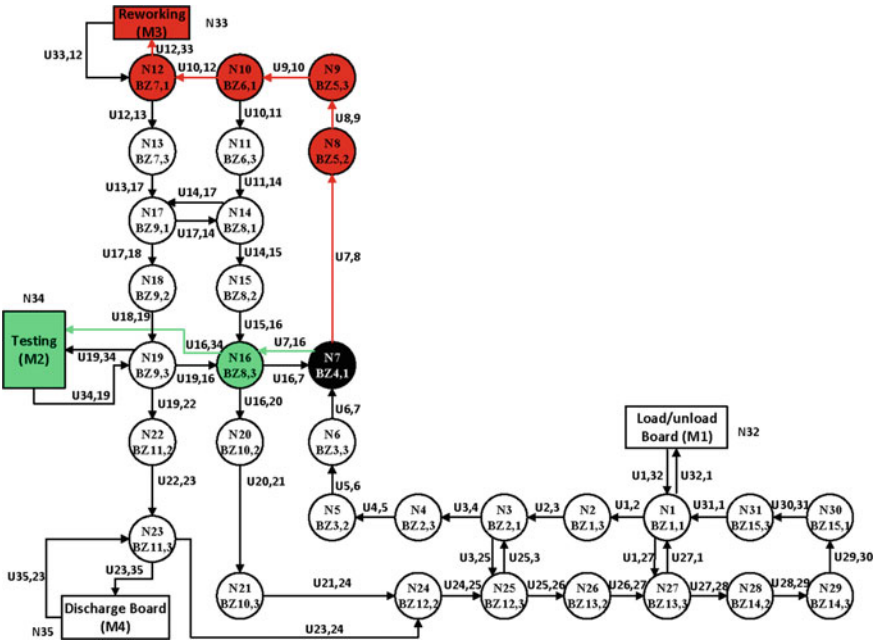

Fig. 2.5 Pilot plant test path definition

lines. One line feeds the reworking machine M3 and is composed of four buffer nodes (N_8, N_9, N_{11}, N_{12}); the other line feeds the testing machine M2 and is composed of one buffer node (N_{16}). Therefore, the approach proposed in Sect. 2.3 can be applied in this part of the pilot plant that is equivalent to a generic plant with two operating machines working with different processing time and two feeder lines with different length.

2.6 Conclusions and Future Research

Motivated by the advanced capabilities of MPC in the process industry and the increasing demands of high performing controllers, this work has applied MPC also to the discrete manufacturing industry by developing a new optimization-based control. We focused on the buffer management and the production scheduling of a multiple-line production plant, aiming to provide optimal scheduling strategies with respect to production requirements. Experimental results show that the algorithm can be highly adapted to obtain different behaviours, by means of simple and easy-to-understand parameters of the cost function. Moreover, such algorithm allows to dynamically change the minimum production and the maximum energy available and to choose, if present, which possible constraint should be violated if necessary [29]. All these

features can be hardly achieved with standard controllers or with simple scheduling. The activities related to real-time communication led to a better understanding of the TCP acknowledgment mechanism in the context of Modbus TCP communication [30], as well as the definition of a CAN transport layer for Modbus [31].

The research activity related to the design of hierarchical control systems for the manufacturing industry has been described in a number of journal and conference papers. Specifically, the *Production scheduling problem*, together with the methodology adopted for its solution, has been the object of the paper [29], and related results have been reported in [32]. The results obtained in the *Pallet routing problem* have been extensively described in [27, 28]. Finally, an innovative methodology to compute the energy consumption of manufacturing systems has been proposed in [33].

Concerning future developments of the activity, the computational effort is obviously one of the main obstacles to the diffusion of Model Predictive Control in manufacturing systems, and many improvements are still required to design efficient control algorithms. Distributed optimization methods must be developed, in particular for MLD models, in order to be able to control larger and larger systems and to provide new and efficient solutions. Another important topic of future research concerns the possibility to customize the algorithms and the creation of libraries of models to reduce the development times.

Acknowledgements This work has been funded by the Italian Ministry of Education, Universities and Research (MIUR) under the Flagship Project "Factories of the Future—Italy" (Progetto Bandiera "La Fabbrica del Futuro") [34], Sottoprogetto 2, research project "Genomic Model Predictive Control Tools for Evolutionary Plants" (IMET2AL).

References

1. Camacho EF, Bordons Alba C (2007) Model predictive control. Springer
2. Maciejowski JM (2002) Predictive control with constraints. Pearson Education Limited, Prentice Hall
3. Qin SJ, Badgwell TA (2003) A survey of industrial model predictive control technology. Control Eng Pract 11:733–764
4. Rawlings JB, Mayne D (2009) Model predictive control: theory and design. Nob Hill Publishing, Madison, WI
5. Scattolini R (2009) Architectures for distributed and hierarchical model predictive control—a review. J Process Control 19:723–731
6. Christofides P et al (2013) Distributed model predictive control: a tutorial review and future research directions. Comput Chem Eng 51:21–41
7. Maestre JM, Negenborn RR (2013) Distributed model predictive control made easy. Springer
8. Modbus-IDA (2006). Modbus application protocol specification V1.1b. http://www.modbus-ida.org/
9. Ballarino A et al (2014) On the performance of an automation solution based on IEC 61499 and OPC UA. In: Proceedings of the 17th IEEE conference on emerging technologies and factory automation (ETFA), pp 1–6
10. McConahey J (2011) Using Modbus for process control and automation, part 1. Appl Autom A12–A14

11. McConahey J (2012) Using Modbus for process control and automation, part 2. Appl Autom A12–A14
12. Torrisi FD, Bemporad A (2004) HYSDEL-a tool for generating computational hybrid models for analysis and synthesis problems. IEEE Trans Control Syst Technol 12(2):235–249
13. Lofberg J (2004) YALMIP: a Toolbox for Modeling and Optimization in MATLAB. In: 2004 IEEE international conference on robotics and automation, Taipei, pp 284–289
14. Bemporad A, Morari M (1999) Control of systems integrating logic, dynamics, and constraints. Automatica 407–427
15. Barham P et al (2003) Xen and the art of virtualization. In: Proceedings of the 19th ACM symposium on operating system principles, pp 164–177
16. Ongaro D, Cox AL, Rixner S (2008) Scheduling I/O in virtual machine monitors. In: Proceedings of the 4th ACM SIGPLAN/SIGOPS international conference on virtual execution environments, pp 1–10
17. Yu P et al (2010) Real-time enhancement for Xen hypervisor. In: Proceedings of the 8th IEEE/IFIP international conference on embedded and ubiquitous computing (EUC), pp 23–30
18. Xi S et al (2011) Rt-Xen: towards real-time hypervisor scheduling in Xen. In: Proceedings of the international conference on embedded software (EMSOFT), pp 39–48
19. Cereia M, Cibrario Bertolotti I (2009) Virtual machines for distributed real-time systems. Comput Stand Interfaces 31:30–39
20. Falcone P et al (2008) Linear time-varying model predictive control and its application to active steering systems: stability analysis and experimental validation. Int J Robust Nonlinear Control 18:862–875
21. Ghezzi C, Jazayeri M, Mandrioli D (1991) Fundamentals of software engineering. Prentice Hall
22. Colledani M, Copani G, Tolio T (2014) Management in manufacturing. In: Proceedings of the 47th CIRP conference on manufacturing systems
23. Copani G et al (2012) Integrated de-manufacturing systems as new approach to end-of-life management of mechatronic devices. In: 10th global conference on sustainable manufacturing, Istanbul, Turkey
24. Tolio T, Copani G, Terkaj W (2019) key research priorities for factories of the future—part II: pilot plants and funding mechanisms. In: Tolio T, Copani G, Terkaj W (eds) Factories of the future. Springer
25. Cataldo A, Scattolini R (2014) Logic control design and discrete event simulation model implementation for a de-manufacturing plant. Automazione-plus on-line J
26. Terkaj W, Urgo M, Andolfatto D (2017) Answer set programming for modeling and reasoning on modular and reconfigurable transportation systems. In: Proceedings of the 2017 federated conference on computer science and information systems
27. Cataldo A, Scattolini R (2016) Dynamic pallet routing in a manufacturing transport line with model predictive control. IEEE Trans Control Syst Technol 24:1812–1819
28. Cataldo A, Perizzato A, Scattolini R (2014) Modeling and model predictive control of a de-manufacturing plant. In: IEEE multi-conference on systems and control
29. Cataldo A, Perizzato A, Scattolini R (2015) Production scheduling of parallel machines with model predictive control. Control Eng Pract 28–40
30. Hu T, Cibrario Bertolotti I (2015) Overhead and ACK-induced jitter in Modbus TCP communication. In: Proceedings of the 1st IEEE international forum on research and technologies for society and industry (RTSI)
31. Cena G et al (2014) Design, verification, and performance of a Modbus–CAN adaptation layer. In: Proceedings of the 10th IEEE international workshop on factory communication systems (WFCS), pp 1–10

32. Cataldo A, Perizzato A, Scattolini R (2014) Management of a production cell lubrication system with model predictive control. In: IFIP international conference advances in production management systems, pp 131–138
33. Cataldo A, Scattolini R, Tolio T (2015) An energy consumption evaluation methodology for a manufacturing plant. CIRP J Manuf Sci Technol 11:53–61
34. Terkaj W, Tolio T (2019) The Italian flagship project: factories of the future. In: Tolio T, Copani G, Terkaj W (eds) Factories of the future. Springer