International Journal of Automation and Computing 17(3), June 2020, 464-478
DOI: 10.1007/s11633-018-1131-1

A Practical Approach to Representation of Real-time Building Control Applications in Simulation

Azzedine Yahiaoui

Center for Building & Systems, Eindhoven University of Technology (TU/e), PO Box 513, Eindhoven 5600MB, Netherlands

Abstract: Computer based automation and control systems are becoming increasingly important in smart sustainable buildings, often referred to as automated buildings (ABs), in order to automatically control, optimize and supervise a wide range of building performance applications over a network while minimizing energy consumption and associated green house gas emission. This technology generally refers to building automation and control systems (BACS) architecture. Instead of costly and time-consuming experiments, this paper focuses on development and design of a distributed dynamic simulation environment with the capability to represent BACS architecture in simulation by run-time coupling two or more different software tools over a network. This involves using distributed dynamic simulations as means to analyze the performance and enhance networked real-time control systems in ABs and improve the functions of real BACS technology. The application and capability of this new dynamic simulation environment are demonstrated by an experimental design, in this paper.

Keywords: Distributed dynamic simulation, networked control systems, building performance applications, smart buildings, building automation and control systems (BACS) architecture.

1 Introduction

With the impact of recent technological advances on computers and communication protocols, a computer-based automation and control system is frequently used to replace so-called hardwired controls with control strategies implemented in software. Such a technology in automated buildings (ABs), named also smart or intelligent buildings, generally refers to building automation and control systems (BACS) architecture. In order for BACS technology to adapt ABs to changing requirements such as the needs of the occupants and environmental changes in a building by control systems design, experiments or similar analyses must be conducted to improve the operational integrity and the automation of heating, ventilation and air-conditioning (HVAC) equipment and lighting components in ABs[1]. However, experiments are time consuming as they require at least 24 hours to obtain the results and because realizing BACS architecture in a real building is expensive. For this reason, this paper deals with the development and implementation of a distributed dynamic simulation environment with the capability to similarly represent BACS architecture in simulation by run-time coupling two or more different software tools over a network.

The current situation is that representing BACS architecture in simulation by means of a single software tool or two different simulation tools running on a single computer is complex and even more challenging because it requires taking into account the physical distance of a network in control loops so as to emulate the real-world applications of BACS as closely as possible. It is also important to represent the BACS architecture in simulation by distributing multiple different software tools at run-time over a network to enable assessment of distributed building control applications by predicting the overall effect of innovative control strategies in ABs (see e.g., [2, 3]). As there exists a software tool very advanced in control modelling, i.e., Matlab/Simulink, and a domain based building performance simulation such as environmental system performance-research version (ESP-r), the combination of both over a network would result in a rational design of distributed control and building performance simulations by means of experimental design for representing, as in a similar way, the BACS architecture in simulation. This has an objective to assist architectural use and design of distributed control applications in ABs in the form of combined simulations in heterogeneous systems (i.e., different operating systems with different access technologies).

Distributed simulations involving two or more different software tools (or diverse applications) at run-time provide the ability to exchange data and events in a distributed and co-operative way. This way of run-time coupling diverse applications over a network offers several advantages such as exploiting different modes of com-

Research Article
Manuscript received March 18, 2017; accepted April 25, 2018; published online August 3, 2018
Recommended by Associate Editor Zhi-Jie Xu

munication (including synchronous, asynchronous and partially asynchronous), increasing the execution speed, running applications on heterogeneous environments, and combining the relative strengths of different tools. However, some advantages can be lost when a run-time coupling is developed only to run in one environment and to exchange data in synchronous mode. For example, previous and ongoing work by others, especially with the development of run-time coupling for the purpose of enhancing building performance simulation, comprise coupling between lighting and building energy simulation (e.g., [4]), coupling between computational fluid dynamic programs and building energy simulation[5], and coupling between systems and building energy simulation[6]. However, these approaches are limited to a particular application, and often based on the coupling of two simulation tools running on the same machine. Besides these approaches, some other works have been developed based on the use of libraries such as: 1) building controls virtual test bed (BCVTB) library used to couple different simulation tools for co-simulation[7], 2) co-simulation between building performance and energy systems based on BCVTB[8], and 3) modelica buildings library used to support simulation models of building energy and control systems[9]. Still these efforts are insufficient as they are based on coupling of two simulation tools in synchronous mode, for which they are inefficient to be used to represent distributed building control operations such as multi-agent systems (MASs) in simulation. Because they also do not take network dynamics into account, these are inadequate to explore real-time control applications in a simulation the same as they are performed in real BACS architecture. For this reason, a novel middleware for distributed dynamic simulations by run-time coupling between a software for control systems design and one or more specific building performance simulation software tool(s) over a network is developed for a more general and wider applicability.

2 Development and implementation

2.1 Description of BACS architecture

ABs are a class of buildings that are automatically supervised and controlled by or from a central computer-based monitoring and control system such as distributed control system (DCS) architecture, or more specifically BACS architecture. Therefore, BACS is an example of a DCS because it uses a computer-based control system to replace so-called hardwired controls with control strategies implemented in software. The basic function of BACS is to automatically monitor and control a wide range of building performance applications including HVAC equipment, lighting components and other tasks such as access control, energy management, and fault diagnoses in a building or a group of buildings over a network. While this technology has several advantages, it also brings inevitable problems due to the network. Fig. 1 shows a complete BACS architecture that can be described at four main levels[2, 10, 11]:

1) The management level consists of a central computer used for managing and analyzing data, communicating with external systems, and operating building equipment and components.

2) The network level consists of an open protocol connected to the network through routers used for data exchange between the central computer and substations (or terminals).

3) The automation level consists of one or more substations used for interfacing building HVAC equipment and lighting components to the network.

4) The field level represents the low level where building HVAC equipment and lighting components (i.e., sensors and actuators) and final users are located.

Because BACS uses a network for data exchange between a central computer and substations, this can degrade both the performance and the stability of HVAC equipment and lighting components in buildings. The most common and straightforward way to evaluate such problems without a full-scale implementation of BACS architecture is by a modelling and simulation approach. Therefore, successful development and application of BACS require a scalable simulation platform that supports evaluation and verification of different control and network algorithms. As a consequence, a distributed simulation environment was developed and implemented mainly for BACS to simultaneously simulate building control applications and communication network dynamics.

2.2 Development and design of run-time coupling

The design of run-time coupling between Matlab/Simulink and one or more ESP-r(s) begins with the definition requirements, and proceeds eventually to conceptual design of the run-time coupling by means of trade-off analysis, and then to detailed design of every part of the run-time coupling being developed. Therefore, a set of requirements were first identified and set forth as the basis for the development and design of the run-time coupling. However, these requirements must then be taken into consideration at the early-stage of development. Among the most important of these requirements are[4, 12]:

1) The ability for run-time coupling between Matlab/Simulink and one or more ESP-r(s) to run on a heterogeneous network as on Windows and Unix operating systems (OS).

2) The ability for run-time coupling between Matlab/Simulink and one or more ESP-r(s) to support data exchange over a network in either unidirectional or bidirectional way.

3) The ability for run-time coupling between Matlab/Simulink and one or more ESP-r(s) to support different data exchange formats including ASCII, binary and extensible markup language (XML).

4) The ability for run-time coupling between Matlab/Simulink and one or more ESP-r(s) to support different communication modes including synchronous, asynchronous, partially synchronous (or asynchronous).

5) The possibility for run-time coupling between Matlab/Simulink and ESP-r to enable simulations with either a real building (e.g., building emulator) or a control test-rig (e.g., hardware in the loop testing), in which the inter-process communication (IPC) must then be platform independent.

After evaluating and selecting the most suitable solution among a number of possible options, using network (or internet) sockets has been chosen and approved by [2, 13, 14] as the best means of implementing run-time coupling between Matlab/Simulink and one or more ESP-r(s) since they meet all the requirements of the run-time coupling, including those described above, and they can also be used to represent in simulations real-time building control implementations over a network, as shown in Fig. 1.

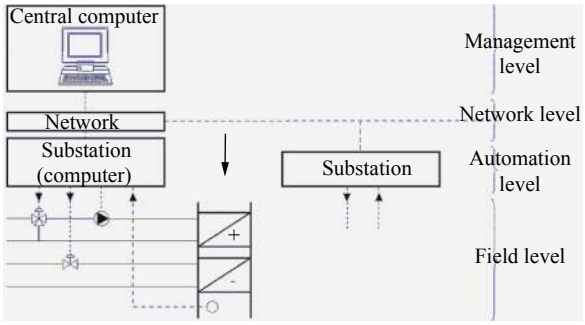


Fig. 1 BACS architecture

Network sockets are an IPC mechanism that is used for run-time coupling between ESP-r and Matlab/Simulink to support modelling of a building model and its external control systems separately. Both the building model and its control systems can be located on either the same machine or different separate machines running different OS such as Unix and MS-Windows connected to a network. They also work together by exchanging data in a common format including ASCII, binary and extensible markup language (XML) over a network, and by supporting communication modes such as synchronous, asynchronous, and partially synchronous. Fig. 2 illustrates the proposed approach to run-time coupling between Matlab/Simulink and ESP-r.

Run-time coupling is implemented in such a way to facilitate data exchange between Matlab/Simulink and ESP-r when they are concurrently operating either on the same machine or to increase the speed of simulations, on

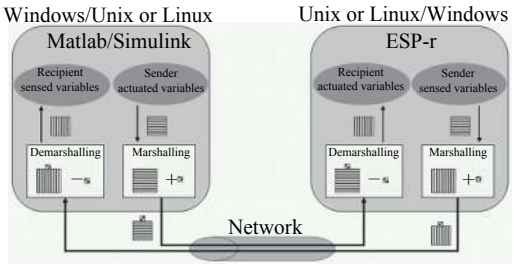


Fig. 2 Run-time coupling between Matlab/Simulink and ESP-r

separate machines connected by a network. In addition, when Matlab/Simulink and ESP-r are located on different machines running over different OSs and/or using different data formats by initiating network protocols, such as LonWorks and Bacnet, run-time coupling can be run to support portability and distributed dynamic simulations over a heterogeneous network (i.e., on different machines with different OSs and/or different data format protocols). For this reason, in this work different methods for marshalling and demarshalling (or unmarshalling) data over a network were implemented within run-time coupling to convert data (i.e., sensed or actuated variables) into a form of external network representation and then back to their native format before being accessed by a building model and its control systems, respectively.

2.2.1 Detailed design of run-time coupling

In order to implement run-time coupling between ESP-r and Matlab/Simulink with network sockets, the C/C++ programming language was used of which socket libraries were originally implemented. As neither Matlab/Simulink nor ESP-r has simple interfaces with socket application programming interfaces (APIs), a detailed design of the run-time coupling is proposed and implemented with the objective of interfacing socket APIs to both ESP-r and Matlab/Simulink, as shown in Fig. 3.

As the detailed design shown in Fig. 3 is based on the idea that run-time coupling should be delivered in such a way with no or minor user interferences, the details of the parallel and distributed computations are then hidden to users, while necessary information, such as the port number and IP address of the Matlab/Simulink location, is provided through user interfaces in order to create sockets that allow one or more ESP-r(s) to exchange data with Matlab/Simulink. As both datagram and stream sockets are supported, the same sockets type should be selected on both sides of the run-time coupling mechanism. By such means, this mechanism can serve as a virtual interface that supports portability between heterogeneous platforms, enables distributed dynamic simulation of BACS, and achieves a higher level of interoperability by using a common middleware platform rather than a non-distributed communication system. Besides this middleware platform, two data encoding methods are integrated to improve interoperability when Matlab/Simulink and ESP-r are running in heterogeneous environments

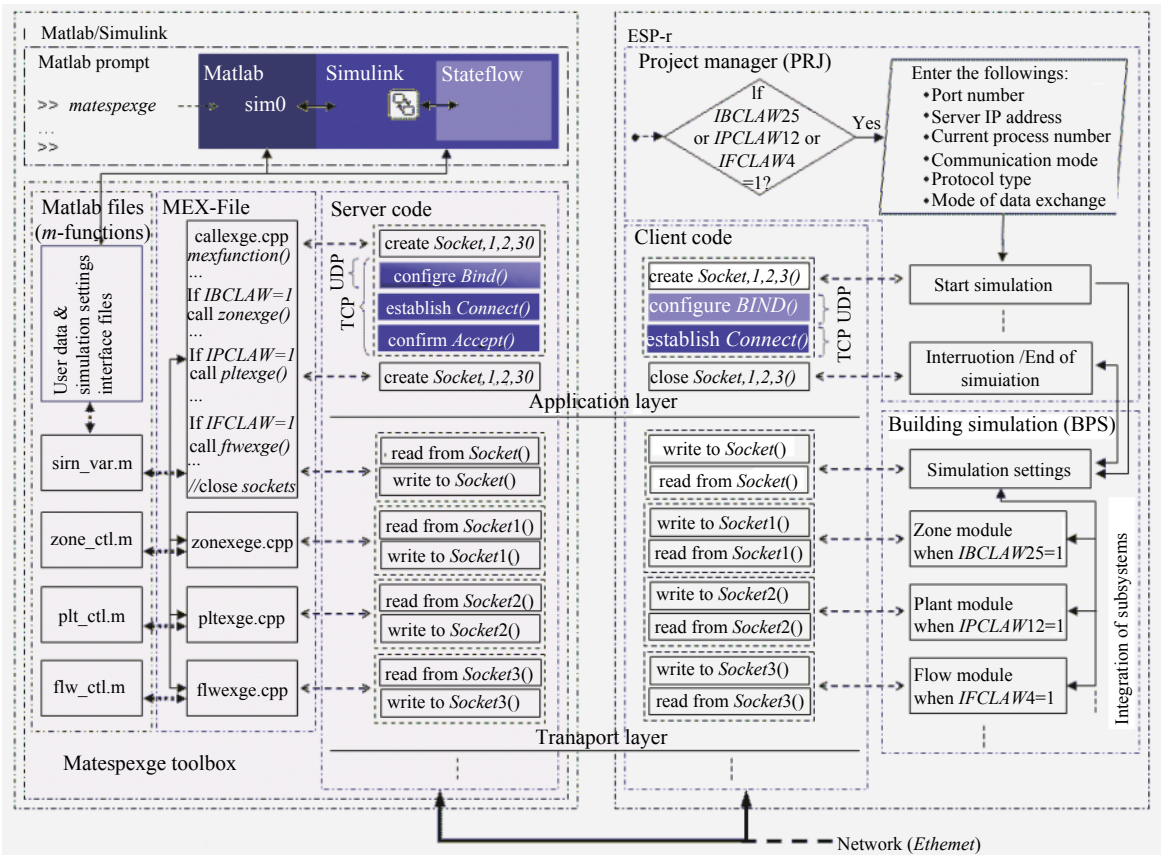


Fig. 3 Detailed design of run-time coupling between ESP-r and Matlab/Simulink

and to increase the speed of data exchange between a building model and its control systems when both ESP-r and Matlab/Simulink are running on distant machines connected to a network. The first method of data encoding consists of implementing two modes (ASCII strings and binary codes) to meaningfully and accurately exchange data between a building model and its control systems over a network. The second method consists of using a set of web-based interoperability specifications, such as web-services based on XML in order to enable building models and their control systems to exchange data with a high level of interoperability between ESP-r and Matlab/Simulink while representing different network technologies, such as BACnet and LonWorks protocols, in a simulation.

The right side of Fig. 3 shows how ESP-r and its integrated subsystems (zone, plant, and flow modules) are bound to socket APIs, while the left side of Fig. 3 details how the "matespexge" toolbox is implemented to bind socket APIs to Matlab/Simulink. On the ESP-r side, several subroutines, such as IBCLAW25, IPCLAW12, and IFCLAW4, are added and implemented in a building simulator (BPS) to allow direct data transmission between the subroutines and their parallel programs, which are implemented in the matespexge toolbox during simulation. Other subroutines, such as that for a test function, are implemented in a project manager (PRJ) to determine whether any of the building simulator (BPS) integrated modules invokes external control system that should be remotely processed from Matlab/Simulink. If one of these integrated modules occurs, a graphical user interface containing data regarding the port number, server IP address, current process number, communication mode, protocol type, and mode of data exchange appears so that the user can modify and choose specific settings. Initially, these settings are set to default values and correspond exactly to those specified in the matespexge toolbox. Changing these settings is possible, although Matlab/Simulink and ESP-r must use the same entries to ensure their connection. On the Matlab side, the matespexge toolbox is designed with graphical interfaces in order to allow users to differentiate between sensed and actuated variables that should be exchanged with ESP-r. Executing the matespexge toolbox at the Matlab prompt results in a graphical user interface appearing and displaying the machine IP address of the Matlab/Simulink location, which should match the ESP-r IP address. As Matlab is the server of ESP-r client(s), executing first the matespexge toolbox is essential before initiating simulation in ESP-r.

As shown in Fig. 3, run-time coupling between ESP-r and Matlab/Simulink is designed in a layered model, where the upper open systems interconnection (OSI) layers resolve different aspects of the communication pro-

cess. As ESP-r consists of legacy Fortran codes, the programs that operate building simulation and those that initiate and terminate it are positioned and executed in different steps (i.e., the first programs are executed every time-step during the simulation period, while the second programs are executed only when initiating and terminating the simulation). Placing functions that open and close sockets in the first programs would certainly incur significant delays while exchanging data with Matlab/Simulink during simulation, and opening and closing sockets at every time step could lead to computer failures.

The main advantages of this developed run-time coupling are that it permits any simulation of a building model and its control systems to be built separately using ESP-r and Matlab/Simulink, respectively, and that it provides the preferred means to handle interoperability tasks, especially cross interdisciplinary data integration and exchange between ESP-r and Matlab/Simulink with no or minor user interferences. Therefore, it requires only modelling a building model on ESP-r and its control systems on Matlab/Simulink, and then indicating their interfaces by specifying the port-numbers, modes of exchange, or variables that they will use to import or export data to or from each other.

2.2.2 Interfacing client socket to ESP-r

Because ESP-r, e.g., [15], is almost completely written in Fortran programming language and socket application programming interface (APIs) can only be implemented in programming languages such as C/C++, mixed-language programming using Fortran and C++ must be used to interface between Fortran and C/C++ programs[16]. Therefore, mixed-language programming is used to develop and implement an approach combining a Fortran common block with global C/C++ extern data structures (or extern structs) of the same name in order to enable the addition of new variables that need to be exchanged with Matlab/Simulink without making large modifications in the existing programming codes.

ESP-r was modified and extended to enable users to obtain data on sensed and actuated variables in the external control systems of building zones, plant components, and/or mass-flow networks, and to choose settings (including server IP address, port number, current process number, network protocol, communication mode, and data-exchange format) for run-time coupling. The added Fortran subroutines that exchange data with Matlab/Simulink and functions indicate when initiating and ending simulations are combined together with socket APIs of the C/C++ client code separately. The C/C++ client code was developed in a hierarchical way in order to support all possible combinations of exchanged variables and settings that a user could choose in run-time coupling with Matlab/Simulink. Compiling the modified and extended ESP-r code together with the socket APIs of the C/C++ client code generates executable ESP-r, respectively, and allows ESP-r to run as a client process.

2.2.3 Interfacing server socket to Matlab/Simulink

In [17], there is a built-in utility called Matlab EXcutable (MEX) that is often used to convert C or C++ programs to a MEX format. The original sense of the Matlab/Simulink word represents two different environments, which are a high-level technical programming language and a graphical block-diagram interface. Depending on which environment is interfaced, two main approaches can be used to link external programs written in C/C++ code:

1) For Matlab, MEX-files are used with dynamically linked programs that, when compiled, can be called from within Matlab in the same way as M-files or built-in functions. In case we need to deal with Simulink, the links can be established between each other by just using "sim" functions.

2) Practically the same procedure is adopted by Simulink, although MEX S-functions are used with dynamically linked programs that, when compiled, can be called from within a Simulink block diagram. However, when there is a need to deal with Matlab, the link should be done via M-file S-functions that are more complicated than using a straightforward "sim" function.

The first approach is preferable not only because it is less complex than the second approach but also because it offers more advantages, such as: 1) the ability to manage a high number of exchanging variables simultaneously, 2) the versatility needed to meet the requirements of run-time coupling, and 3) the ability to implement functionalities that are not accessible by M-file S-functions.

Although the MEX-files were originally designed to allow the inclusion of external routines written mainly in C/C++, they are also capable of integrating external shared libraries, such as socket APIs, into Matlab. For these reasons, a MEX-file was used for the development and implementation of the "matespexge" toolbox.

By combining MEX-file functions and socket APIs, access from ESP-r to Matlab and Simulink functionalities, especially to the application toolboxes for advanced control systems, is realized by just invoking the name "matespexge" from the Matlab prompt. Once the matespexge toolbox has been executed, a graphical user interface including icons and menus will display and provide the dialogue for users to create M-files to remotely control a building zone, plant, and/or flow model as built on ESP-r accordingly. Further access from these M-files to Simulink can be obtained by using "sim" functions, although access from Simulink to Stateflow should be obtained by incorporating a Stateflow block in the Simulink block diagram. Moreover, these M-files include Matlab functions that contain the left- and right-hand arguments with which the MEX-file is invoked. Therefore, the matespexge toolbox was designed with the use of MEX-files that include facilities for enabling run-time coupling between Matlab/Simulink and one or multiple ESP-r(s). After compiling the matespexge toolbox, a dy-

namic executable file is generated with an extension corresponding to the OS over which Matlab/Simulink is running. Within the implementation of the matespexge toolbox, the external routines that specifically exchange data with subroutines for building zone, plant, and flow network modules of ESP-r are encapsulated into a single MEX-file. A global identifier is also integrated to determine which building zone, plant, and/or flow network model will exchange data with the created control file. Due to this fact, the user must provide valid information (i.e., as stated in ESP-r) on the input interface. In addition, as Matlab is an interactive tool, the handling callbacks from ESP-r are ensured by default in order to access Matlab/Simulink as a computational engine. For these reasons, the matespexge toolbox is designed in such a way to let Matlab/Simulink operate as a server process when its created control files are invoked by one of the three ESP-r modules. Because stateflow can be used together with Simulink for the simulation of MASs, the use of the matespexge toolbox becomes essential for enabling the integration of advanced control systems, such as hybrid systems and MASs in building performance simulation. It enables a user to interactively build, test, and simulate distributed applications between ESP-r and Matlab/Simulink, even when both software tools are running on separate and different OSs. Therefore, it is a key solution in enabling the analysis of multi-variable control systems of building performance applications (or operations) that had previously not been feasible.

2.3 A practical approach to representing BACS architecture in simulation

Although representing BACS technology in simulation, as shown in Fig. 1, is difficult if not impossible, this design of run-time coupling between multiple ESP-r(s) and Matlab/Simulink can allow identifying practical solutions for the integration of advanced control systems into building performance simulation, and improving distributed control applications such as planning and coordination of control actions, in ABs for better control and operation. As one of constrained BACS architecture has network-induced time delays, distributed simulations are required to analyze and simulate both the performance and stability of building HVAC equipment and lighting components in ABs. Hence, the need for distributed simulations originates from the fact that BACS requires the study of both control theory and communication networks in design architecture[18].

By assuming that Matlab/Simulink represents a central computer and ESP-r, a terminal in a similar way to BACS architecture, the IPC meshcanism used for run-time coupling Matlab/Simulink and ESP-r is thus designed to support cooperative applications through an interoperable middleware that is involved in facilitating the interface of any building model built in ESP-r with its external control system modelled in Matlab/Simulink, as shown in Fig. 3. This has for an objective to simplify data management and distribution over a network, provide for the independence and transparency of data exchange between building models and their control systems, and allow web-services to be highly portable in distributed simulations, in a similar way as the real-time control applications are represented in BACS architecture. Therefore, this work has enhanced the traditional approach to run-time coupling between ESP-r and Matlab/Simulink as shown in Fig. 2, with the addition of involving more ESP-r(s) to represent BACS architecture in simulation. Fig. 4 illustrates a more practical approach to distributed control and building performance simulation by run-time coupling multiple ESP-r(s) and Matlab/Simulink over a network.

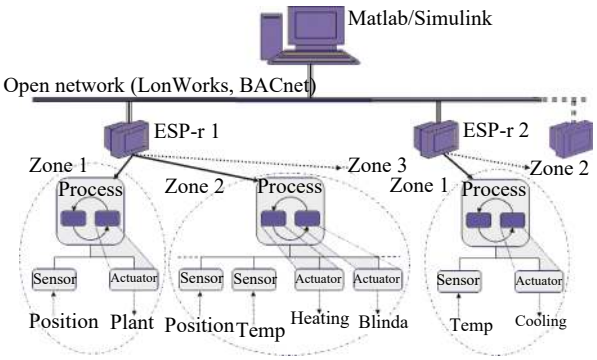


Fig. 4 A practical approach to run-time coupling Matlab/Simulink with multiple ESP-r(s)

As shown in Fig. 4, the framework that is implemented to support distributed and parallel simulations by run-time coupling one or more ESP-r(s) with Matlab/Simulink over a network occurs practically the same as BACS architecture. This framework was developed using multi-threads to optimally handle and connect multiple ESP-r(s) to Matlab/Simulink, for which an application similar to the one used in BACS architecture can at once run across several machines distributed over a network[1]. In a similar way to BACS architecture, each ESP-r in the framework must be used to simply model building zone(s), plant system(s) and/or flow network(s), while Matlab/Simulink must be used to model all their remote control systems.

2.4 Run-time coupling two or more ESP-r(s) with Matlab/Simulink

Of the many possible ways to run-time couple more than one ESP-r with Matlab/Simulink at the same time, the portable operating system interface (POSIX) standard for threads has been the most widely adopted[12]. The use of POSIX threads is very advantageous because of its standardization, flexibility, and portability, as well as fact

that POSIX threads provide a standardized programming interface for the dynamic creation and destruction of threads (i.e., sub-threads). It also enables using the same port and a single shared address space to make Matlab/Simulink accessible to all ESP-r(s) connections that are handled on the network. By using a single address space abstraction, it is however possible to avoid the overhead inherent to data exchange and provide better support for concurrency, parallelism, and consistency of data exchange in run-time coupling of Matlab/Simulink and multiple ESP-r(s) with substantial ease. To nearly represent BACS architecture in simulation, the previous approach, shown in Fig. 2, was therefore extended with a capability to run-time couple one or more ESP-r(s) with Matlab/Simulink. This capability was developed with multi-threads in conjunction with C^{++} codes to support parallel and distributed control and building performance applications between multiple ESP-r(s) and Matlab/Simulink in the same simulation environment. Within this capability, all ESP-r(s) should share the same connection as Matlab/Simulink and be able to run either on the same machine as Matlab/Simulink or on separate machines connected to a network. Each time a new ESP-r is connected to Matlab/Simulink, its specific thread is thus created by the matespexge toolbox so as to avoid conflicts and data inconsistencies with other concurrent ESP-r(s) participating in the same simulation environment. As all participating (or connected) ESP-r(s) exchange data with the same Matlab/Simulink, any ESP-r can access all the global variables exchanged by Matlab/Simulink via its specific sub-thread. Fig. 5 illustrates how run-time coupling is implemented between Matlab/Simulink (i.e., matespexge toolbox) and multiple ESP-r(s) using POSIX threads.

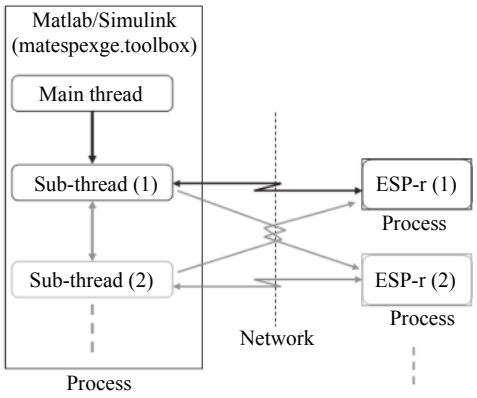


Fig. 5 Conceptual view of multi-threading Matlab/Simulink with multiple ESP-r(s)

As illustrated in Fig. 5, the matespexge toolbox is implemented in such a way that one or more ESP-r(s) can connect and interact with Matlab/Simulink concurrently. The number of ESP-r(s) to run-time couple to Matlab/Simulink depends on the application, varying from one (1) to nine (9) ESP-r(s) simultaneously. This implementation is fairly complex as it requires that the main thread of the matespexge toolbox accept incoming connections and create one ESP-r sub-thread for each ESP-r connection that is handled. These ESP-r sub-threads are a part of the matespexge toolbox used by shared data structures to communicate with their all parallel connected ESP-r(s). Because the matespexge toolbox can run-time couple with multiple ESP-r(s),

1) each data exchange to/from ESP-r is handled by the corresponding ESP-r sub-thread on the matespexge toolbox side;

2) each ESP-r sub-thread can send data to other connected ESP-r(s) by accessing the shared data structure that contains their references;

3) the sockets connecting the matespexge toolbox to each ESP-r can be retrieved through this shared data structure.

Consequently, all interactions between ESP-r(s) and Matlab/Simulink occur via the matespexge toolbox, where every ESP-r is handled by a particular sub-thread. Also this toolbox is implemented with call-back methods to allow remote control systems (i.e., control systems modelled on Matlab/Simulink) to be invoked as they receive data from their corresponding building models built on one or more ESP-r(s). Because building models built on multiple ESP-r(s) can interact with each other via the matespexge toolbox, their corresponding remote control systems can also interact with each other on the Matlab/Simulink side. The main objective of using this approach is to represent the BACS architecture in simulation and enable unrelated remote control systems, particularly advanced control systems such as MASs, to communicate with each other when their corresponding building models are built on seperate ESP-r(s). In effect, permitting control systems – particularly MASs – to communicate with each other while remotely regulating building zone, plant, and mass-flow models built on diverse ESP-r(s) connected to a network results in the design and development of advanced building control applications that had previously not been feasible, such as:

1) the use of coordinated and interconnected control systems, especially MASs, to better operate and regulate building HVAC equipment and lighting components in ABs;

2) the use of self-adapting control systems to react to climate changes, the addition or removal of equipment in a building, or building plant variations;

3) the use of self-upgrading control systems to meet occupant needs when damping effects or changes are critical factors in the functioning of the systems.

2.5 Communication modes in run-time coupling

Because the main feature distinguishing a distributed simulation from a standalone simulation (or a sequential simulation) is the method of advancing simulation time-

steps, run-rime coupling between Matlab/Simulink and ESP-r was implemented with all the options accessible from the user interface to specify the number of simulation time-steps per hour and to determine whether the simulation should run in a synchronous, an asynchronous, or a partially synchronous mode.

2.5.1 Synchronous mode

Synchronous mode is used when ESP-r and Matlab/Simulink are run-time coupled and synchronized with the same number of simulation time-steps in execution. When either ESP-r or Matlab/Simulink must wait for incoming data from the other, the number of simulation time-steps is defined by ESP-r, which is the client for Matlab/Simulink, the server. Fig. 6 shows how ESP-r and Matlab/Simulink must wait for each other and exchange data by run-time coupling at several predetermined time steps for the completion of their computations.

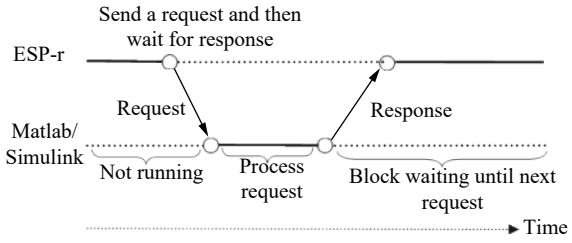


Fig. 6 Run-time coupling between ESP-r and Matlab/Simulink in synchronous mode

Because both ESP-r and Matlab/Simulink are executed sequentially, the exchange of data in this mode blocks the entire simulation at each predetermined time-step until the data exchange has been totally completed. Therefore, when Matlab/Simulink and ESP-r are run-time coupled in synchronous mode, the time constraints of scheduled transitions must be satisfied by such means as adjusting the timing of a control loop for several applications.

2.5.2 Asynchronous mode

Asynchronous mode is used when ESP-r and Matlab/Simulink are run-time coupled and processing separately from each other, and not synchronized totally. As such, neither ESP-r nor Matlab/Simulink must wait for incoming data from the other and can continue their computations with the existing data, although the data might be outdated, until the updated data become available for computation. Running distributed simulations in asynchronous mode can be positive in some cases because ESP-r and Matlab/Simulink can be computed with different numbers of simulation time-steps, although in other cases the accuracy of obtained simulation results cannot be ensured. Fig. 7 shows a case where the time-step of either Matlab/Simulink or ESP-r differs from that of the other.

The asynchronous mode is difficult to program because it requires that Matlab/Simulink and ESP-r be run-

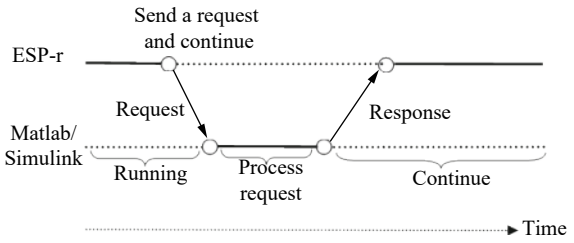


Fig. 7 Run-time coupling between ESP-r and Matlab/Simulink in asynchronous mode

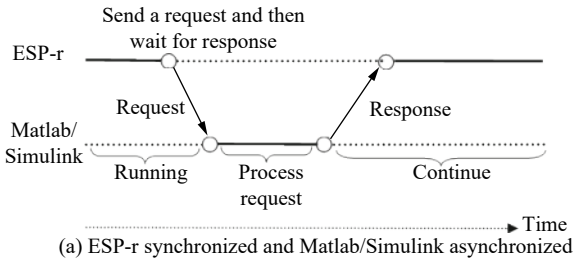
time coupled in a chaotic manner; i.e., in such a way that neither Matlab/Simulink nor ESP-r must wait for incoming data from the other but instead proceed with its computation until the common task is fulfilled. As run-time coupling between Matlab/Simulink and ESP-r is implemented with network sockets, the asynchronous communication is characterized by the fact that the client and server programs integrated in ESP-r and Matlab/Simulink, respectively, contain functions that change sockets to non-blocking mode. Moreover, such changing signal is returned immediately when the operations reading data from the sockets are invoked by another to indicate that there are no data to be received, see e.g., [19].

The asynchronous mode was developed in such a way that Matlab/Simulink began computing once it received the first data from ESP-r. After data were received, ESP-r and Matlab/Simulink computed independently from each other, and when no data were available to be received, both continued computing using the most recently received data. Using asynchronous mode allows run-time coupling between Matlab/Simulink and ESP-r to handle existing data in a manner that may significantly reduce the execution time, see e.g., [20]. As it imposes no constraints on the control performance, this mode can be used in situations where network-induced time delays are unpredictable. When synchronous mode is used in these situations, Matlab/Simulink and ESP-r may have to wait for incoming data for an extended time, which could result in very low or even impractical computation efficiency. Therefore, computation in asynchronous mode can be useful for the simulation of large building control applications, such as supported by BACS technology, although it is much more difficult to parallelize and distribute efficiently due to various independencies in run-time coupling between Matlab/Simulink and multiple ESP-r(s).

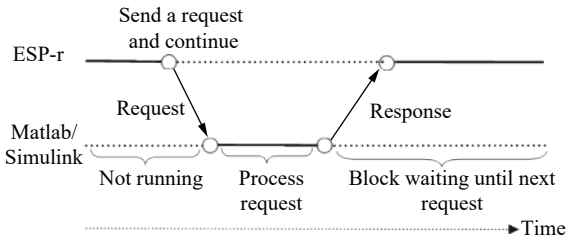
2.5.3 Partially synchronous mode

The partially synchronous mode is used when ESP-r and Matlab/Simulink are run-time coupled in partially synchronized (or partially asynchronized) mode, which partly imposes time restrictions on the synchronization of their events. However, the exchange of data between Matlab/Simulink and ESP-r by run-time coupling does not occur in lock-time-step, as it does in synchronous mode. When using the partially synchronized mode, it is

assumed that the computation time-step and data delivery time of either Matlab/Simulink or ESP-r is between the upper and lower bounds. Therefore, as a partially synchronous mode lies between the synchronous and asynchronous modes, two possibilities are available for run-time coupling Matlab/Simulink and ESP-r in partially synchronized mode: 1) when ESP-r is running in synchronous mode and Matlab/Simulink in asynchronous mode, and 2) when ESP-r is running in asynchronous mode and Matlab/Simulink in synchronous mode. Fig. 8 shows both cases of the partially synchronous mode.



(a) ESP-r synchronized and Matlab/Simulink asynchronized



(b) ESP-r asynchronized and Matlab/Simulink synchronized

Fig. 8 Run-time coupling between ESP-r and Matlab/Simulink in partially synchronized mode

The partially synchronized mode, when ESP-r is running in asynchronous mode and Matlab/Simulink is in synchronous mode, represents the most realistic communication characteristics of BACS technology (i.e., such the real-time control applications are performed in BACS). Although time delays associated with network (i.e., with communication) are difficult to predict in BACS architecture, these delays usually have an upper bond, mainly when data are exchanged over a network. Hence, use of the partially synchronous mode leads to less uncertainty than does the fully asynchronous mode, but is very difficult to program due to the timing differences between Matlab/Simulink and ESP-r, see e.g., [21].

3 User interfaces for run-time coupling of ESP-r and Matlab

The user interfaces were developed and implemented on both sides of the run-time coupling in order to facilitate its use (i.e., the setting up of a connection session and exchanging variables). Fig. 9 depicts the user interfaces for run-time coupling settings on the side of ESP-r.

On the ESP-r side, the user-interface of the run-time

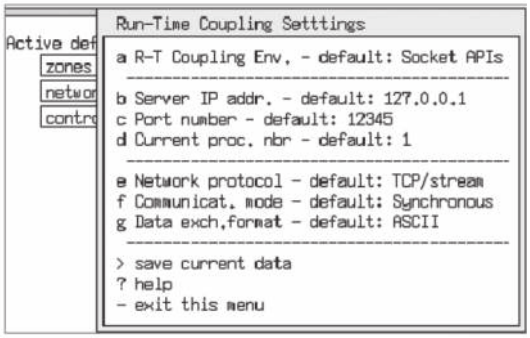


Fig. 9 User interfaces for run-time coupling settings: ESP-r side

coupling settings menu can be displayed only if one of the three control laws (zone, plant and systems, and flow network) chooses "Calling Matlab/Simulink" as a control system/strategy, in which there will be requiring using then external (or remote) control systems, either existing or newly designed, modelled on Matlab/Simulink. Fig. 10 depicts the user interfaces for run-time coupling settings on the side of Matlab/Simulink.

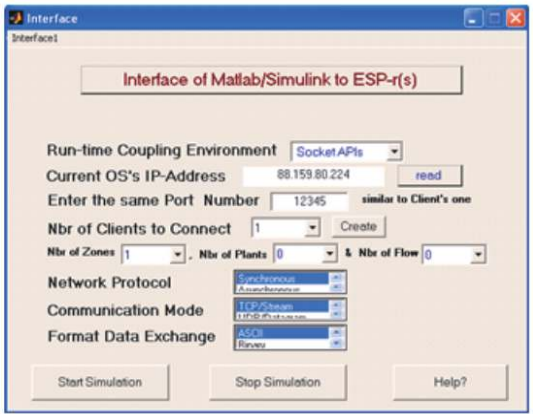


Fig. 10 User interfaces for run-time coupling settings: Matlab side

On the Matlab side, the user interface of the run-time coupling settings menu is displayed by typing "≫ matespexge" at the Matlab prompt, as shown in Fig. 11. After providing and choosing the necessary entries for run-time coupling with ESP-r, and invoking the creation of one of the three control laws (zone, plant and systems, and flow networks), a sub user interface is then displayed. Fig. 10 shows this developed and implemented sub user interface on the Matlab side to set up which sensed and actuated variables should be exchanged with ESP-r by run-time coupling.

The manner of entering all the necessary data required for run-time coupling in Matlab side is rather similar to that in ESP-r. In addition, the sub user interface on the Matlab/Simulink side was designed in a way as is typically done in ESP-r. All possible sensed variables (inputs) and actuated variables (outputs) for control of

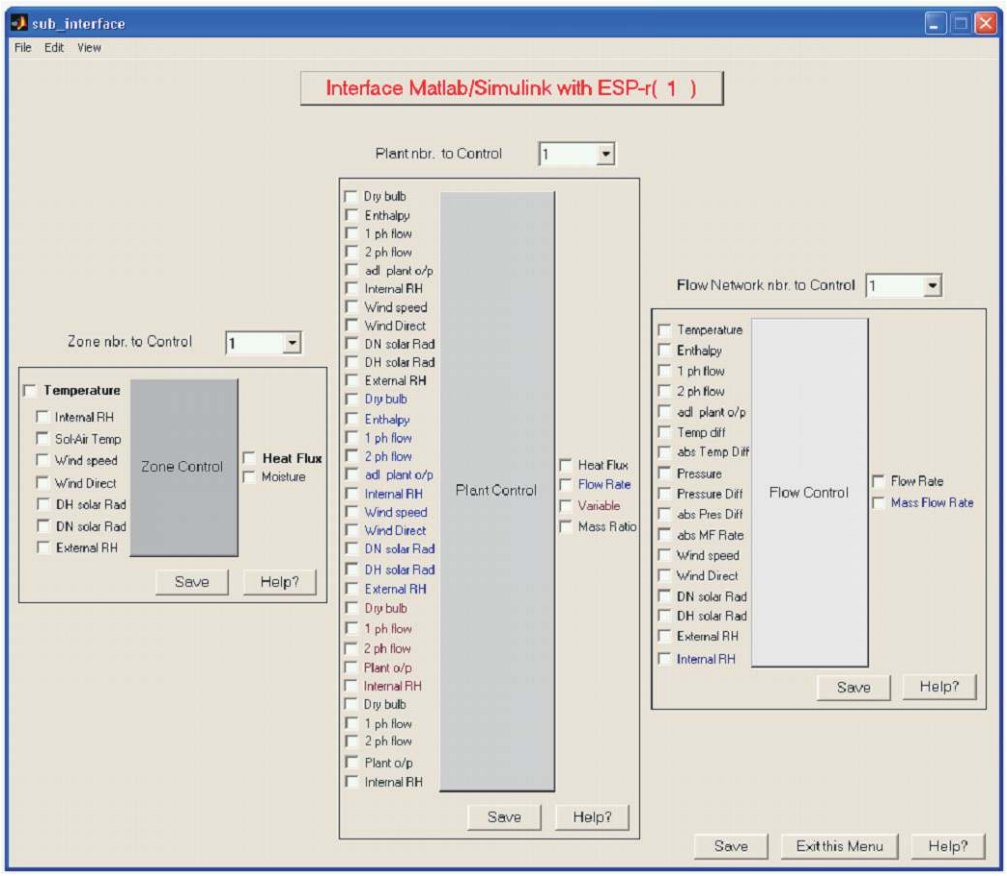


Fig. 11 Matlab sub user interface for zone, plant and systems, and flow network control variables with ESP-r by run-time coupling

zones, plant and systems, and flow components that ESP-r can exchange by run-time coupling with Matlab/Simulink are as shown in Fig. 11.

4 Experimental design

The resulting experimental design, in this study, consists of representing important functions of BACS technology in simulation, and comparing simulation results obtained in different modes of communication using at least two instances of ESP-r(s) by run-time coupling with Matlab/Simulink over a network, as referred to BACS architecture, shown in Fig. 1. One way of doing this was to run ESP-r(s) and Matlab/Simulink on different machines over different OSs like Unix, Cygwin, and Windows. These machines were a Sun Blade Workstation (or Server), a laptop PC, and an office PC, respectively. ESP-r1 was installed over the Unix 10 (or Solaris 10) OS on the Sun sever, ESP-r2 was installed over the Cygwin 6.1 over the Windows XP OS on the laptop PC, and Matlab was installed over the Windows 7 OS on the office PC. The Sun server and laptop PC were connected to the network by an Ethernet LAN cable at a speed of 10 MBps, and at distance of about 7 m from each other, whereas the office PC was connected to the network by an Ethernet LAN cable at a speed of 100 MBps, and distanced from the two other machines (Sun blade workstation and office PC) by about 2 km. The compilers used for Fortran and C/C^{++} programs were of the GNU public license on Unix and of the Visual Studio 2010 on Windows. Fig. 12 shows how distributed dynamic simulations are performed as in a grid computing environment by run-time coupling of Matlab/Simulink and multiple ESP-r(s) while exchanging data over a network in the form of ASCII, binary or XML format, and in different communication modes such as synchronous, asynchronous, and partially synchronous during the simulation.

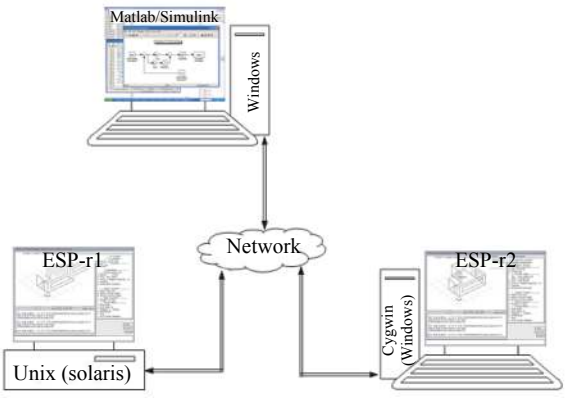


Fig. 12 Distributed simulations by run-time coupling of Matlab/Simulink and two ESP-r(s) as in a grid-computing environment

4.1 Distributed building control application

The research study presented in [2] has highlighted the importance of run-time coupling between ESP-r and Matlab/Simulink over standalone simulations within the integration of advanced control systems in building performance simulation for improving all quality aspects of building indoor environments, as well as in the simulation of building control applications requiring multivariable control systems. In order to demonstrate the development and design of run-time coupling between multiple instances of ESP-r and Matlab/Simulink, and compare the simulation results obtained by different modes of communication, the same building model, which is actually an existing exemplar in ESP-r, was built in two distanced instances of ESP-r, as shown in Fig. 12, and in combination with an external (or remote) proportional integral (PI) control modelled in Matlab/Simulink, i.e., in the form of a closed control loop. This external PI control is here used simply to regulate the air-temperature in a building zone by supplying the required heating flux capacity to it (which is at maximum 3 000 W). Fig. 13 shows a building model built on two instances of ESP-r (left) in a closed loop with its external PI control modelled on Matlab/Simulink (right).

In this application, the continuous PI control is set to maintain the indoor air-temperature (i.e., the control response) at the set-point of 22°C during a period which is valid from 07:00 till 18:00 o′clock. Consequently, the input to the PI control implemented in Matlab/Simulink is the error signal created by subtracting the sensed indoor air-temperature of the building model built on ESP-r from the set-point. The output of this PI control, a weighted sum of the error signal and its gains, is the actuated heating flux to that building model built on ESP-r.

4.2 Simulation results

The simulation results were obtained with run-time coupling between Matlab/Simulink and two instances of ESP-r, as shown in Fig. 11, while exchanging data in a binary format over a network, and in different modes of communication including synchronous, asynchronous, and partially synchronous. Fig. 14 shows the simulation results obtained for the PI control by run-time coupling between Matlab/Simulink and two ESP-r(s) in synchronous mode for simulation time-steps of 1 min (left) and 2 min (right).

By comparing the simulation results obtained by ESP-r1 with those obtained by ESP-r2, it can easily be observed that they are similar and very comparable to each other for both simulation time-steps of 1 min and 2 min. This is mainly because every coupled Matlab/Simulink and ESP-r is synchronized at run-time, and thus takes no network induced time delays into account as both Matlab/Simulink and ESP-r must wait for each other to receive data during the simulation. Fig. 15 shows the simulation results obtained for the PI control by run-time coupling between Matlab/Simulink and two ESP-r(s) in asynchronous mode for simulation time-steps of 1 min (left) and 2 min (right).

By comparing the simulation results obtained by ESP-r1 with those obtained by ESP-r2, it can easily be observed that they are quite different and incomparable to each other for both simulation time-steps of 1 min and 2 min. They are different especially when the control action is applied (or valid which is during the period between 7:00 and 18:00 o′clock), i.e., once the control response has reached the set-point of 22°C because every coupled Mat-

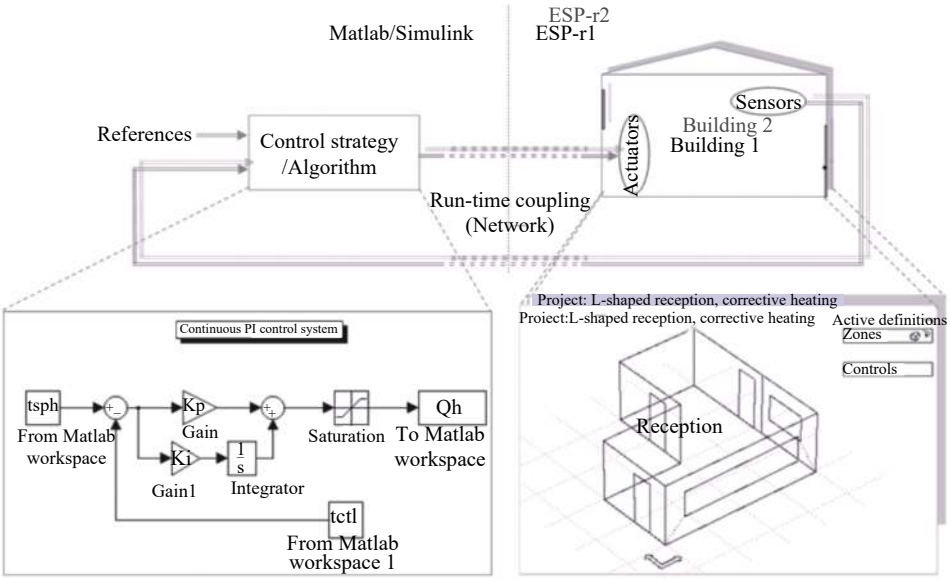


Fig. 13 A PI control modelled on Matlab/Simulink (left) with a building model built on two instance of ESP-r (right)

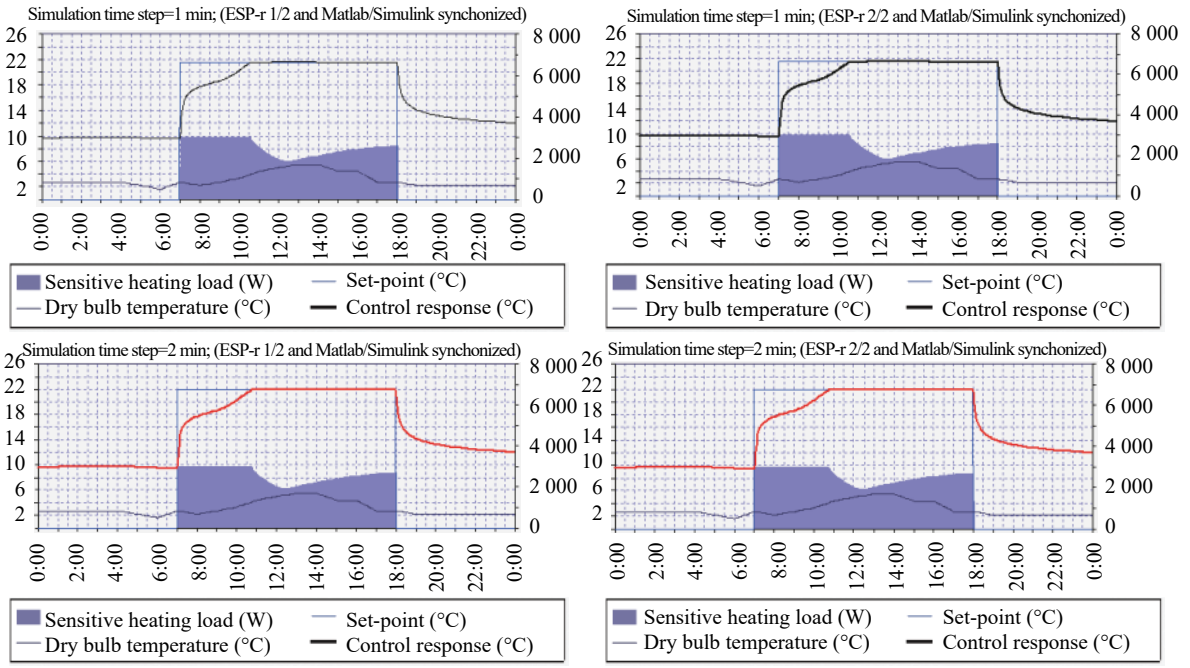


Fig. 14 Simulation results obtained for the PI control by run-time coupling between Matlab/Simulink and two ESP-r(s) in synchronous mode

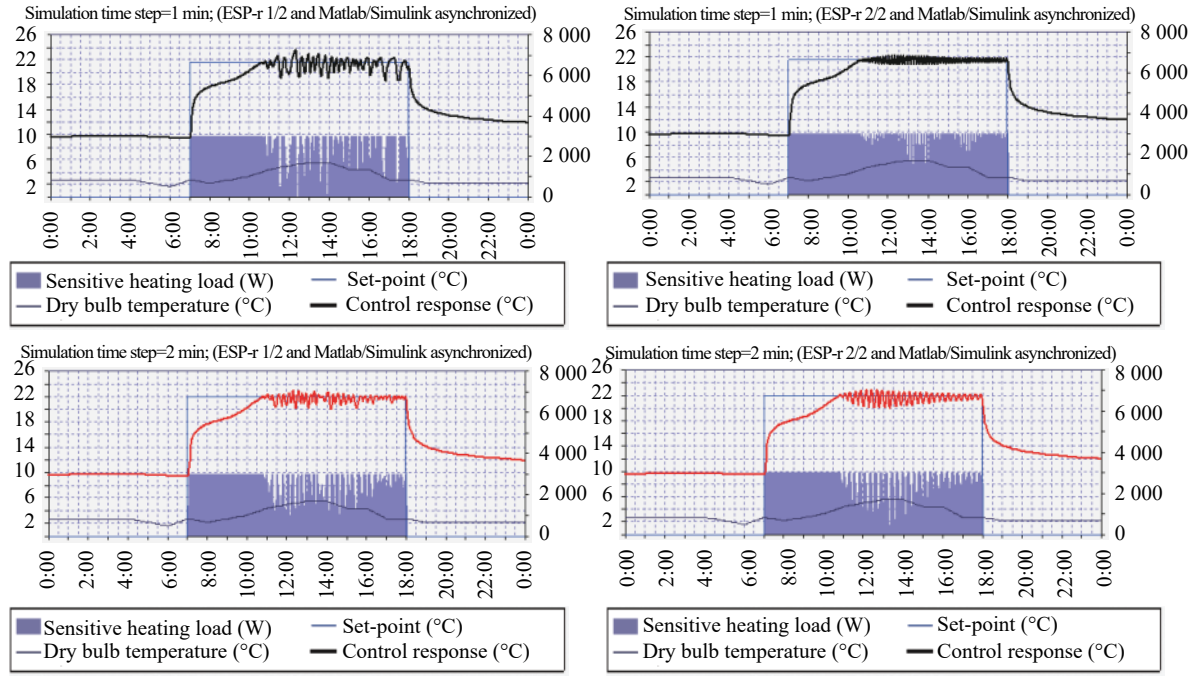


Fig. 15 Simulation results obtained for the PI control by run-time coupling between Matlab/Simulink and two ESP-r(s) in asynchronous mode

lab/Simulink and ESP-r is asynchronized at run-time. Neither Matlab/Simulink nor ESP-r waits for one another to receive data during the simulation as both precede their computations independently. It also appears that these simulation results are different from those shown in Fig. 14 for both ESP-r(s) and different simulation time steps. The reason of this difference is because the addition of a network (i.e., physical distance) into a control loop and the use of more than one ESP-r by run-time coupling with Matlab/Simulink adversely affect the control response due to time delays associated with that network in principal, and time delays associated with the processing capabilities of the used computers for simulation (i.e., over which Matlab/Simulink and ESP-r run). When considering the control performance, it can be seen from Fig. 15 that the control responses are continually in-

fluenced by network induced time delays once they reach the control set-point. Therefore, those results validate the fact that network induced time delays degrade both the performance and the stability of building HVAC equipment and lighting components. Fig. 16 shows the simulation results obtained for the PI control by run-time coupling between Matlab/Simulink and two ESP-r(s) in partially synchronous mode, i.e., when Matlab/Simulink is asynchronized and ESP-r(s) synchronized for simulation time-steps of 1 min (left) and 2 min (right).

It appears that the simulation results shown in Fig. 16 are relatively identical to those shown in Fig. 14, because ESP-r(s) are synchronized and wait to receive data from Matlab/Simulink. Therefore, network induced time delays in this communication mode have no impact on the control responses because building zones and plant models built in ESP-r wait during the simulation (i.e. at every simulation time step) to receive data from Matlab/Simulink. Even though Matlab/Simulink is asynchronized, it does affect the control responses of building models built in ESP-r while being synchronized. However, these simulation results can also be used to validate the design and implementation of run-time coupling between ESP-r and Matlab/Simulink as they are accurately similar to those obtained in synchronous mode. As Matlab/Simulink uses the latest available data from ESP-r, this proves that when ESP-r is synchronized, the simulation results obtained by the two different modes for the two different simulation time steps are perfectly the same. Therefore, this developed distributed dynamic simulation environment by run-time coupling between Matlab/Simulink and one or more ESP-r(s) over a network is accurately implemented and completely operationalized throughout this study. Fig. 17 shows the simulation results obtained for the PI control by run-time coupling between Matlab/Simulink and two ESP-r(s) in partially synchronous mode, i.e., when Matlab/Simulink is synchronized and ESP-r(s) asynchronized for simulation time-steps of 1 min (left) and 2 min (right).

By comparing the simulation results obtained by ESP-r1 and ESP-r2, as shown in Fig. 17, it appears that the results for the simulation time step of 1 min are absolutely different and not comparable to each other, but the results for the simulation time step of 2 min are relatively similar and more or less comparable to each other. From this comparison, it can be deduced that network induced time delays have an impact on the control responses depending mainly on the simulation time step, as it has actually been stated in [1, 18] for networked control systems (NCSs). It also appears that the simulation results shown in Fig. 17 are different from those obtained in Figs. 14 and 16, but partly similar to those obtained in Fig. 15, as especially for ESP-r2 for the simulation time step of 1 min only, because when ESP-r(s) are asynchronized – and Matlab synchronized – the introduction of a network in a control loop impacts the automation and operational integrity of building HVAC equipment and lighting components while adversely affecting their control responses depending on the distance of this network and the time step used for the simulation. In principle,

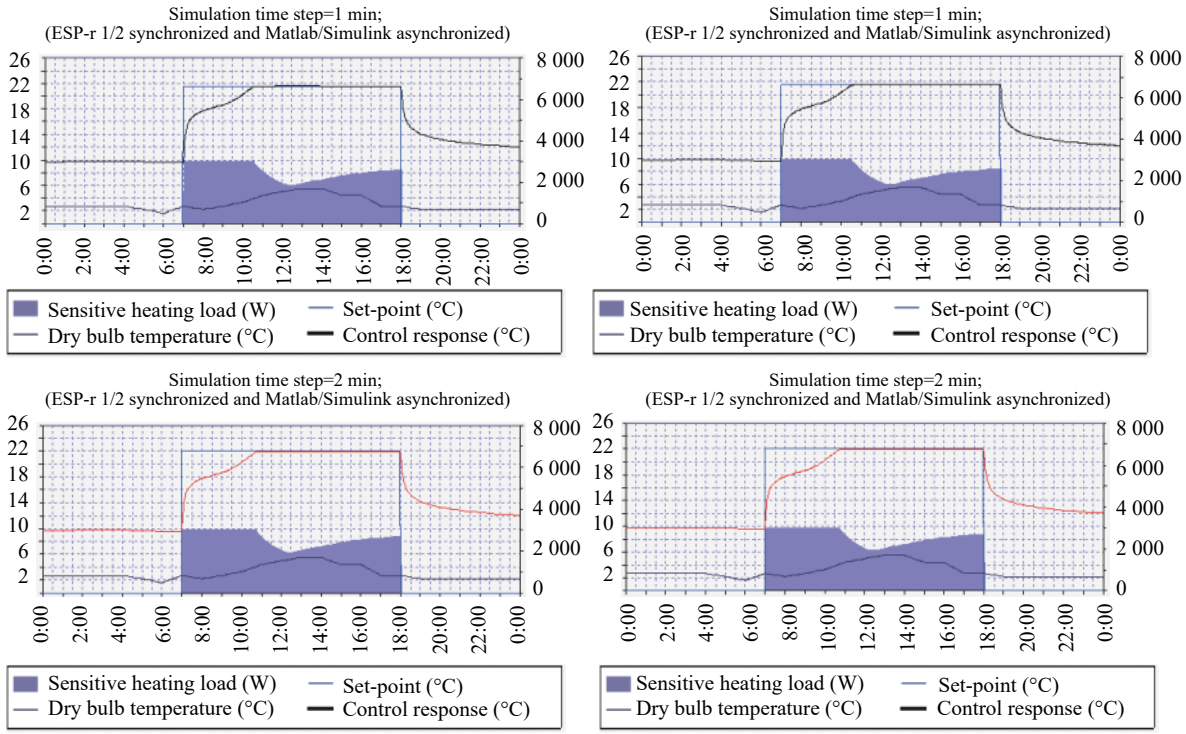


Fig. 16 Simulation results obtained for the PI control by run-time coupling between Matlab/Simulink and two ESP-r(s) in partially synchronous mode (when Matlab/Simulink is asynchronized and ESP-r(s) synchronized)

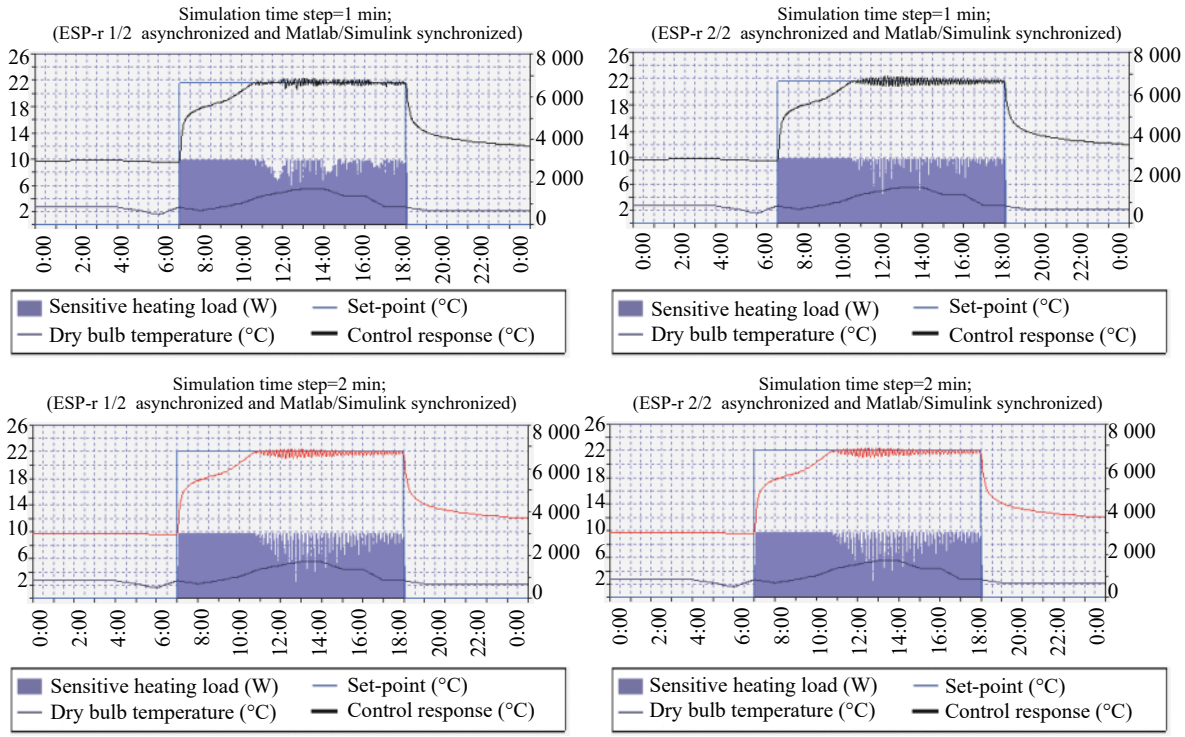


Fig. 17 Simulation results obtained for the PI control by run-time coupling between Matlab/Simulink and two ESP-r(s) in partially synchronous mode (i.e., when Matlab/Simulink is synchronized and ESP-r(s) asynchronized)

this is the communication mode representing real-time control implementations, as they are actually performed in BACS architecture. By assessing the results obtained by ESP-r1 with those of ESP-r2 mainly for the simulation time step of 1 min, it can also be observed that these are different because of less uncertainty that this communication mode deals with compared to the asynchronous mode, and this is also valid even when the network distance that separates the machines over where ESP-r(s) from Matlab/Simulink run is important. In consequence, the use of a network by BACS architecture for distributed control applications in buildings can severely degrade both the performance and the stability of HVAC equipment and lighting components depending on network induced time delays. It is therefore important to note that distributed simulations are essential to investigate and analyze the influence of network induced time delays in distributed building control applications.

5 Conclusions

This paper has demonstrated the application of a distributed dynamic simulation environment that is developed with a capability of representing BACS architecture in simulation by run-time coupling between Matlab/Simulink and multiple ESP-r(s) over a network. It briefly described the design of this dynamic simulation environment and how this is used to similarly represent the BACS architecture in simulation through the use of different communication modes such as synchronous, asynchronous and partially synchronous while performing distributed simulations by run-time coupling between Matlab/Simulink and one or more instances of ESP-r. It finally presented the simulation results by experimental design for the representation of BACS architecture in simulation as a grid-computing environment.

The design of a distributed dynamic simulation environment for BACS exploits the strengths of both Matlab/Simulink and ESP-r to enable the integration of innovative control strategies in building environments, and the representation of real-time network-based building control implementations in simulation. While Matlab/Simulink is used for control systems modelling and design, ESP-r is used for assessing building performance and energy consumption. In principal, this design enables the run-time coupling between Matlab/Simulink and one or more instances of ESP-r over a network in modelling distributed control and building performance applications while exchanging data in different formats such as ASCII, binary, and XML, and in different communication modes including synchronous, asynchronous and partially synchronous.

The development of a distributed dynamic simulation environment for BACS is managed in such a way to support complex and large-scale control applications such as the integration of coordinated control strategies in building performance simulation. Therefore, future work will include simulation of building control applications with coordination of control actions in a distributed network such as of MASs in ABs.

Acknowledgments

The author would like to thank Pieter Smit, Buurman aan Jeroen Boschlaan van Eindhoven in the Netherlands for his true friendship and continuous encouragement, Professor Abdelkader Sahraoui at LAAS-CNRS of Toulouse in France for his significant support in helping me during these years of hard work, and Professor Jan Hensen at Eindhoven University of Technology (TU/e) in the Netherlands for his critical help in doing this research, as well as the anonymous reviewers for their valuable feedback to the manuscript.

References

[1] A. Yahiaoui. A systems engineering approach to distributed control and building performance simulation. In *Proceedings of the 29th International Conference of CIB W78*, Beirut, Lebanon, pp. 422–431, 2012.

[2] A. Yahiaoui, J. Hensen, L. Soethout, D. Van Paassen. Integrating building performance simulation with control modeling using Internet sockets. In *Proceedings of the 9th International IBPSA Conference*, Montreal, Canada, pp. 1377–1384, 2005.

[3] A. Yahiaoui, R. Staal. KR26 A systems engineering approach to embedded control system implementation in buildings. *INCOSE International Symposium*, vol. 18, no. 1, pp. 1717–1730, 2008. DOI: 10.1002/j.2334-5837.2008.tb00912.x.

[4] M. Janak. Coupling building energy and lighting simulation. In *Proceedings of the 5th International IBPSA Conference*, Prague, Czech Republic, pp.307–312, 1997.

[5] Z. Q. Zhai. Developing An Integrated Building Design Tool by Coupling Building Energy Simulation and Computational Fluid Dynamics Programs, Ph.D. dissertation, MIT, USA, 2003.

[6] CSTB. *Type 155-A new TRNSYS type for coupling TRNSYS and Matlab*, Centre Scienfigique et Technique du Bâtiment, [Online], Available: https://www.powershow.com/view/14bc9b-MjlmO/TRNSYSMATLAB_5.powerpoint_ppt_presentation, 2005.

[7] M. Wetter. Co-simulation of building energy and control systems with the building controls virtual test bed. *Journal of Building Performance Simulation*, vol. 4, no. 3, pp. 185–203, 2011. DOI: 10.1080/19401493.2010.518631.

[8] I. Beausoleil-Morrison, F. Macdonald, M. Kummert, T. McDowell, R. Jost. Co-simulation between ESP-r and TRNSYS. *Journal of Building Performance Simulation*, vol. 7, no. 2, pp. 133–151, 2014. DOI: 10.1080/19401493.2013.794864.

[9] M. Wetter, W. D. Zuo, S. T. Nouidui, X. F. Pang. Modelica buildings library. *Journal of Building Performance Simulation*, vol. 7, no. 4, pp. 253–270, 2014. DOI: 10.1080/19401493.2013.765506.

[10] ISO. Building automation and control systems (BACS)-Part 2: Hardware, ISO Std. 16484–2, 2005.

[11] ISO. Building automation and control systems (BACS)-Part 5: Data communication protocol, ISO Std. 16484–5, 2014.

[12] C. Hughes, T. Hughes. *Parallel and Distributed Programming using C++*, Boston, USA: Addison-Wesley, 2004.

[13] A. Yahiaoui, J. Hensen, L. Soethout. Integration of control and building performance simulation software by run-time coupling. In *Proceedings of the 8th International IBPSA Conference and Exhibition*, Eindhoven, Netherlands, pp.1435–1441, 2003.

[14] A. Yahiaoui, J. L. M. Hensen, L. L. Soethout. Developing CORBA-based distributed control and building performance environments by run-time coupling. In *Proceedings of the 10th International Conference on Computing in Civil and Building Engineering*, Weimar, Germany, pp. 86–93, 2004.

[15] ESRU. The ESP-r System for Building Energy Simulation-User Guide Version 10 Series, ESRU Manual U02/1, University of Strathclyde, Scotland, 2002.

[16] B. Einarsson. Mixed language programming, Part 4, mixing ANSI-C with Fortran 77 or Fortran 90. In *Proceedings of International Workshop on Current Directions in Numerical Software and High Performance Computing*, Kyoto, Japan, 1995.

[17] Matlab/Simulink Documentation, (Math Works's website), [Online], Available: https://nl.mathworks.com/, 2015.

[18] A. Yahiaoui, A. E. K. Sahraoui. A framework for distributed control and building performance simulation. In *Proceedings of the 21st International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, Hammamet, Tunisia, pp. 232–237, 2012. DOI: 10.1109/WETICE.2012.44.

[19] W. R. Stevens. *UNIX Network Programming, Vol. 2: Interprocess Communications*, 2nd ed., Upper Saddle River, USA: Prentice-Hall, 1998.

[20] A. Fumagalli, R. Grasso. An efficient asynchronous simulation technique for high speed slotted networks. In *Proceeding of the 32nd Annual Simulation Symposium*, San Diego, USA, pp. 11–18, 1999. DOI: 10.1109/SIMSYM.1999.766448.

[21] J. Shamsi, C. B. Chu, M. Brockmeyer. Towards partially synchronous overlays: Issues and challenges. In *Proceedings of 1st International Workshop on Advanced Architectures and Algorithms for Internet Delivery and Applications*, Orlando, USA, pp.10–17, 2005. DOI: 10.1109/AAA-IDEA.2005.17.

Azzedine Yahiaoui received the B.Sc. degree in electrical and telecommunications engineering from University of Boumerdes, Algeria in 1995, the M.Sc. degree in electronics engineering specialized control systems from University of Blida, Algeria in 1999, master of advanced studies (MAS) degree in automatic systems from LAAS-CNRS of Toulouse, France in 2001, and the Ph.D. degree in distributed computing and control systems from Eindhoven University of Technology (TU/e), the Netherlands in 2013. He is a researcher at TU/e, the Netherlands. He has extensive experience in modelling and simulation, implementation and evaluation of distributed control systems, application of systems engineering concepts, as well as analysis and design of control systems for different domains of engineering namely aerospace, rail, automotive and buildings.

His research interests include systems of systems engineering, autonomous vehicles, distributed and hybrid control systems, multi-agent control systems, verification and validation of complex systems, and buildings automation and control systems.

E-mail: azzedine.y@outlook.com (Corresponding author)

ORCID iD: 0000-0003-2581-4521