INTERFACING BUILDING PERFORMANCE SIMULATION WITH CONTROL MODELING USING INTERNET SOCKETS

Azzedine Yahiaoui[*1], Jan Hensen[1], Luc Soethout[2], Dolf van Paassen[3]
[1]Center for Building & Systems TNO-TU/e
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
[2]TNO Built Environment and Geosciences
Postbus 49, 2600 AA Delft, The Netherlands
[3]TU Delft, Department of Mechanical Engineering
Mekelweg 2, 2628 CD Delft, The Netherlands

ABSTRACT

This paper reports on progress of an ongoing research project, which aims to achieve better control modeling in building performance simulation by integrating distributed computer programs. Recent developments show that there is a need to enhance building performance assessments by integrating new simulation features in order to predict the overall effect of innovative control strategies for integrated building systems. However, both domain independent control modeling environments and domain specific building performance simulation, have their own restrictions. For example, certain control features are represented in one simulation environment while others are only available in other simulation software. To alleviate these practical problems, this paper describes a mechanism that can be used to allow a building simulation environment to exchange data with an external control simulation environment.

In particular, this paper focuses on the problem of developing run-time coupling of control and building performance environments over TCP/IP using Internet sockets. The socket implementation is analyzed in terms of minimizing overhead, communication efficiency, and the integration into existing software tools. Perspectives for a run-time coupling specification are given to enable connection-oriented sockets to easily exchange data as well as coupling software. Data requirements in view of integration in real building control protocols (BACnet and LonWorks) are discussed. An early implementation of run-time coupling is demonstrated with a case-study, and the paper finishes with some conclusions and directions for future work.

INTRODUCTION

With the rapid advances in microelectronics, powerful computer-based building monitoring and control systems have emerged. These Building Automation Systems (BAS) or Building (Energy) Management Systems (BEMS) control systems such as for heating, ventilation and air-conditioning (HVAC) and lighting. They also perform tasks such as access control, energy management, and fault detection and diagnoses. The principal setup of a BEMS as depicted in Figure 1 is to provide a healthy, comfortable and productive indoor environment while reducing fossil fuel consumption and green house gas emissions, and to achieve this in an efficient and rational way (see e.g. IEA 2002).

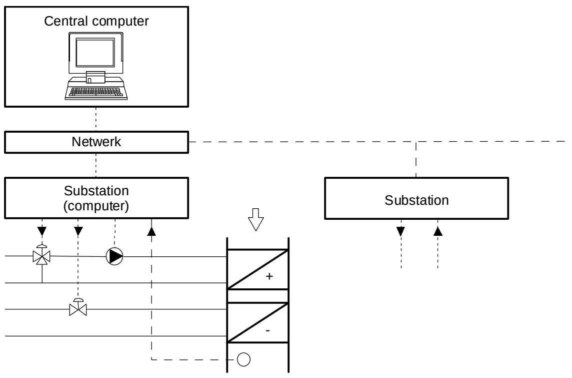

Figure1 General architecture of a building energy management system

As mentioned by e.g. Van Paassen (1986), a computer based control system is much more flexible than traditional control systems. The software allows adapting the control strategy to changing conditions. The central computer defines which control loops are suitable to be activated and what would be the appropriate set-points for the current situation. This computer is connected to substations via a network. Substations constitute the automation level that collects message (data) and regulates actuators. For communication, a protocol is used to exchange those data between the central computer and the system components. Usually, a message includes the address of the component and the series that refer to the real physical quantity (e.g. the desired position of valve).

It is very important that the advanced control strategies that are possible with BEMS are taken into account already during the design phase of the building and its installation. This requires that such

[*] Corresponding e-mail: *a.yahiaoui@bwk.tue.nl*

strategies are properly modeled and incorporated in building performance simulation. However, existing domain specific software for building performance simulation (BPS) is usually relatively basic in terms of control modeling and simulation capabilities (e.g. ESP-r, TRNSYS). On the other hand, we now have domain independent control modeling environments (CME), which are very advanced in control modeling and simulation features, but rather limited in building performance simulation concepts (e.g. Matlab/ Simulink). Marrying the two approaches by run-time coupling would potentially enable integrated performance assessment by predicting the overall effect of innovative control strategies for integrated building systems.

Although, our current work starts from specific BPS and CME simulation environments, the communication mechanism and data-exchange protocol that is being developed and implemented, have much wider and more general applicability. ESP-r and Matlab / Simulink were selected as BPS and CME software tools respectively. ESP-r and Matlab/Simuling can run on different operating system platforms. ESP-r is used for building modeling only and Matlab/Simulink is considered as a controller that can be configured remotely. It is felt that this type of technology development will enable design and evaluation of advanced building control applications that are currently infeasible.

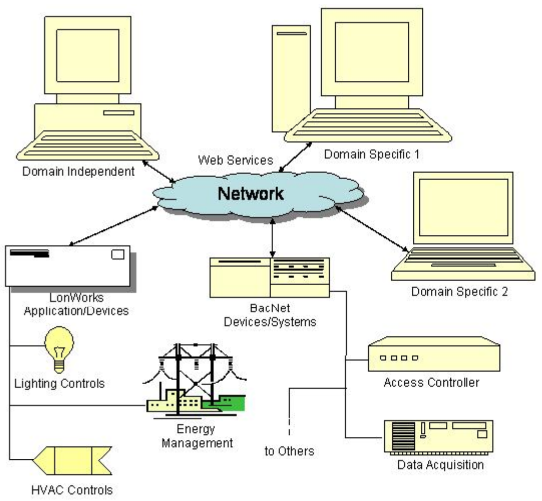

Figure 2 Networked building simulation and automatic control systems

For the coupling between the different simulation environments, we want to mimic as much as possible the situation of a real BEMS. As schematically shown in Figure 2, overall system integration is in our approach achieved by connecting the various hardware and software subsystems to a network, such as the Internet or a local area network (LAN). In order that the subsystems can communicate information, it is necessary to use a common data exchange format such as the eXtensible Markup Language (XML; see e.g. Chervet 2004). Nowadays XML might even be used as a gateway between different communication protocols such as LonMark, BACnet and Modbus, all of which are used in BEMS. Communication protocols that are used by BEMS have come up with a structure to describe the information that is sent across the network. The same kind of information needs to be exchanged in our distributed simulation environments where building modeling and control system are separated and work together through run-time coupling. This motivates our choice to use internet techniques based on socket communication for the run-time coupling between simulation environments

The remainder of this paper is organized as follows: the next section describes the background of run-time coupling. Then it follows the reasoning behind using Internet sockets in the implementation. This is followed by case study demonstration, and finally conclusions and directions for future work.

BACKGROUND

Coupling different applications (simulation executables) at run-time provides the facility to exchange the information in a co-operative way. Usually, one application controls the overall simulation procedure and requests the other application (s) when necessary. For example, Janak (1997) has enabled run-time coupling between ESP-r and the ray-tracing lighting simulation and visualization software Radiance.

In principle, the coupling of tools requires an underlying communication mechanism to allow the transfer of both data and control between codes. The data transfer corresponds to the sending of the results calculated by one of the processes and to the receiving by another process whereas the transfer of control is the execution of one particular function to be performed remotely in another process. Most of the attempts to run-time couple codes rely on the use of shared (or intermediate) files, which is not a very efficient way to exchange data at an adequate level of abstraction (Hughes and Hughes 2003, Ranganathan et al. 1996). In (Yahiaoui et al. 2003), we described and compared also various other options for inter-application data transfer facilities.

Inter-process communication (IPC) can be viewed at different levels of abstraction. At a higher level, it appears as an exchange of information between processes. These processes are either contained in a single parallel application or in distributed applications, or may otherwise belong to different applications that need to communicate. At a lower level, communication appears as the mere traffic of bits from one address space to another. Distributed simulation requires support for expressing communication at high-level of abstraction. A powerful abstraction in distributed applications is the IPC mechanism that deals with distributed object systems, such as CORBA (Common Object Request Broker Architecture), in which computer applications, written in any oriented-object programming language, can work together over the network. However, as reported earlier (Yahiaoui et al. 2004), adapting CORBA to explicitly run-time couple ESP-r with Matlab (neither of which are written in an object-oriented language) raises major difficulties in dealing with the CORBA interface definition language (IDL).

In contrast, socket programming is complicated, but we can build more sophisticated communication abstractions on top of sockets (e.g. Stevens 1999). Socket communication has been used earlier for evaluation of HVAC control performance (e.g. ASHREA 1997). In addition, a socket domain is an abstraction that provides addressing structure and a set of protocols such as: Internet Protocol, BacNet, LonTalk, and so on. For such reasons, the use of internet sockets seems a good alternative for developing and implementing a coherent paradigm for communication between BPS and CME software.

DEVELOPING RUN-TIME COUPLING FOR SIMULATIONS

The initial objective of the current research is to marry the ESP-r and Matlab environments in order to be able to perform distributed building control simulations. This involved identifying the specific components that should be connected in order to communicate data between ESP-r and Matlab, which, coincidentally, may run on different machines using different operating systems. That is to say that in the run-time coupling approach, applications can be distributed over heterogeneous environments. In this regard, socket based IPC has been developed with a set of techniques that permit communication over the network as shown in Figure 3.

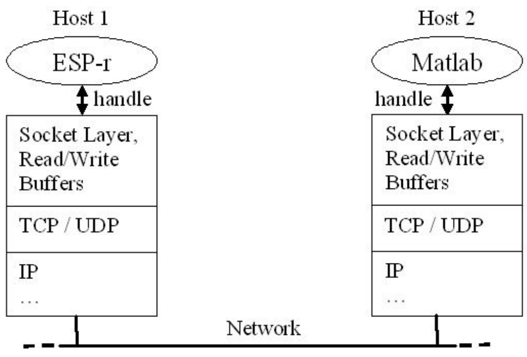

Figure 3 Networked communication between ESP-r and Matlab using sockets

Sockets are abstract communication endpoints. They are basic components of inter-process and inter-system communicating over a network. A socket is made up of an IP-address concatenated with a port number. In general, sockets use a client-server architecture. The server waits for incoming client request by listening to a specified port. Once a request is received, the server accepts a connection from the client socket to complete the connection. In our case, ESP-r is the client/master and Matlab/simulink is the server/slave. Hence, Matlab/simulink can be launched at every ESP-r time-step as a separate process.

IPC USING INTERNET SOCKETS

While run-time coupling involves many aspects, the current paper focuses on programming network communication. Internet domain communication which uses the TCP/IP (Transmission Control Protocol/Internet Protocol) suite is the most prevalent set of communications protocols in use today. TCP/IP provides communication services that are interoperable with virtually any network configuration, allowing the internet sockets to act as the basic element of the network.

UDP vs. TCP protocol

Socket types define the communication proprieties visible to the application. Indeed, the term TCP/IP refers to more than just the TCP protocol itself. It specifies an entire that bridges communications between the transport layer and the upper (session/ presentation/ application) layers through a network medium. Within the TCP/IP suite, two protocols are mainly used: TCP (Transmission Control Protocol) and UDP (User Datagram Protocol). Both of them have their own advantages and disadvantages. Each of them can be better for a particular problem related to run-time integration of BPS and CME software. What follows is a short description:

1. TCP is the most prevalent of internet domain communication. It supports connection-oriented communication between client (e.g. ESP-r) and server (e.g. Matlab/ Simulink). Before any data is transmitted, TCP establishes a mutually acknowledged connection between the sending components and the receiving components (or nodes). During communication, this connection is maintained through the transmission of data by one component followed by an acknowledgement (ACK) of receiving data at the other node. If the sender of data does not receive this ACK, the data in question will be retransmitted. Finally when all data has been transmitted, the connection is closed. Although TCP offers the advantage of reliability of transmissions (sent data is guaranteed to be delivered), this feature costs time in connection set-up and error checking, especially during large transmitting volumes. In essence, TCP may experience low

performance due to significant overhead per transmitted data segment.

2. UDP provides communication in connectionless mode (no notion of connection). UDP communication consists only of transmitting one data packet with attached routing information. UDP is not responsible for notification prior to the transmission of data nor for confirmation of data delivery. As a result, UDP offers no delivery guarantees. This simplicity, however, offers significant benefits in saving time for connection set-up and error checking. Furthermore, since no connection must be established, the components of sending and receiving data can be made to run promptly, regardless of whether they are successfully completed. UDP is usually used to efficiently exchange small datagrams (*<64MB*). UDP allows applications to exchange data across the network with a minimum of overhead.

In the development of run-time coupling between ESP-r and Matlab over the network, more particularly between controller and building model, the time that it takes to send and receive small data volumes might be the limiting factor on allowable sampling interval. This can happen when a type of controller, which uses or depends on dynamic parameters, requires to instantaneously changing the position of a specific building component. For speed of network communications, the UDP protocol would be a better choice than TCP. We envisage using a building domain specific control protocol, i.e. BACnet/IP (a data communication protocol for Building Automation and Control Networks). The BACnet protocol itself provides the guaranteed delivery of data. This does not require the use of TCP. Moreover, the added reliability of TCP may seem to give an advantage over UDP, but this may not necessarily be the case. Improved reliability in TCP is achieved by the re-transmitting data feature that guarantees delivery but also guarantees that the re-transmitted data will arrive later than intended. The delivery of this late data is not worth the additional network bandwidth it requires, as closed–loop would be better served by a new transmission of recent data than by re-transmission old data. Indeed, the timing of a run-time coupling mechanism requires the ability to execute send and receive functions with great precision. Summing up, UDP seems to be the better choice for our type of applications. Therefore an appropriate signal for necessary error checking and possible repeated function calls is implemented (i.e., *select()* function is used).

Exchange data with UDP

At the top layer, application level communications within the control loop are performed using functions in sockets application programming interface (Sock APIs). The underlying concept upon which those functions are constructed is that of a socket, which simply refers to the communication between ESP-r and Matlab through the network.

The data exchanged with UDP represent the physical quantities as they are measured in a real application. However the data that should be exchanged from ESP-r to Matlab and required passing back from Matlab to ESP-r are first converted by flattening of structured data items to external (network) representation on one side, and then reconverted by rebuilding data structures to internal representation on other side. This exchange format has for the purpose to transfer data in safe mode. The main advantage of this approach is to enable run-time coupling of the simulation environment with a real building (e.g. for control purposes) or with building components (e.g. HVAC systems).

DATA COMMUNICATION IN A REAL BUILDING

In real operations, it is desirable for control loops to communicate with building components or HVAC systems. The communication between controllers and building systems requires an interface or a gateway, due to their different communication speeds and data formatting. The proper operation of the gateway is dependent on the continued use at the corporate level to implement between controllers and building systems. In consequence, two standard protocols “BACnet” and “LonWorks” (see e.g. SmartHome 2000) may be distinguished:

Both protocols cover most or all layers of the OSI (Open System Interconnection) model for network communication. For applying some of the ideas developed for these protocols in our case of run-time coupling, only application layer is considered. The other layers deal with hardware specifications and the low-level details of the network communication. In our case, these details are already covered by the socket implementation. The application layer however deals with the conceptual part, i.e. with the way of information is modeled.

BACnet is a specification for a standard protocol published by ASHRAE organization. The run-time coupling mechanism is implemented in order to create a communication protocol that complies with this specification. The standard defines protocol implementation conformance statements (PICS) that identify different levels of compliance. This meets a field level of Building Energy Management System (BEMS) that run-time coupling between ESP-r and Matlab uses to communicate over BACnet protocol.

To support communications on a variety of LANs or to allow different components to co-exist in the same network, LON (or LonWorks) that communicates via

Standard Network Variable Types (SNVT) facilitates interoperability by providing a well-defined interface. Basically a proprietary communication protocol is called LonTalk, created by ECHELON Corporation. A chip is required for any device that uses LON. Standard network variable formats can be established by run-time coupling to allow the transfer of data between Matlab and ESP-r using the LonMark subset of LON capabilities to interoperate with each other.

IPC TO ESP-R AND TO MATLAB/SIMULINK BINDING

To use sockets IPC, C and C++ programming languages are used of which socket libraries are exploited. Neither building performance simulation software (ESP-r) nor control modeling environments (Matlab/Simulink) have simple interface with sockets communication mechanism. Therefore we have describes an approach, which is developed in usual way of interfacing sockets communication mechanism to ESP-r and Matlab/Simulink as well.

Interfacing to ESP-r

Currently ESP-r is the legacy of Fortran codes. Mixed language programming is used to interface between Fortran and C/C++ programs (e.g. Einarsson 1995). This is often a very appropriate way of combining the strengths of various programming languages, in which the language subset is common between C and C++, and between Fortran 77 and 90. The developed approach consists of using a method that can be functional for both Unix and Linux Operating Systems in which ESP-r can be installed. This method is based on external structure function (*extern struct)* in order to add new variables that need to be exchanged without modifying programming codes. With this function, a ESP-r subroutine can exchange data with the C client code of sockets communication mechanism through the function was made. Afterward, the C client codes, which developed and support sockets communication are compiled within the concerned ESP-r packages. This creates one executable, a client process which is formed by combining ESP-r packages and sockets communication codes together.

Interfacing to Matlab/Simulink

Matlab has a built-in utility called MEX (Matlab Executable), which is used to convert C/C++ to Matlab-Executable format. The original sense of the Matlab/Simulink word represents two different environments, which are a high-level programming environment and a graphical user interface respectively. The link to external programs depends on which environment we would interface. For Matlab, MEX-files are used and are dynamically linked programs that can be called from within Matlab as regular Matlab. In case we need to deal with Simulink, the link should also be performed between each other. Practically the same approach is used for Simulink, although MEX S-functions are used and are dynamically linked programs within Simulink. Thus, when an application requires computational control dynamics, it should call Matlab code. For the use of Matlab / Simulink interoperability; technically permutations flexible usage model between Matlab, Simulink and C/C++ are described in table 1. More detail about can be found in (MathWorks, 2005).

Table 1 Interoperability between MATLAB, Simulink, and C/C++

FROM…	**CALL…**	HOW?
C/C++	Matlab	MATLAB Engine ActiveX/COM & DDE MATLAB Compiler & runtime
Matlab	Simulink	Using *sim* function: >> sim('mymodel')
Simulink	C/C++	S-function Builder C-code S-function
C/C++	Simulink	Engine or COM [x,y,t] = sim('mymodel',u0)
Simulink	Matlab	M-code S-function Caution: interpreted, but not compiled
Matlab	C/C++	MEX ActiveX/COM & DDE

Moreover, those external interface functions (MEX-files and MEX S-functions) as described previously can send and receive data to/from C program through a gateway (or a bridge) that was made. This can be done once we compile them with MEX utility. After the compilation, it generates a DLL (Dynamic Link Library) file for any operating system. With this procedure, Matlab control files can be invoked when the server receives data from a client process. A server which is generated by compiling MEX-file and sockets communication codes in a group is like a DLL file which is used and recognized by Matlab.

RUN-TIME COUPLING OF BUILDING CONTROL STRATEGY

In the current implemented approach of run-time coupling between ESP-r and Matlab, it is ESP-r which starts simulation. As Matlab is a server for ESP-r (a client). If the connection is not established between client and server the data to be exchanged are not transferred.

The run-time coupling of ESP-r and Matlab strategy developed in this work takes the form of a closed-loop control system, as shown in figure 4. This can performed either for the feedback control structure or for the feedforward control structure. Both control loops can be used for a simulation of building control systems. In case we would use open-loop control for building application, it would just necessitate to setting, in Matlab side the variables requires passing back to ESP-r to zero, and more exactly as defined in Matlab language are: the *rhs* (right hand side) arguments. Finally, both unidirectional or bidirectional communication are carried out within this run-time coupling strategy.

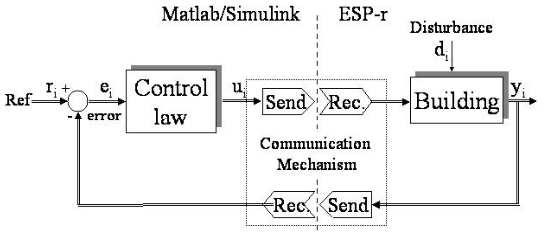

Figure 4 Structure of run-time coupling between control and building performance simulation

A user-interface in ESP-r is modified by adding one row of "External Control" at the end of the control period window. This allows us to setup an external controller for a specific period of day. Once the simulation has been started, ESP-r alters data transfer over a network to the involved controller in Matlab by run-time coupling when necessary. Then if data transfer requires passing back to ESP-r, the controller invokes ESP-r when its processing in Matlab is terminated. In this case, the implemented run-time coupling of ESP-r and Matlab is supported with basis data transfer protocols. The manner of data exchange with Internet sockets depends on whether the communication is synchronous or asynchronous. If some building control applications requires running in real-time, it is necessary that run-time coupling of ESP-r with Matlab is in asynchronous mode. Therefore the different communication events are briefly described below.

Asynchronous Mode

This event occurs when the processing of ESP-r and Matlab are independent from each other. Figure 5 shows the event when the time step of one of them (either ESP-r or Matlab) is different from another.

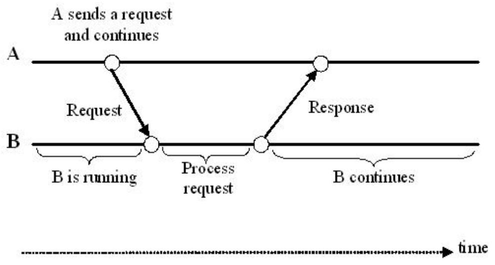

Figure 5 Asynchronous mode

However, asynchronous (parallel) simulation of run-time coupling between ESP-r and Matlab is characterized in that they contain 'no-blocking send' and 'no-blocking receive' functions. Indeed within socket communication, the use of some signals, like for example *SIGIO*, allows the transmission of data in asynchronous manner (e.g. Leffrer et al. 1993). Although the asynchronous simulation of coupled programs is able to deal with any updated information, the execution time can be significantly reduced (e.g Fumagalli et al. 1999). Therefore, the controller is updated each time the events arrive from building components, and verse versa. This step is important in control systems when building components are in the presence of delays or when objects located in building change their actions within the time. Consequently, asynchronous simulation is efficient for an assessment of some control analysis for integrated building systems. But it is harder to parallelize, as the receiver does not know when the transmitter is going to send data.

Synchronous Mode

This event occurs when the two processes (ESP-r and Matlab) are synchronizing with the same defined time in execution. Each process (either ESP-r or Matlab) waits for incoming data from another process. In this case, the time step is defined by the client process (ESP-r). On the other hand, the server process (Matlab/Simulink) is in charge of the same time step defined for ESP-r. This is a usual and simple way of run-time coupling when ESP-r and Matlab exchange data at end of each time step, as shown in figure 6.

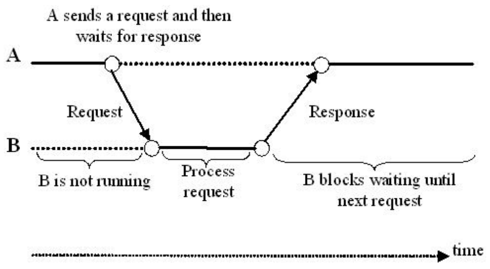
A sends a request and then waits for response
A
Request
Response
B
B is not running
Process request
B blocks waiting until next request
time

Figure 6 Synchronous mode

Since a building control model is executed sequentially, the transmission by synchronous mode blocks the whole simulation until the simulation/execution environment receives data. In real-time simulation, some of scheduled transitions may be delayed because of this blockings. As a result, if the implemented run-time of ESP-r and Matlab uses synchronous mode, it should be ensured that the time constraints of scheduled transitions are satisfied (adjusting a timing of a control loop).

BUILDING CONTROL APPLICATION

To demonstrate the application of the developed and implemented run-time coupling between ESP-r and Matlab, an application case is presented here. The application comprises a working office space unit (4.8*4.2*2 m^3) with two radiant-ceilings used for both heating and cooling mode. The on/off controller is used to regulate the appropriate temperature inside the room by opening or closing the valves on pipelines (either the pipeline of warm water or the one of cold water). The constructions used in this office space are internally insulated cavity walls and internally single glazed walls. The office is located in a six floor of the building sited around the atrium, as shown in figure 8.

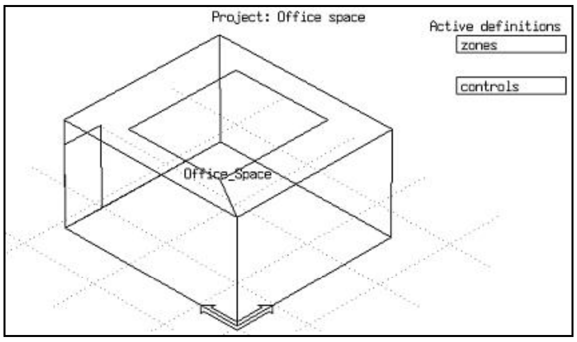

Figure 8 The Building model

The walls facing south and the atrium are in a single glazed structure. It has a thermostat in which the user is allowed to set a temperature at five degrees higher or lower than the common set-point, which is 21degree-Celsius. This illustrated case-study investigates an application that has two objectives. The first consists of comparing the results of (internal and external) controllers to the set-point within the same time step of 10mn/hour. The second is to qualify the importance of the run-time coupling approach. Figure 9 compares the simulated results by regulating a temperature in the office. The simulated temperature results are obtained using internal control, which refers to the On/Off controller implemented and available in ESP-r, and external control which refers to the On/Off controller implemented in Matlab. The obtained results for external control are within the use of implemented run-time coupling between ESP-r and Matlab using Internet sockets. This external controller is presented when ESP-r and Matlab are executed in the same host and when they are run in different hosts. In the present application, the building model is carried- out in ESP-r and the controller is set-up in Matlab. By combining them by run-time coupling, the simulation is performed in synchronous mode. Therefore, Matlab is launched at every ESP-r time step as a separate process. However, the obtained results for internal control and external control are basically similar during the working period (from 7:00 to 18:00 o'clock). An accurate comparison shows that the simulation results are identical. The results also show the oscillation of different responses of controllers around the set-point in working period, this is due to controller capability in which control action takes place every time a deviation occurs from set point. This action responds quickly but is sensitive to input noise, which causes chattering at short intervals.

The importance of run-time coupling between ESP-r and Matlab is to incorporate advanced control analysis methods in building performance simulation for improving all quality aspects of an indoor environment, and for enabling an integrated approach to independent variables such as temperature, light, solar penetration, … etc. Using more than one host at a problem might enhance the performance and the reliability of run-time coupling when the time plays a critical part for the simulation of advanced building control systems. The principal result of such an approach will certainly enhance building efficiency while maintaining optimum energy consumption.

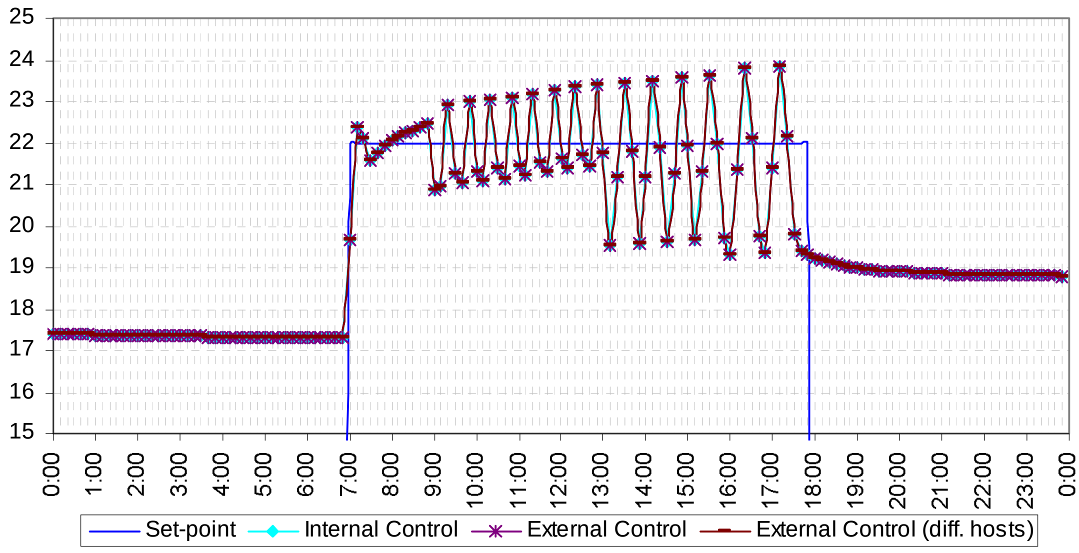

Figure 8 Simulation results

It should be noted that it might be dangerous to use the asynchronous communication for the current application, as the output signal from the On/Off controller is either fully on or completely OFF depending on the error deviated from a set point.

CONCLUSION AND PERSPECTIVES

The paper has described a promising approach to run-time coupling between ESP-r and Matlab/ simulink as implemented using internet sockets. This approach is based on the UDP protocol to exchange data between building model and its controller. The development of this new advent would potentially enable integrated performance assessment of new building control strategies that are not yet possible.

This type of approach must be validated through application in a realistic system, especially since the developed run-time coupling approach incorporates advanced control strategies for integrated building systems with time-varying delays. Issues concerning the network's reliability and utilization have been discussed. The purpose of this research is to evolve the prospects for integration in real building control protocols.

Concerning the perspectives of this research, there is a necessity to formalize the run-time coupling mechanism that uses a connection-oriented socket. TCP is a reliable mechanism that provides a better reliability in communication and guarantees transfer of data. An approach is envisaged to run-time couple ESP-r and Matlab with TCP. This approach will be based on an architecture that consists of reducing the overhead by minimizing the time for communication, including latency and bandwidth. Furthermore, this overhead can be reduced by minimizing function calls and optimizing the size of communication buffers to data types need to be exchange.

Future research will involve applying this approach to realistic application case study settings to prove the success of implementing the current run-time coupling. However, and more importantly, this research will generate associated knowledge for general and wider applicability.

ACKNOWLEDGMENT

Drs. A. Keizer of Centrum voor Taal en Techniek is gratefully acknowledged for her helpful assistance in editing this paper.

REFERENCES

ASHRAE 1997. A Standard Simulation Testbed for the Evaluation of Control Algorithms and Strategies, ASHRAE 825-RP, GA, USA.

Chervet, A., 2004. XML and Building Automation, Journal ASHREA, N. 12, pp 24-32

Einarsson, B., 1995. Mixed Language Programming, Part 4, Mixing ANSI-C with Fortran 77 or Fortran 90, Proc. International Workshop on Current Directions in Numerical Software and High Performance Computing, Kyoto, Japan

Fumagalli, A. and Grasso, R., 1999. An Efficient Asynchronous Simulation Technique for High Speed Slotted Networks, Proc. 32nd Annual Simulation Symposium, San Diego, USA.

IEA, 2002. Control Strategies for Hybrid Ventilation in New and Retrofitted Office Buildings (HYBVENT), Annex-35 Report, University of Aalborg, Denmark.

Hughes, C. and Hughes, T., 2003. Parallel and Distributed Programming using C++, Addison-Wesley, Boston, USA.

Janak, M. 1997. Coupling Building Energy and Lighting Simulation, Proc. Building Simulation, Vol. 2, pp. 313-319, Prague, Czech Republic

Leffler J., Fabry R.S., Joy W.N., Lapsley P., Miller, S. and Torek. C. 1993. An advanced 4.4.BSD interprocess communication tutorial, Technical report, Computer Science Research Group, University of California, Berkeley, USA.

MathWorks, 2005. MathWorks's website. < http://www.mathworks.com/>

SmartHome, 2000. SmartHomeFurom's website < http://www.smarthomeforum.com/>

Schmitt, M., Acharya, A., Ibel, M. and Iancu, C. 2001. Service Sockets: A Uniform User-Level Interface for Networking Applications. Technical Report, Computer Science Department, University of California at Santa Barbara, USA

Stevens, W.R., 1999. UNIX Network Programming. Vol. 2: Interprocess Communications, 2nd Ed., Prentice-Hall, Upper Saddle River, NJ, USA

Ranganathan, M., Acharya, A., Edjlali, G., Sussman, A. and Saltz, J., 1996. Flexible and efficient coupling of data parallel programs, Proc. Parallel Object-Oriented Methods and Applications, Santa Fe, Mexico.

Van Paassen, A.H.C., 1986. Control of Indoor Climate Technology, Journal A, Vol-22, N. 03

Yahiaoui. A., Hensen J.L.M. and Soethout L.L. 2003. Integration of control and building performance simulation software by run-time coupling, Proc. IBPSA Conference and Exhibition, Vol. 3, pp. 1435-1441, Eindhoven, Netherlands.

Yahiaoui, A., Hensen, J., and Soethout, L., 2004. Developing CORBA-based distributed control and building performance environments by run-time coupling, Proc. 10th ICCCBE, Weimar, Germany.