A Data Management Framework Providing Online-Connectivity in Symbiotic Simulation

Sebastian Bohlmann, Matthias Becker, Helena Szczerbicka
Department of Simulation and Modelling
Leibniz University Hannover
Welfengarten 1
30167 Hannover, Germany
{bohlmann, xmb, hsz}@sim.uni-hannover.de

Volkhard Klinger
Department of Embedded Systems
FHDW Hannover
Freundallee 15
30173 Hannover, Germany
volkhard.klinger@fhdw.de

KEYWORDS

Hybrid Process Simulation, Process Control, Communication Protocols, Symbiotic Simulation

ABSTRACT

Symbiotic simulation in industrial applications requires efficient connectivity between industrial processes and embedded legacy simulators. One main challenge is to handle heterogeneous system parameters like bandwidth, latency, redundancy, security and data representation. Moreover, data management needs to be improved in terms of unified access providing an interface to online, historical and corresponding simulation data.

This paper proposes a framework for symbiotic simulation addressing the problem of connectivity. We introduce the Process Data Streaming Protocol (PDSP) managing distributed process data flows. Additionally we present PDSP based modules covering different modes of operation for data processing. The Framework interacts with distributed control systems via object linking and embedding for process control as well as to embedded systems using different hierarchic operation modes. Historical data can be provided transparently through an integrated stream-database. The framework is primarily optimized to be used in JAVA-based simulation environments, but is not limited to these. Finally we demonstrate the usability of the system while interacting with a simulation environment for a hybrid process and present some experimental results.

INTRODUCTION

Nowadays it is possible by the help of today's microprocessor technology as well as the simulation software to simulate complex technical and industrial systems in a speed corresponding to the real process. Furthermore simulators use data provided by the underlying process (for example for reasons of the performance) to get better simulation results in a narrowly limited time period. As a consequence it is necessary to bind process and simulation with each other closely. This synchronization with the original process is seen as problematic because of the heterogeneous descriptions and interfaces. Since a consistent connection is, however, a key feature to maintain a symbiotic behaviour between a simulation and the real process, we will introduce a protocol to handle this. Furthermore experiences with a reference implementation of this component are presented. These are suitably for simulators targeting hybrid (continuous and discrete) processes. They can for example be used in plant simulation in the pulp and paper industry. For a prototypical integration to achieve experimental results the Hybrid Process Net Simulator (HPNS) framework (Bohlmann et al., 2009) will be used here. This currently under development hybrid simulator is based on models similar to petri nets as description form. The underlying communication protocol and the areas of application of the current implementation are in the foreground, though.

Technical systems are specified by the combination of different components which allow only a part-specific and heterogeneous description on account of the mechanical, electronic and information-processing subsystems. Hence, the modeling of the entire system in a closed simulation framework is not possible as a rule. The hardware/software-co-simulation represents a good example for this situation: Based on an interface and a synchronization process the simulators of the hardware and for the software subsystems are coupled together to simulate and to evaluate without a standardized model view. In particular in the automation technology there are almost no closed model-based simulation methods for both, the process and the necessary process control. In this paper we want to show a strategy to online simulate complex systems consisting of physical/chemical processes, mechanical systems and Hardware/Software-subsystems with the help of PDSP based components. Physical, chemical and mechanical subsystems are characterized primary by a continuous behaviour. Hardware subsystems can be characterized by continuous and discrete behaviour, while software is characterized by discrete behaviour.

PDSP ARCHITECTURE

In this section we describe the architecture of the Process Data Streaming Protocol, characterised by different modes of operation and requirements in simulation technology in general. PDSP is designed to be used in mixed continuous and discrete environments (Zeigler et al., 2000) referred to as hybrid. Focusing on symbiotic simulation (Fujimoto et al., 2002) PDPS is primary

Proceedings 24th European Conference on Modelling and Simulation ©ECMS Andrzej Bargiela, Sayed Azam Ali David Crowley, Eugène J.H. Kerckhoffs (Editors)
ISBN: 978-0-9564944-0-5 / ISBN: 978-0-9564944-1-2 (CD)

designed to satisfy four modes of operation.

- Analytic mode
  The analytic mode is used to analyse a process based on captured data. Therefore historical data is used to create and tune models. In this mode no real-time or redundancy requirements exist. Data throughput and repeatability are the most important parameters.
- Transparent mode
  In this mode an online connection to the process is available. However the system is not influenced. Offline and online data are available and will be switched depending on the time index. This is transparent to the connected simulator application. Transparent mode is designed to be used for online and offline verification and process supervision e.g. soft sensor applications. In this way an online verification procedure can be re-executed on historical data multiple times with deterministic results if the simulator is deterministic.
- Online Mode
  Online mode is used to simulate a process and transmit the results back to the process. The data is directly transmitted between the process control system and the simulator. Therefore latency is minimized although proxy servers may be necessary for large scale simulations.
- Prediction mode
  This mode is the most advanced and primary used for proactive and symbiotic simulation. Advanced synchronisation methods are provided and multiple prediction results over a time horizon can be transferred back to the process. The transparent data source switching feature is also included and extended to be able to reuse previously predicted data. In this mode historical, present and future data are available to the process control system and the simulator application. Redundancy in this operation mode is an important feature to be integrated in industrial process control. Multiple simulation runs can connect in parallel to the data source with different wall-clock time. This is especially useful to simulate models with dead times.

PDSP should be classified as an application protocol (layer 7) in the Open System Interconnection reference model. The protocol encapsulates three inner layers. Figure 1 visualizes this structure. The bottom layer is used to multiplex multiple connections over a single connection. In Ethernet based TCP/IP networks it would also be possible to use multiple connections on lower OSI-layers but the resulting data flows would differ. This happens because of a PDSP assurance. All data flows being multiplexed in one connection can be linked to provide cross connection time-ordered data transmission. This means that no earlier sample or event than the current one can occur. Further no new authentication is processed resulting in lower response times. The layer 1 frame consists

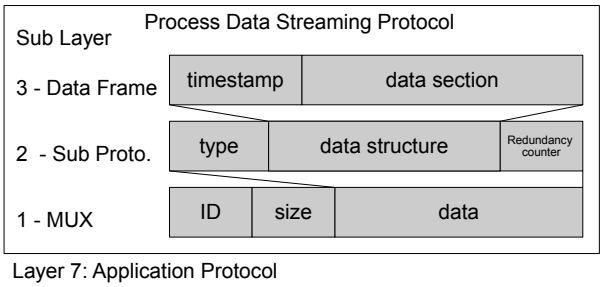

Figure 1: PDSP frame format

of a 32-bit stream id, a data size descriptor and the data part for the next layer. Layer 2 is used to implement different sub protocols e.g. for data transmission or flow management. The complete list will follow later on. The top layer 3 is not used by all layer 2 protocols. It is a data representation frame consisting of a 64-bit time stamp and a well defined data section. The format of the data section is constant for a single transmission stream and is communicated when the stream is initialized. It should be mentioned that the 3 layer structure is not used due to security reasons until authentication procedure is positive.

The PDSP protocol can be extended by protocols in inner layer 2. Although some sub protocols are predefined and necessary during startup operation. When a simulation environment connects to a process using PDSP it must define a minimum connectivity level. This number is used to let PDSP scale from simple embedded systems to Distributed Control Systems. Not every small device is able to handle all sub protocols. For example a dynamic online code loading and execution mechanism is defined in PDSP Java to Java machine communication. Useful to distribute precompiled simulation kernel over a multiple simulation nodes. A simple embedded system without a Java VM would be unable to provide a handler for this. Based on this a set of PDSP schema level is defined. When a connection is established both sites transmit their own maximum schema level and compare the result with a for this connection required level. The highest possible level is used for further communication. Note that it is possible that no connection can be established if the requested features are not provided. PDSP protocol is defined as a symmetric protocol. This means that there is no server and both sides provide and use exactly the same interface. PDSP defines multiple sub protocols:

- Data Stream
  This is the basic protocol for stream based data transmission. Layer 3 frames are transmitted.
- Flow management
  Used to establish, close, rewind connections, adjust connection properties like buffer sizes and transmit control commands e.g. end of stream if no online source is connected.
- Multiplex management

Manages new layer 1 connections. At the beginning a single default layer 1 connection is established to be able to use this protocol.

- Redundancy management
  Distributes a layer one connection on multiple PDSP connections to provide redundancy. If more than 2 connections are used a voting procedure is used.
- Synchronisation
  Provides a fast synchronisation path to read remote system timer and delays. Protocol frames of this sub protocol are transmitted with the highest priority.
- Remote Methode Invocation (RMI)
  Each application using PDSP can provide own RMI based interfaces. These are compiled to a binary command protocol simple enough to be used by 8-bit embedded systems. For example the proxy and database server of the later introduced framework uses this mechanism. Three level of operation are supported. Higher levels support more complex data structures for parameters and results.
- Online code loading
  If a PDSP Java to PDSP Java connection is established code which is not present on one side can be obtained form the corresponding partner during RMI usage.

The authentication is defined separately and always enforced. It is based on a certificate store or passwords depending on the PDSP schema level. Table 1 explains the used sub protocols corresponding to a schema level.

FRAMEWORK

The PDSP-Framework implementation consists of a central server, multiple connectors, a set of data processing modules, a management console and a report generator. In this section we present implementation details of these components. The central design pattern for data access is a stream based one. Its structure is presented in Figure 2. Although streaming interfaces usually only use one direction in data flow, this interface includes the necessary control commands in the opposite direction. In Figure 2 data flows from left to right. Status indicators e.g. end of stream are transmitted in parallel. Control commands use the opposite direction. Each stream has a beginning and an end; cycles are not allowed. To be able to use a data stream a stream-controller has to be attached. It is automatically attached at the correct position which is usually at the beginning. Some stream components e.g. buffers attach their own controller on their input side and capture events from the originally attached controller. For example some buffers read bigger chunks of data in order to decrease latency when historical data is accessed. It should be mentioned that this interface is identical to the combination of the data stream and flow control subprotocols of the PDSP definition. So each connection can be mapped over a network via PDSP or executed directly inside one machine.

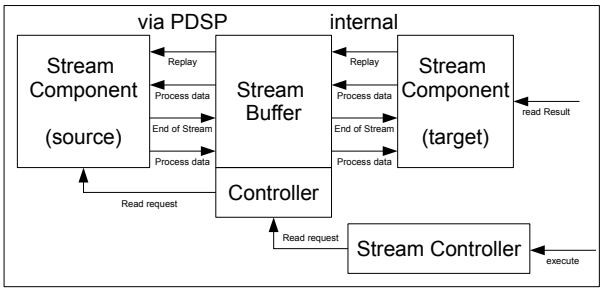

Figure 2: PDSP Stream Interface

Next we take a look into the central server. This server is combining features useful for standard employment scenarios. It is written in pure JAVA version 1.6 and can be extended by external jar-files. The structure is visualized in Figure 3. The central component is the Java implementation of the PDSP. There are several other modules accessible with the help of this interface:

- Database
  This component stores historical or predicted data streams. The back-end for this implementation is the relational database Derby. The Structured Query Language (SQL) interface of Derby is completely hidden. A stream based interface is used to hide all database operations. In symbiotic simulation environments the addressed data frequently is out of a defined sliding time window. This statement is supported by the database by the usage of a memory buffer. It provides fast access to recent data. Furthermore the database can be replicated to a remote database via PDSP without interruption.
- Proxy
  The server proxy can be used to split data streams over all connected PDSP links. In combination with the database component it is possible to access data streams at any time stamp where data is present. In symbiotic simulation this is can be used to spread the calculation across multiple compute nodes or implement security filter. Security filters are used to eliminate illegal output produced by a simulator.
- Catalog
  All PDSP based nodes connected to a server can announce services or data streams at a central place. The catalog can supervise data stream endpoints or RMI-interfaces.
- RMI Server
  The Remote Method Invocation server component is used to load Java based extensions provided by the user in a jar-file. The component must define an interface to be accessible via PDSP and can dynamically provide data stream endpoints to the catalog. Java based data connectors can use this to connect external systems.

To connect the framework to data sources usually a data connector has to be implemented. For faster eval-

Table 1: PDSP Schema Level and Features

Level	Authentication	Data Streaming	Multiplexing	Redundancy	Synchronisation	RMI	Online Code Loader
9	Yes	Yes	Yes	multi-path	Yes	full	Yes
8	Yes	Yes	Yes	partial	Yes	full	Yes
7	Yes	Yes	Yes	partial	Yes	full	No
6	Yes	Yes	Yes	partial	Yes	normal	No
5	Yes	partial	Yes	partial	basic	normal	No
4	Yes	partial	Yes	no	partial	basic	No
3	password	partial	Yes	no	partial	basic	No
2	password	partial	Yes	no	no	basic	No
1	password	partial	no	no	no	basic	No

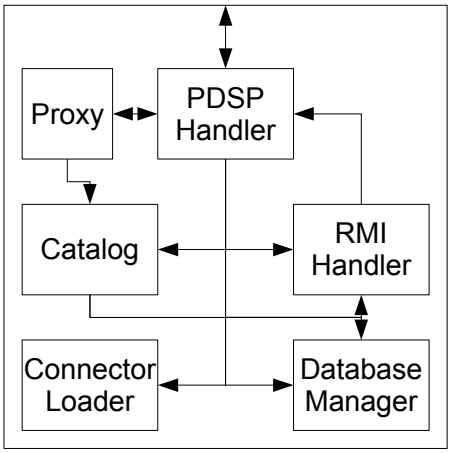

Figure 3: PDSP Server Structure

uation and easy usage some default connectors are implemented. The first one is a wrapper implemented in Java and uses the RMI server interface. It connects one or more serial devices or files. The in-/output format is equivalent to a comma separated text. The second more advanced connector is primary used to connect industrial equipment via OLE (Object Linking and Embedding) for Process Control (OPC-Task-Force, 1998). Because OPC uses the DCOM (Distributed Component Object Model) interface for communication this component is not written in Java. As PDSP is not limited to Java this connector uses C# and the .NET framework for OPC access via PDSP over TCP/IP. OPC is a popular interface in industrial automation equipment. Some of the functions are similar to the PDSP. Data can be read and written to items. The current PDSP-framework extends this data interface by the possibility to use other operation systems and programming languages, a transparent historical data access, reusable data processing components and so on. For industrial symbiotic simulation the performance of OPC is usually not sufficient. The third connector is still under development and enables a MSP430 microcontroller for PDSP access.

A core feature of the framework are the rich data processing components. All of these follow the stream pattern in Figure 2. In total there are more than 50 blocks implemented. Due to space constraints only four examples are presented. A continuous data endpoint transfers data samples with time stamps.

- Sample rate conversion filter
  For some simulations and analytic methods it is necessary to provide equidistant samples. Continuous signals have a bandwidth defined by the largest gap between two signals. The sample rate conversion filter uses an approximation algorithm to output higher rate equidistant data streams.

- Alignment filter
  This filter handles multiple continuous data streams in parallel. If a Distributed Control System (DCS) is this data source for the simulation usually not all sources provide a sample at the same time. For simulation it is sometimes useful to read the current system state at a well defined time. The alignment filter corrects this behaviour. It write the interpolated state to each output at fixed time stamp (with regard to the signal bandwidth).
- Forward filter
  A simple but sometimes useful filter. Some connected systems output data samples in incorrect temporal order. This filter has an internal buffer to provide the right ordered samples within a range and drop other samples.
- Event to RMI
  This stream component connects events arriving from a PDSP discrete stream to a PDSP remote method invocation interface. This is useful to control embedded simulations by the usage of the already existing DCS.

As mentioned before the PDSP framework includes a management console. Especially during Java development it can be used to test all RMI interfaces without extra implementation overhead. The structure is automatically generated from the Java class files. Special annotations can be used to provide extra information e.g. for the Linux style 'man' command. The console can be remote accessed only via PDSP to enforce authentication. It is enabled with a command line option in the framework server. Scripted command sequences are supported.

Last but not least the framework provides a interface for reporting. It is based on the eclipse Business Intelligence and Reporting Tools (BIRT). The Java PDSP implementation can be used to connect to BIRT using an Open Data Access connector (Weathersby et al., 2006). By the help of this tool chain high quality reports can be generated online during simulation and for example exported to a web server. Because reporting is often application specific to the simulation model, it is good to use the graphical BIRT editor for this. Almost no programming skills are necessary if the used simulator is supporting the framework database server. For symbiotic simulation the online reporting feature is essential to observe the performance of the hole system. With Java annotations all RMI interfaces can also be marked for ODA/BIRT usage. The corresponding stubs are automatically generated.

PDSP IN SYMBIOTIC SIMULATION

The rapid changing environment of manufacturing processes such as paper manufacturing requires a framework to identify process parameters and to optimize the process itself. State-of-the-art simulation systems (Low et al., 2007) provide no closed-coupled integration into the entire process. Even it concerns an on-line-simulator it does not emphasise this close relationship between the simulation system and the physical system. The paradigm of symbiotic simulation systems, defined at the Dagstuhl seminar (Fujimoto et al., 2002), reflects this interlocking of hard- and soft-information assembled inside the framework. Therefore the symbiotic simulation approach allows an interaction between the physical system, the automation and the simulation in an efficient and beneficial manner. To point out the characteristics of the PDSP-framework and to present its symbiotic characteristics we describe our work in developing this symbiotic simulation framework for a manufacturing process application.

In automation each technical process consists of three different parts: The process itself, the control equipment like control loops, PLCs (Programmable Logic Controller), process computer and the interface between process and automation, depicted in Figure 4. The whole automation environment is connected to an archive database. To achieve the project objective it is necessary to integrate the Hybrid Process Net Simulator framework into the entire automation environment. Therefore using the HPNS framework with a manufacturing or in more general with an arbitrary technical or functional process it is essential to link the HPNS framework with the process itself. There are different scenarios possible to realize this linkage and to fit the framework to the automation system. All scenarios provide another system perspective, for example the offline- or the online analysis. In the following sections the different modes of operation, introduced in section 2 are described without specifying the details of the connection.

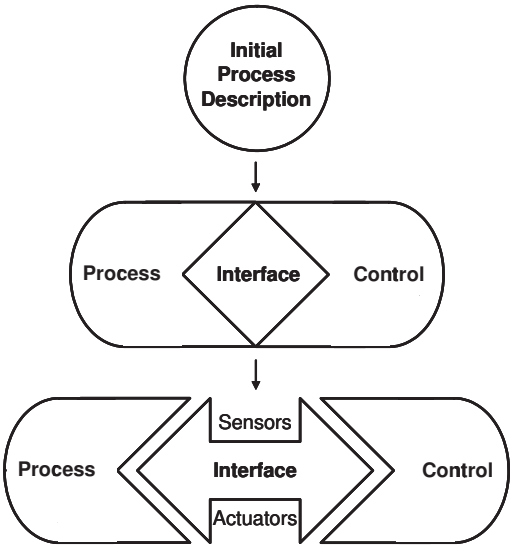

Figure 4: Process Decomposition

By using the analytic mode the HPNS framework can verify offline given scenarios based on available process data saved in the archive process database. This scenario provides the chance to evaluate the process behaviour and the process stability in special operating points. It avoids critical process states and supports the process operators to evaluate and to analyze the process characteristics and the process progression in more detail. To realize this it is necessary to connect the HPNS framework to the archive database. In addition a second database for triggering special states by using a list of event, called base sequences, is necessary. Based on this trigger data, special process working points can be defined for evaluation and analysis.

The transparent PDSP mode embeds the HPNS framework into the entire process without interaction. Here the HPNS is executed concurrent to the running manufacturing process. Due to the flexible architecture of the HPNS framework different options of embedding are possible. Every option shows other characteristics and provide a special point of view to fit different demands of interest. The two most interesting terms of embedding are introduced here:

- HPNS and automation (Figure 5, 1))
  HPNS is modelling the process on a required abstraction level. Therefore it is possible to connect the HPNS framework to the automation system and to replace the manufacturing process by the executed model. This is the reference scenario for the verification and testing of the automation system.
- HPNS and process (Figure 5, 2))
  As shown in Figure 5, 2) the process model consists of three different parts: The process, the control equipment and the interface between them. The HPNS framework is able to replace both entire parts or different subsystems to provide a type of development framework. Consequently, based on the

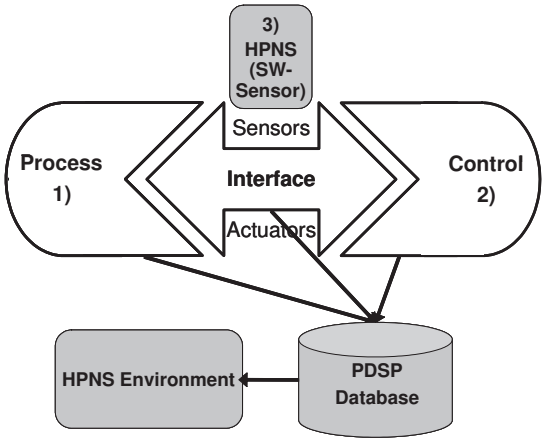

Figure 5: PDSP Process Integration

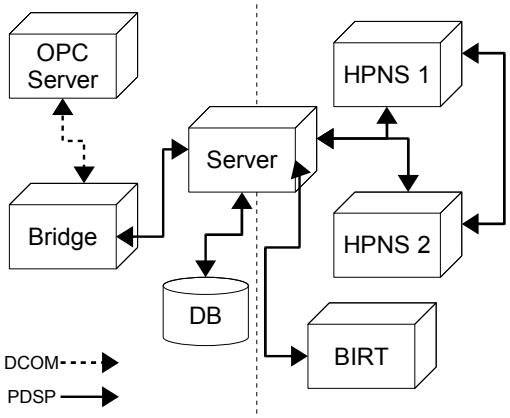

Figure 6: Data Flow

model the HPNS framework simulates the automation equipment.

The online mode is used to interact with the process. This is used e.g. for the SW-sensor capability (Figure 5, 3). The considered manufacturing process is realized based on state-of-the-art technology. Therefore not all process data can be achieved online, due to installation space, process dead-times, sensor failure et cetera. Soft sensors are inferential estimators, drawing conclusions from process observations when hardware sensors are unavailable, unsuitable or out of order. Based on a process model soft-sensors for monitoring and control of manufacturing processes requires an online model execution which can be realized using the HPNS framework. Therefore it is possible to deduce special process data from the process model even if the physical sensor is broken.

The prediction mode is the most complex interaction between simulator and process. Here the PDSP framework in combination with the HPNS is used to execute symbiotic dynamic models. In this mode the simulation environment can take advantage of low PDSP overhead to clone simulation instances for prediction (Bradley and Levent, 2008). In symbiotic simulation a signal feedback is provided using different voting algorithms. The application of modern SIMD Stream-Processors seems under real time restrictions promisingly and PDSP access had been implemented.

INTEGRATION EXAMPLES/RESULTS

To be able to get some first results and test the flexibility of the PDSP-framework it is integrated to the Hybrid Process Net Simulation environment. The HPNS-framework consists of different modules including one for model fitting and a simulation kernel for symbiotic simulation. Here we present two different experimental arrangements. The first one uses the framework database server connected via the PDSP-OPC-Bridge to a DCS in a paper mill. Over a few month about 1 TB data had been collected from 6,000 different data points. The data types were scalar and vector floating points. Then we connected a compute node via Gbit Ethernet to the server and applied a chain of data filters to the streams. These configuration is shown in Figure 6 on the left side. The single core database server could output a 190 Mbit constant data stream in analytic mode. This is equivalent to nearly 10,000,000 64-bit samples per second in random access to the different streams. But the compute node was unable to process this. The bottleneck as expected is the used alignment filter processing about 8,000 samples per second on a modern quad core machine. This indicates the usability of the framework in data analysis. The data throughput of two matching systems in common operation tasks should not be limited by the PDSP data transfer rates. For very simple computations the operation could also be invoked directly on the database server. This would reduce the transmission overhead.

The second scenario uses a USB-PLC connected by a specific implemented PDSP-bridge to an interface node. The PLC is interfacing a two tank system. A database server stores historical data a second transparent server connects the bridge in-line with an instance of the HPNS-Kernel. This second connection chain is using the predictive PDSP mode. Continuous sensor data e.g. water level or temperature and discrete events (for example limit switches) are transmitted to the simulator. Inside the HPNS are two models implemented. One for the two tank system and a second for the control logic (see Figure 5). Both are simulated in parallel. This structure is visualized in Figure 6. The control commands generated by the HPNS-Framework controlling the heater and the pumps are written back to the PDPS connection. In this experiment HPNS reverts time of the input data about 100 times a second to predict 100 possible behaviours. During operation no data transmission interruption could be measured. As expected the chronological access pattern enforced by the PDSP-server in prediction mode fits the data input behaviour of the HPNS-kernel. For test purposes this enforcement was disabled resulting in data transmission gaps of about 2ms. In combination with 100 occurrence this results in a 20% idling simulation

node. Beside the extra knowledge gained by chronological transmission order this indicates that for common simulation kernel a better performance could be generated. The round trip time for this control loop is 19ms in average. It is dominated by the two 8ms average delays in USB transmission. The four PDSP-links remain below 1ms each. This behaviour also proves that the sliding memory buffer of the PDSP server works because typical I/O-subsystems have higher access times.

CONCLUSIONS

The paradigm of symbiotic simulation environments can generate benefits in manufacturing processes. On one side symbiotic simulation environments have a distinct need of state of the art connectivity concerning performance and efficiency. On the other side the interfaces are getting more heterogeneous while simulation tools are used in more and more application areas. In this paper we introduce the PDSP framework as a possible solution. It couples tightly the data management and system connectivity. Furthermore online reporting for symbiotic simulation is part of the PDSP framework. While the framework is still under development we are able to obtain first performance parameters. In near future especially the consistent connectivity for embedded systems and radio based ad hoc networks will be enhanced. This will allow symbiotic simulation in new application areas using data from small wireless nodes.

REFERENCES

Bohlmann, S., Klinger, V., and Szczerbicka, H. (2009). Hpns : A hybrid process net simulation environment executing online dynamic models of industrial manufacturing systems. In Rosetti, M. D., Hill, R., and Johannsen, B., editors, *To be published in: Proceedings of the 2009 Winter Simulation Conference*. WSC Foundation.

Bradley, M. and Levent, Y. (2008). Symbiotic adaptive multisimulation: An autonomic simulation framework for real-time decision support under uncertainty. *ACM Transactions on Modeling and Computer Simulation*, 19.

Fujimoto, R., Lunceford, D., Page, E., and Uhrmacher, A. (2002). Technical report of the dagstuhl-seminar grand challenges for modelling and simulation. Technical Report No. 02351, Schloss Dagstuhl, Leibniz-Zentrum fr Informatik, Wadern, Germany.

Low, M. Y. H., Turner, S. J., Chan, L. P., Lendermann, P., Buckley, S., Ling, D., and Peng, H. L. (2007). Symbiotic simulation for business process re-engineering in high-tech manufacturing and service networks. In Henderson, S. G., Biller, B., Hsieh, M.-H., Shortle, J., Tew, J. D., and Barton, R. R., editors, *Proceedings of the 2007 Winter Simulation Conference*, pages 576–586. WSC Foundation.

OPC-Task-Force (1998). Opc overview 1.00. Technical report, www.opcfoundation.org.

Weathersby, J., French, D., Bondur, T., Tatchell, J., and Chatalbasheva, I. (2006). *Integrating and Extending BIRT (The Eclipse Series)*. Addison-Wesley Professional.

Zeigler, B. P., Praehofer, H., and Kim, T. G. (2000). *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*. Academic Press, San Diego, USA, 2 edition.

AUTHOR BIOGRAPHIES

SEBASTIAN BOHLMANN is a Ph.D. candidate at Department of Simulation and Modelling - Institute of Systems Engineering at the Gottfried Wilhelm Leibniz Universität Hannover. He received a Dipl.-Ing. (FH) degree in mechatronics engineering from FHDW university of applied science. His research interests are machine learning and heuristic optimization algorithms, complex dynamic systems, control system synthesis and grid computing. His email address is <bohlmann@sim.uni-hannover.de>.

VOLKHARD KLINGER is a professor for embedded systems and computer science at the university of applied science FHDW in Hannover and Celle since 2002. After his academic studies at the RWTH Aachen he received his Ph.D. in Electrical Engineering from Technische Universität Hamburg-Harburg. During his 8-year research activity at the Technische Universität Hamburg-Harburg and research centres he focused on parallel and reconfigurable systems. He teaches courses in computer science, embedded systems, electrical engineering and ASIC/system design. His email address is <Volkhard.Klinger@fhdw.de>.

MATTHIAS BECKER is researcher and lecturer at the Department of Simulation and Modelling - Institute of Systems Engineering at the Gottfried Wilhelm Leibniz Universitt Hannover. He received his degree in Computer Science in Wrzburg 1996 and his Ph.D. from University Bremen in 2000. His research interests are simulation, modeling and optimization of discrete event systems. His email address is <xmb@sim.uni-hannover.de>.

HELENA SZCZERBICKA is head of the Department of Simulation and Modelling - Institute of Systems Engineering at the Gottfried Wilhelm Leibniz Universität Hannover. She received her Ph.D. in Engineering and her M.S in Applied Mathematics from the Warsaw University of Technology, Poland. Dr. Szczerbicka was formerly Professor for Computer Science at the University of Bremen, Germany. She teaches courses in discrete-event simulation, modelling methodology, queuing theory, stochastic Petri Nets, distributed simulation, computer organization and computer architecture. Her email address is <hsz@sim.uni-hannover.de>.