from: "System Design Automation - Fundamentals, Principles, Methods, Examples", Kluwer Academic Publishers, Maerz 2001, pp.185 ff. Editors: R. Merker and W. Schwarz

Object Oriented System Simulation of Large Heterogeneous Communication Systems

Uwe Hatnik, Jürgen Haufe, Peter Schwarz
Fraunhofer Institut für Integrierte Schaltungen, Germany
email: hatnik@eas.iis.fhg.de

Abstract

Communication systems consist of many soft- and hardware components with a wide range of parameters which affect mainly the provided quality of service. One of the main challenges for configuration and structuring such a heterogeneous system is to guarantee the specified quality of service with a minimum of costs. In this paper, we introduce a simulation based approach which helps the designer to determine the best fitting parameter values. Our approach combines prototyping and simulation in a common environment.

1 Introduction

Modern communication systems are very complex heterogeneous systems realizing world-wide video and audio communication and using different networks and protocols with a specified quality of service. Such communication systems consist of servers and clients. Especially clients are very different user devices, from powerful personal computers to small cellular phones. A client can communicate with other clients and servers, using services like live video conferences or it can store and can demand video and audio records (see also Figure 1). Clients and servers are connected by common local and wide area networks.

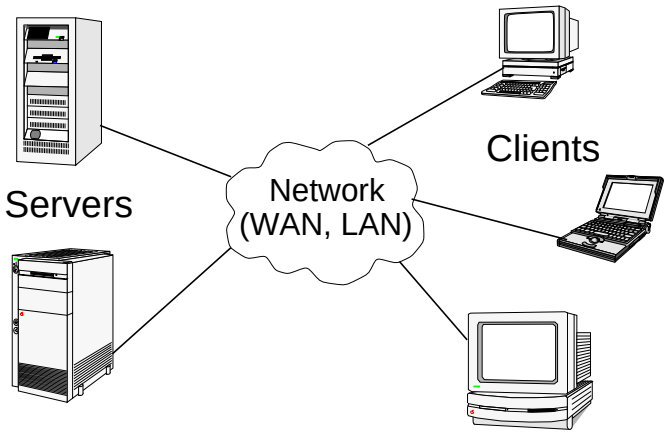

Fig. 1: Basic elements of a communication system.

One of the main challenges for configuration and structuring of such a heterogeneous system is to guarantee the specified quality of service with a minimum of costs. The designer may meet the challenge by using his practical knowledge or by building up prototypes or by utilising formal methods such as performance analysis and simulation.

In this contribution, we introduce a simulation based analysis approach which combines the fore-mentioned analysing methods. In our approach both simulation models and real hardware and real software prototypes can be executed in a common environment. Results of the application of formal methods may be integrated into the simulation models, e.g. distribution functions, profiling results as well as measured values. The approach was driven by our experience that only a mix of different analysis methods which complement one another may bridge the analysis gap of such huge heterogeneous systems.
The text is organized as follows. Section 2 details the analysis requirements of the system we focus on. Section 3 gives an overview of our modelling approach. Implementation aspects are described in section 4.

2 Requirements in communication system design analysis

Clients and servers of a communication system consist of software and hardware components like real-time and non real-time operation systems [3], conference system applications [1], data bases, network processors [5], data processing units, coprocessing hardware[2] [4], RAM, and persistent memory. Some parts of the system are already implemented, others exist only as abstract or detailed models. The system designer has to find an optimal configuration in accordance with the requested parameter. He has also to determine all important parameters of a given system configuration to check the system reserve. The main target is then the determination of an optimal system load with minimal soft- and hardware costs as well as low power dissipation. To figure out the system parameters the designer can measure, analyse and simulate the whole system or parts of it. In practice a complete assessment of the whole system will only be possible with a mixture of these methods. Therefore we propose a procedure that combines all three methods, e.g measuring the already implemented components and analysing or simulating the non-existing parts in such way, that the approaches complement each other. The main objectives of such an simulation environment are:

Parameter determination: A lot of parameters influence the system behaviour. One goal of the system simulation is to find optimal parameter values for a special configuration. Some parameters are specified by the service demand, for example the video resolution, the number of colours, the net bandwidth, and the used network protocol. Other parameters depend on the computer used, like CPU performance, memory size and so on. Additionally there are software parameters like buffer size and the used algorithm for data processing. There is a large amount of parameters and the optimal configuration is very system specific. Therefore the parameters can not determined completely analytically.

Performance analysis: Since optimal system parameters can hardly be determined only analytically, simulation is also important for examining system performance depending on the hard- and software parameters.

Configuration analysis: The configuration of a server or client depends on the demanded service and the client system. For example a specific data compressing algorithm is used depending on system parameters of the client like CPU performance and memory size. There are a lot of possible hard- and software combinations. It would be useful to determine what combination is suitable for a special service and configuration.

Relationships between the components: There is a more or less tight correlation between the components of a configuration. For that reason, the system has to be treated as a whole. For example, swapping out parts of the software to hardware would decrease the load of the CPU, but

on the other hand, this would lead to a higher load of the system bus since of the gaining effort for communication. The simulation of the whole system allows detailed analysis of the mutual dependencies and makes it possible to separate between important and non relevant correlations.

Data transport mechanism: Data packets are exchanged between the system components. The size, organization and management of these data packages have a great influence on the system load. The simulation will help to find out and test suitable data representations and their dependence of the component granularity.

Framework: For the practical development the simulation framework will be useful for design, implementation and test of real hard- and software components, because real components can be integrated in the simulation framework.

3 Object oriented system modeling

In a first paragraph, we introduce typical system architectures of a communication system by an example. For simulation based analysing of such system architecures, the system architecture has to be transformed into a simulator specific simulation model using various modelling languages (e.g. SDL, VHDL, Verilog, Modelica). In a second paragraph, we show methods how the system architecures can be mapped into a simulation environment. Therefore, we prefer the **object oriented modelling** approach, which keeps the system hierachy, the system structure and the system component interfaces untouched when mapping the system architecture into a simulation model. Consequently, the system architecture is found again in the model structure. In fact, the object oriented modelling reduces the modelling effort as well as the error-susceptibility of the modelling process. The modelling approach includes the incorporation of existing real hardware or real software into the system model to decrease the modelling effort or to increase the simulation performance. The methodology used for the real component incorporation is more detailed explained in [9] [10].

System architecture

Figure 2 shows typical hardware structures of real clients and servers. The client in Figure 2 a) is a common personal computer. The CPU and RAM exchange data over the PCI bridge. Additional components can be coupled over the PCI system bus which is connected with the PCI bridge and provides some PCI ports (slots) for the components. In each common PC this will be a hard disc controller (e.g. SCSI) and a graphic adapter. Moreover, a conference system client needs a video card to connect a video camera to the system and a net adapter to realize the network connection. This configuration can be extended by special hardware e.g. a network processor, FPGA based prototypes or custom hardware. The purpose of this special hardware is to support the CPU because the conference system software running on the CPU contains a lot of time consuming algorithms for video and audio data processing. This algorithm (e.g. a MPEG coder or decoder) are often very complex and can overload the system resources, e.g. the CPU performance.

Other very different client structures are possible, as shown in Figure 2 b). For example the video camera can include analog and digital data processing units and already provide digital picture data over a standard interface, e.g. USB.

There are depicted two possible server structures in Figure 2 c) and d). Their base structure is similar to the clients one, but normally a server is more powerful than a client, because it has to serve many of clients. A video adapter, like the clients use, is usually not necessary because the

server stores and provides video data but does not generate them. The server software contains a database with all stored records. If a client ask for a video, the server searches it in the database as well as initializes and controls the data transmission if it can provide the requested service.
The network which connects clients and servers may have very different topologies. For example it can be a local area network but also a wide area network like the internet. The transmission way can include different technologies like cable, fiber and radio as well as different protocols.

Model architecture
The system consists of many heterogeneous components, like shown in Figure 2. Our object oriented model approach use this system structure to build a model structure, like illustrated in Figure 3. The component models can consist of a hierarchical structure of submodels and be realized using different abstraction levels.
As briefly described in the paragraph above, there is no unique architecture and no unique data protocol format for a today's communication systems. The system components are very heterogeneous. Therefore, a wide range of modelling means has to be considered to map the system into an simulation environment. Furthermore, depending on the simulation goal, the model abstraction may be high or low. Figure 3 shows as an example some typical system views. The model abstraction starts with abstract queuing models describing statistical data rates and delays and ends with a detailed descriptions of hardware blocks using hardware description languages such as VHDL or software components using programming languages. For system analysis it is important to handle all this abstraction layers within a common environment.
The most abstract model approach is to consider all or some of the clients and/or servers as traffic generators and sinks, like shown in Figure 3 a) and d). For example a media server which

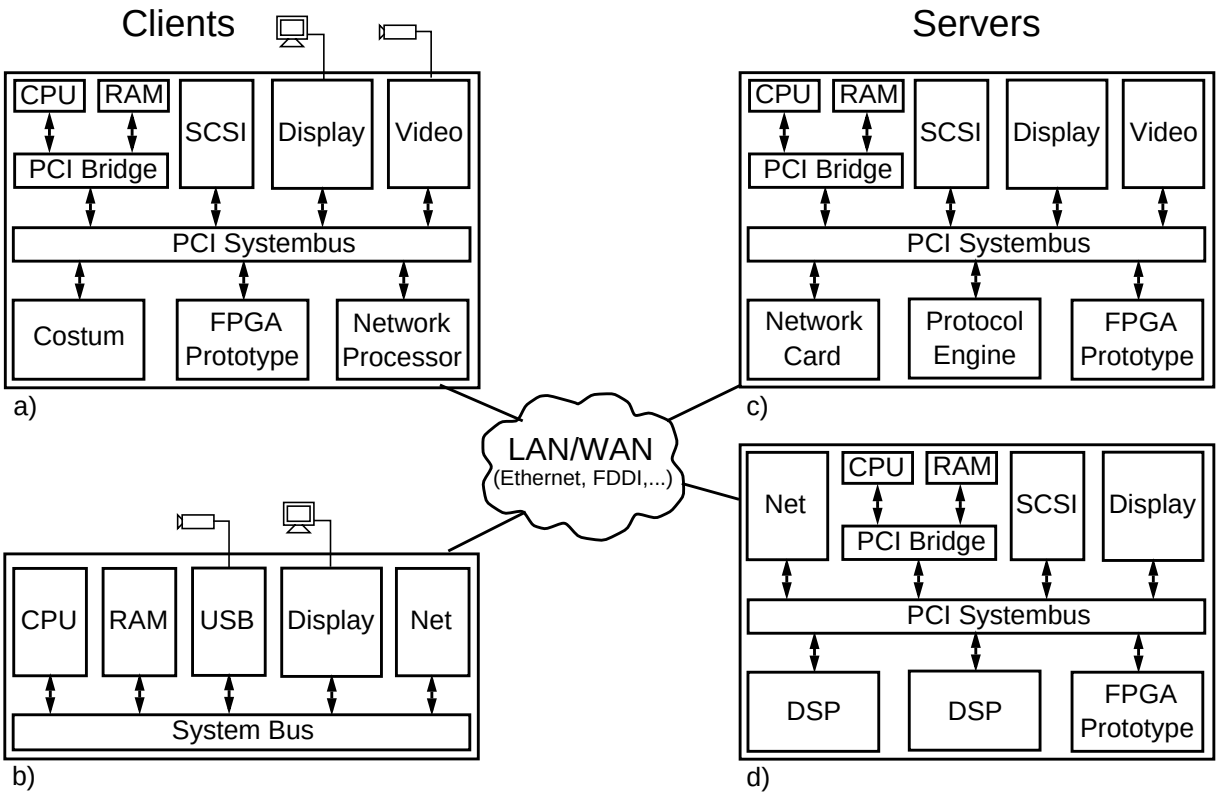

Fig. 2: Client and Server Structure

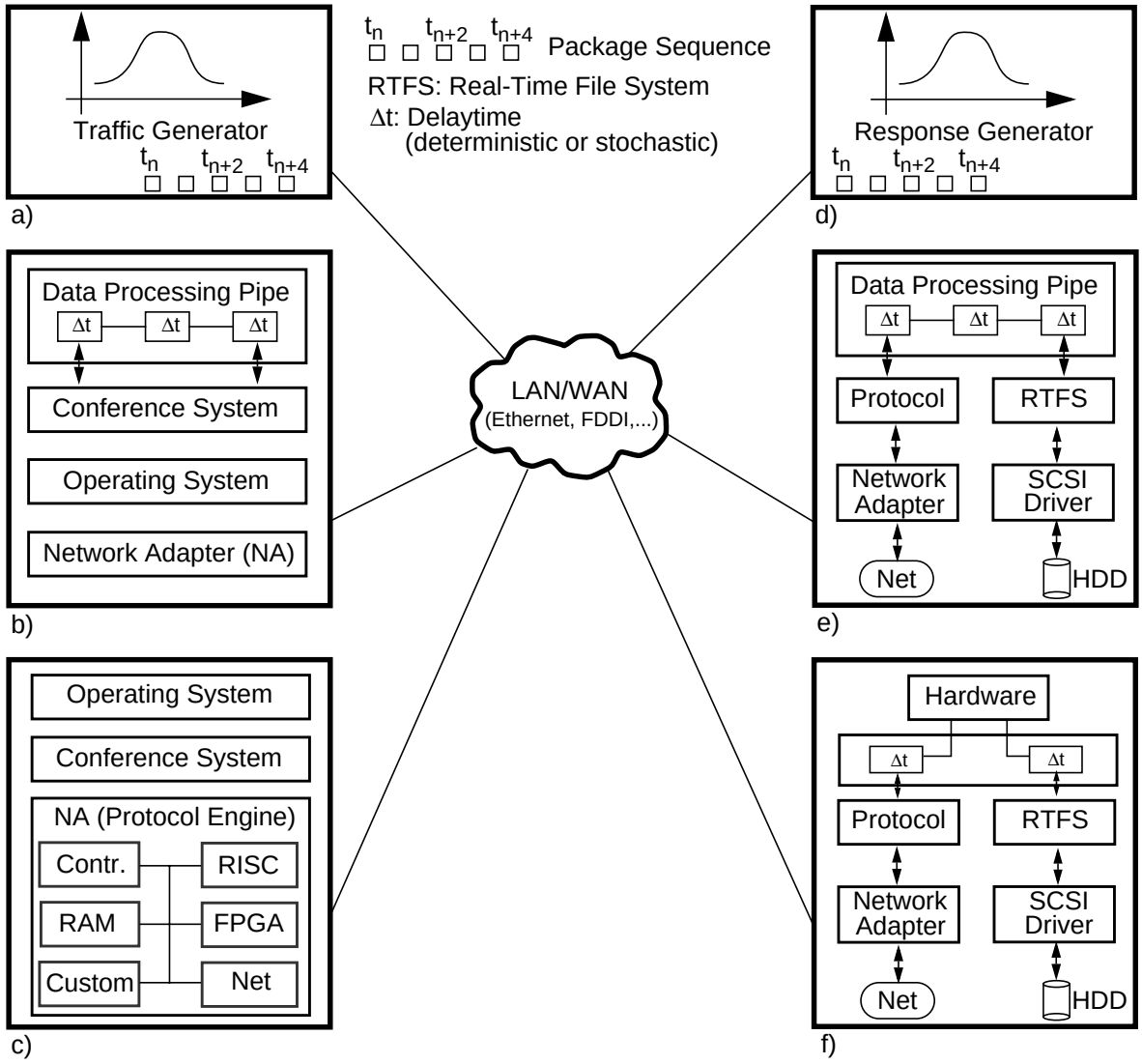

Fig. 3: Model Abstraction Levels

sends data to a client will put data packages in the network and the client will receive it. This process can be described with some parameters like package size and package distribution function. This parameters can be determined analytically or by measuring of a already existing system. Another abstraction level is shown in part b) and e) of Figure 3. In this case the model is built with soft- and hardware components. The exact model structure depends on the client or server configuration as well as on the specific simulation goal. Besides some control components this model approach includes a data processing pipe. This pipe contains all components passed by the data stream. Therefore this model type is e.g. especially suitable to observe the data flow through a data processing pipe and to determine its parameters or to test various pipe configurations. A pipe component can be an abstract model that handle abstract data or also real software.

A more detailed model approach is shown in Figure 3 c) and f). Here, one or more components are modelled very exactly. In the case of clients it is a Network Engine (c) with its components and in the case of severs it is a component from the data processing pipe (f). For example the

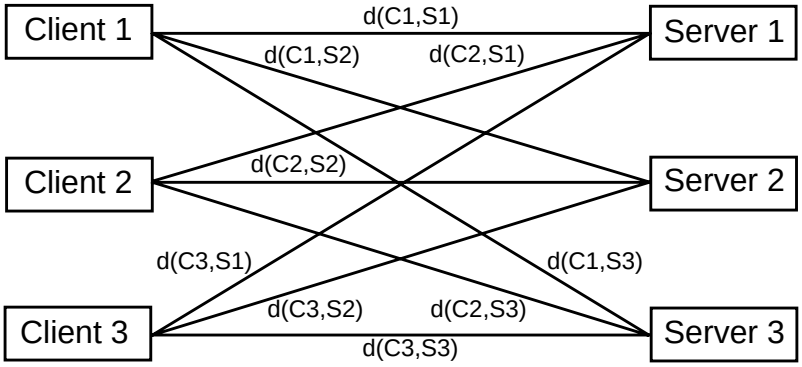

Fig. 4: Connection oriented Network Model Approach

components of the Network Engine could be modelled with a hardware description language (e.g. VHDL or Verilog) or implemented with real hardware.
For the network modelling there exists two basic model approaches: the connection oriented approach (Figure 4) and the topology oriented approach (Figure 5). In the connection oriented approach, each possible connection within a network and the appropriate link delay is described. In contrast, in the topology oriented approach the model includes the network topology which can consist of different network types and their parameters.
If a designer has to develop a new hardware component he can use the environment to simulate it before it really exists. In this case the new component will be in the focus of interest and the remaining system can be abstracted to a data source or sink. Figure 6 shows the system model scenario derived from Figure 3.
Abstract clients and servers as well as the conference system of the focused client work as traffic generators and sinks. They send data packages to the more detailed client model, that is in the focus of interest, e.g. a „protocol engine“. In this example the protocol engine, which realizes the network protocol, consists of six components: PCI controller, RISC, DSP, FPGA, RAM module, network adapter, and bus which connects the components.

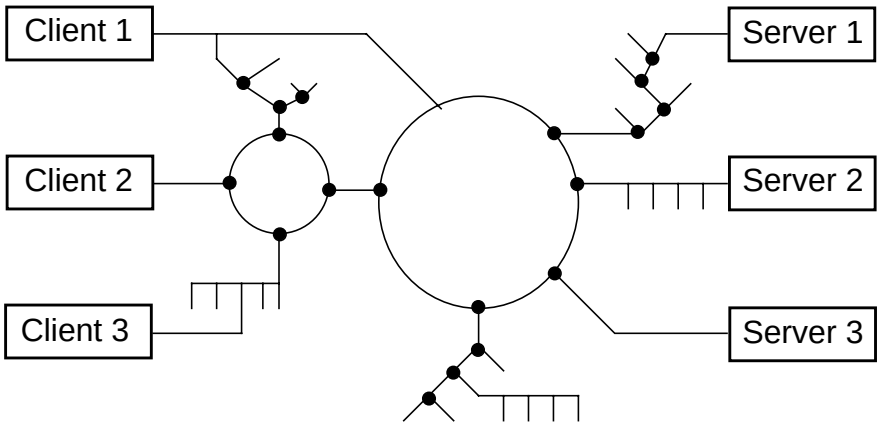

Fig. 5: Topology oriented Network Model Approach

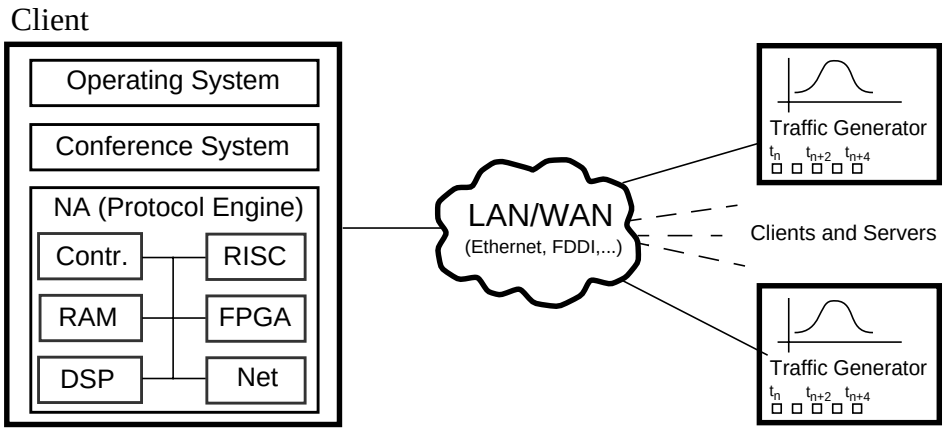

Fig. 6: Example Scenario

4 Object oriented system simulation

As shown above, a communication system contains many heterogeneous components, e.g. the network, client and server hardware components and their software. The simulation of each single component or of component groups is state of the art. A simulator which allows to simulate the whole system is currently not known. Therefore we use a **object oriented simulation** approach to develop a large heterogeneous simulation framework.

4.1 Simulation Approach

In the last years, the object oriented simulation approach has been established and also our approach follows this idea [7]. Object oriented simulation means that a system is split into subsystems, which are simulated autonomously [17]. Such a subsystem is named "object" and may contain other objects, so that an object hierarchy may be built. Figure 7 shows the object structure (a) and an object hierarchy (b). An object embeds its own simulation algorithm, which can be a small code fragment but also an extensive simulator. All implementation details are encapsulated by the object, only an interface allows data exchange and simulation control.
The advantage of that approach is its flexibility. It makes it possible to mix and exchange objects easy, whereby very different simulation algorithms can be combined. Furthermore the simulation algorithms can be optimally adapted to the models.

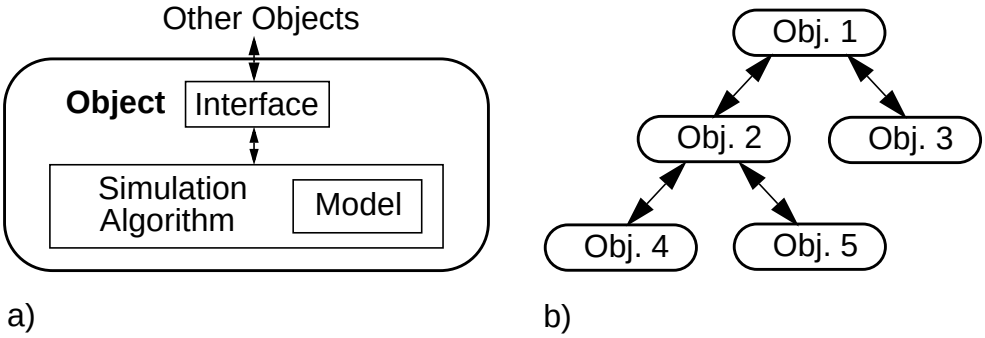

Fig. 7: Basic Object Structure (a) and hierarchical composition (b)

4.2 Simulation tools

Because a single simulator which allows to simulate a large heterogeneous communication system is currently not known, various simulators have to be coupled. We use the simulation framework **Ptolemy** as top-level simulator to combine all necessary models and simulators (objects). Ptolemy follows the object oriented approach described above and allows the simulation of heterogeneous systems, particularly those that mix technologies, including for example analog and digital electronics as well as hardware and software [6]. It is also possible to mix different simulation algorithms like data-driven simulation and discrete-event simulation. The framework provides C++ class structures that can be expanded to build new simulation algorithms and models. It is possible to distribute the simulation task to several computers that are coupled over a network. For example each client and server of the communication system can be simulated on one separate computer.
Theoretically, it is possible to develop models for each system component and simulate all within the Ptolemy framework. However this is not the goal of our work, because some components already exist and other will be developed with specific design tools.
Because some important models are only available in VHDL, we use a VHDL simulator. This permits the simulation of very large models on hardware level. Since VHDL is a very popular and flexible hardware description language, we can fall back on many own and foreign models and model libraries.
The simulation of complex networks requires a suitable network simulator and network models. We use the simulator NS (version 2), because it is very flexible and supports complex network scenarios with many transport protocols, routing mechanisms, application models etc. NS is an object oriented, discrete event driven network simulator developed at UC Berkley written in C++ and OTcl [15]. It is very useful for simulation of local and wide area networks including mobile and satellite networks.
A network scenario consists of three levels. The first level defines the network type and topology (e.g. a LAN with CSMA/CD MAC protocol), using nodes and links which connect the nodes. The second level defines the used transport protocols (e.g. UDP or TCP). Finally the application-level defines applications (e.g. telnet and ftp), which use the transport protocols.
Main simulation goal is to obtain information about the availability, workload, efficiency, and failure behaviour of the simulated scenario, dependent on a lot of parameters (link bandwidth, transport delay, protocols etc.).
The fast simulation of complex processors (e.g. DSP or RISC) requires instruction set simulators (ISS) [14]. An ISS simulates the program running of a processor. The ISS reads commands and data from a memory model, performs the data processing, and writes the results back to the memory. Neither the memory nor the processor is normally modelled on hardware level, but more abstract data processing models are used. This allows a high simulation performance, because the simulation of the exact hardware behaviour is not necessary. The abstract model level requires an adaptation if an ISS has to be combined with more detailed models, e.g. a memory model designed with VHDL.
Figure 8 shows one possible simulator structure. It is a combination of the simulation framework Ptolemy, a VHDL simulator (VSS), a network simulator (NS v2), and an instruction set simulator. Furthermore a real hardware component is included. In this example we assume that the PCI controller, network adapter, conference system, and other clients and server models are available as C code. The FPGA and RAM modules are described in VHDL and for the RISC and DSP are special instruction set simulators available. The concept is not limited to the listed simulators, so also other tools (e.g. a Verilog simulator) can be included.

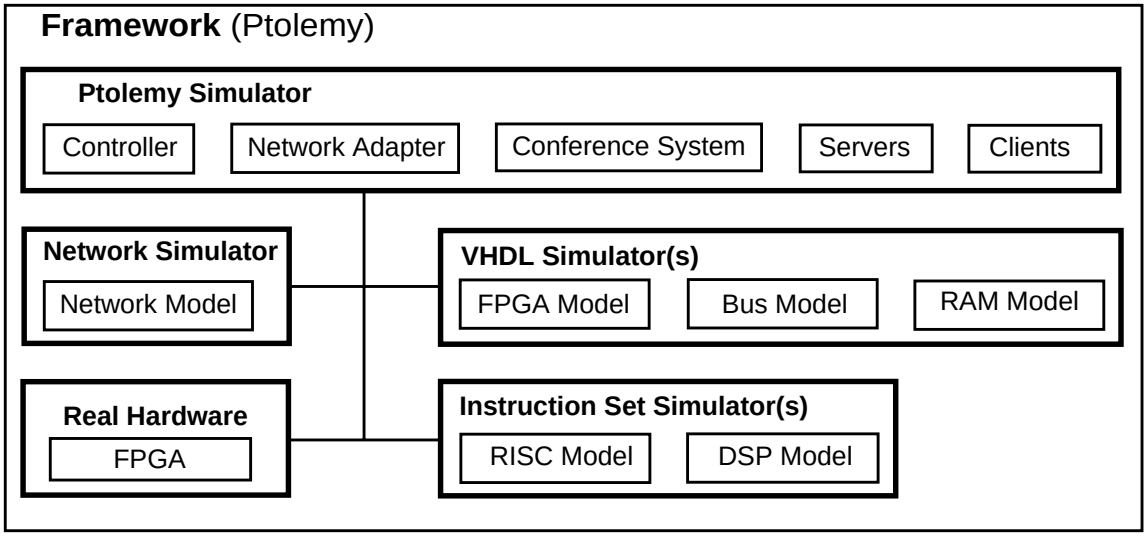

Fig. 8: Example Simulator Structure

4.3 Simulator coupling

As seen in Figure 8, more than one simulator is necessary to simulate the whole system. Therefore, different simulators must be coupled. For simulator coupling different techniques can be used. To simplify this task, we looked for a standardized interface to couple simulators. Commercially, the **backplane** approach is widely spread [16]. The backplane is a separate and simulator independent software to manage the communication between the simulators. As a disadvantage, the backplane produces a high communication overhead so that the relevancy of that technique is decreasing. To overcome the overhead while keeping the independency, vendor and simulator independent standard interfaces between complex simulation models and simulators are being developed such as the **Open Model Interface** OMI [11] [13].
The OMI, since 1999 a IEEE standard (P1499), was developed to couple simulators with complex digital models. It is an open standard interface, which allows interoperability between an application (e.g. a simulator) and functional models, which are presented in a binary form. The models can be developed in a variety of languages, for example VHDL, Verilog HDL, and C. To generate a binary form, a suitable compiler is necessary. Because models are supplied in a binary form, a simple but effective IP (intellectual property) protection is possible. Furthermore, a single model library may contain models of different model providers. These multiple models may be used concurrently during one session.
A separate software component exists between application and models, the OMI model manager. The model manager connects the application with the models and provides built-in or external models. If the application wants to use a model, the model manager creates and manages a unique customized instance, derived from the demanded model. Figure 9 shows the resulting structure. Only the interface between application and model manager is defined by the OMI specification. The connection between model manager and models as well as instances is undefined. This allows a very high flexibility as well as easy model generation and reuse. The model manager mainly provides routines to realize the functionality of the models and to deliver model information. Furthermore the model manager may implement additional functions, for example

version and licence management. A model query mechanism may be used by the application to get model information, e.g. ports, parameters, supported data types, viewports etc. Viewports are model-internal ports, visible from the outside to simplify verification and test. They have to be provided by the model developer and allow to control the internal visibility with respect to IP protection.
The interaction between the model and the application is not fixed but can be adapted dynamically during runtime. A so-called callback mechanism allows to register and remove model function calls (named callbacks), dependent on the required interaction. More information about the goals and advantages of the OMI can be found in [196].
We use the OMI approach not only to couple simulators with functional models, but also with other simulators. This is possible, because the Open Model Interface (OMI) supports the described object oriented simulation approach, so the applied simulators can be encapsulated in an object.

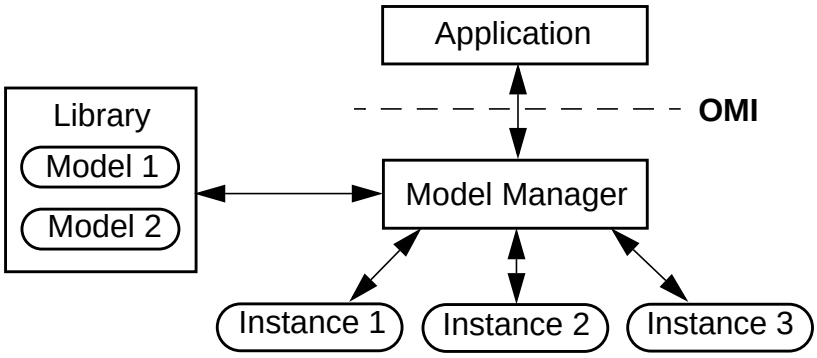

Fig. 9: OMI Basic Structure

In our simulation environment the application is the simulator Ptolemy working as "master" simulator. All other simulators are simulation objects as depicted in Figure 7. Therefore we have developed a suitable interface to couple NS with other simulators, e.g. a VHDL simulator. The interface allows to replace one or more application and protocol models with external models, which can be performed using another simulator, e.g Ptolemy or VSS.
Figure 10 shows the basic structure if Ptolemy objects use external applications via the OMI. The OMI provides the external objects to the Ptolemy models (PCI controller, network adapter, conference system as well as other clients and server). These models can use the FPGA, RAM, and Bus model that runs on an external VHDL simulator, the network model running on the network simulator, the RISC and DSP model simulated with special instruction set simulators, as well as a real FPGA. The Ptolemy models "see" only the Open Model Interface but not the external model implementation.

4.4 Integration of real hardware

In [8], we have shown the possibility to include real hardware into simulation environments via the OMI, exemplarily based on the interface board SimConnect. This board controls the data transfer between a host computer and a FPGA prototyping board. An application programming interface (API) realizes data transfer and control functions, which are executed by the SimConnect board. The SimConnect board realizes data exchange between host and FPGA and controls the FPGA prototyping board according to the called API commands. A more detailed description can be found in [9].

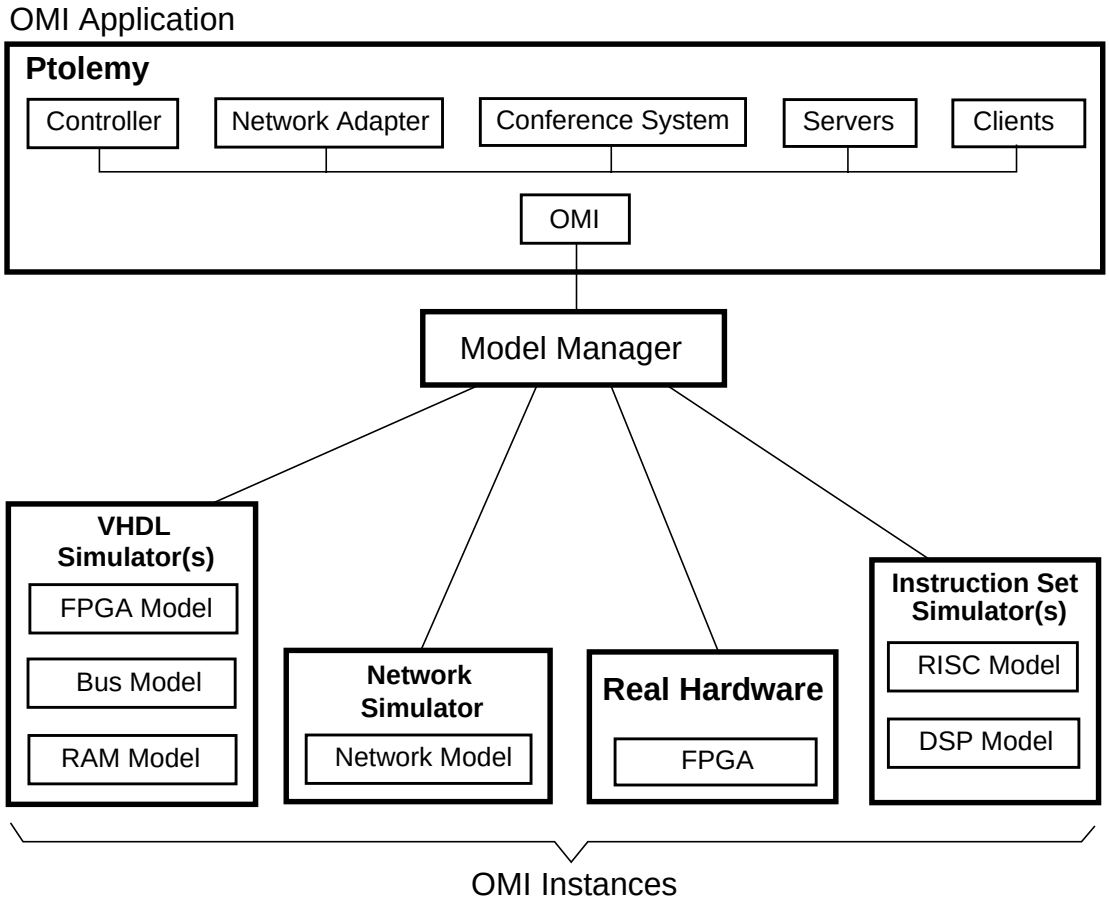

Fig. 10: OMI Example Structure

To include the FPGA board via the OMI, the SimConnect API is encapsulated by a special OMI model. Neither the model manager nor the simulator knows that the model functions are realized using hardware. Besides the FPGA instance the simulator may also create other model instances. If the host provides several I/O-ports, also several FPGA boards can be used at the same time.

5 Conclusion

This contribution describes an approach to simulate a complex communication system. It makes it possible to mix heterogeneous components, that can be modelled using various abstraction levels, from an abstract traffic generator to a very detailed VHDL model for example. The designer can integrate the focused component as detailed as necessary and choose a suitable abstraction level for all other components.
The implementation approach uses Ptolemy as a framework and the OMl as interface to real hardware and external design tools, like VHDL simulators. However this approach is not limited to the OMI or a specific tool.
We have implemented an OMI adaption prototype for the simulation framework Ptolemy, the VHDL simulator VSS and a hardware prototyping board. Additionally, we have developed an OMI model manager, which supports some external models as well as the hardware prototyping board.

Presently we are working on the NS simulator interface, which is necessary to couple NS with Ptolemy via OMI. Because the simulators will run on different computers and platforms, we implement also a connection via sockets, encapsulated by the OMI.
The main objective of our work is to provide a framework, which permits the simulation of large heterogeneous communication systems, using very different model abstraction levels.

6 References

[1] Benz, M., Hess, R., Hutschenreuther, T., Kümmel, S., Schill, A.: A Framework For High Quality/Low Cost Conferencing Systems. 6th International Workshop on Interactive Distributed Multimedia Systems, Toulouse, France, October 1999

[2] Benz, M.: The Protocol Engine Project - An Integrated Hardware/Software Architecture for Protocol Processing Acceleration. SDA 2000, Rathen, Germany, March 2000

[3] Härtig, H., Baumgartl, R., Borriss, M., Hamann, Cl.J. , Hohmuth, M., Mehnert, F., Reuther, L., Schönberg, S., Wolter, J.: DROPS - OS Support for Distributed Multimedia Applications. (Proceedings of the Eigth ACM SIGOPS European Workshop, September 7-10, 1998, Sintra, Portugal)

[4] Fettweis, G., Weiss, M., Drescher, W., Walter, U., Engel, F., Kobayashi, S., and Richter, T.: Breaking new Grounds over 3000 M MAC/s: A Broadband Mobile Multimedia Modem DSP. DSP Deutschland ‘98, October 1998

[5] Level1, IXP 1200 Network Processor, URL: http://www.level1.com/product/index.htm

[6] Buck, J., Ha, S., Lee, E.A., Messerschmitt, D.G.: Ptolemy: A Framework for Simulating and Prototyping Heterogeneous Systems. Computer Simulation, special issue on "Simulation Software Development" vol. 4, pp. 155-182, 1994

[7] Edited by Zobrist, G., Leonard, J.V.: Object-Oriented Simulation - Reusability, Adaptability, Maintainability. IEEE PRESS, 1997, ISBN: 0-7803-1061-6

[8] Hatnik, U.; Haufe, J.; Schwarz, P.: An Innovative Approach to Couple EDA Tools with Reconfigurable Hardware. Field-Programmable Logic and Applications FPL 2000, Villach, 28.-30. 8.2000, 826-829

[9] Haufe, J.; Schwarz, P.; Berndt, T.; Große, J.: Accelerated Logic Simulation by Using Prototype Boards. Conf. Design, Automation and Test in Europe DATE 1998, Paris, 23.-26.2.98, 183-189 (Designer Track)

[10] Fritsch, C., Haufe, J.: HW/SW-Covalidation of Telecommunication Systems in an Industrial Environment. Proc. Workshop on System Design Automation SDA`98, Dresden, 30-31. March 1998, 81-88

[11] IEEE P1499 Technical Committee: Draft P1499 - Standard Interface for Hardware Description Models of Electronic Components. IEEE Standard Department, March 1998

[12] Hobbs, W.: Model Availibility, Portability, and Accuracy - An IC Vendor’s Perspective, Effort and Vision for the Future. Computer Simulation, special issue on "Simulation Software Development", 1995, vol. 4, pp. 155-182

[13] Hatnik, U.: OMI - Eine innovative Modellschnittstelle. Proc. DASS 1999, Dresden, Germany, May 1999

[14] Engel, F.: A Fast and Retargetable Simulator for Application Specific Processor Architectures. SDA 2000, Rathen, Germany, March 2000

[15] Fall, K.: Varadhan, K.: "ns Notes and Documentation", January 2000, http://www-mash.cs.berkeley.edu/ns

[16] Maliniak,L.: Backplanes Mesh Simulators Into One Environment. Electronic Design 42(1994)16, 59-69

[17] Cellier, F. E.: Continuous System Modeling. Springer Verlag, 1991, ISBN: 0-387-97502-0