*Proceedings of the 2023 Winter Simulation Conference*
*C. G. Corlu, S. R. Hunter, H. Lam, B. S. Onggo, J. Shortle, and B. Biller, eds.*

# A VIRTUAL TESTBED FOR THE DEVELOPMENT AND VERIFICATION OF CYBER-PHYSICAL SYSTEMS

Jan Reitz
David Böken
Jürgen Roßmann

Institute for Man-Machine Interaction
RWTH Aachen University
Ahornstraße 55
52074 Aachen, GERMANY

## ABSTRACT

This paper presents a virtual testbed for the development and verification of cyber-physical systems, integrating network simulation, physics, and hardware emulation within the multi-domain simulation framework VEROSIM. The testbed facilitates comprehensive software-in-the-loop testing, enabling accurate and reliable evaluation of control algorithms in complex, interconnected systems. The integrated approach simplifies simulation setup and model management, while allowing natural treatment of mobility and the use of sophisticated physical radio wave propagation models. The testbed also enables the simulation of various fault scenarios, supporting the assessment of system resilience and fault-tolerant strategies. A case study involving a capsule approaching the International Space Station demonstrates the effectiveness of the proposed testbed in capturing the interactions between software, hardware, and physical elements, and verifying the overall behavior of a cyber-physical system under adverse conditions.

## 1 INTRODUCTION

Cyber-physical systems (CPS) emerge as a new class of networked and intelligent systems that interact with the physical world. This range of systems includes autonomous vehicles, smart grids, and space exploration. Reliability, safety, and performance of these complex systems are of utmost importance. Consequently, the development and verification of CPS require methodologies that can facilitate accurate modeling and efficient testing of components across the domains of physics, computation and networking, as well as their interactions.

In this paper, we propose a novel virtual testbed that integrates the multi-domain simulator VEROSIM, (Rossmann et al. 2013), with network simulation capabilities and the Qemu hardware emulation platform, (Bellard 2005), for the comprehensive simulation and analysis of CPS behavior. The testbed enables the seamless coupling of Qemu's network emulation capabilities with the integrated network simulator through the utilization of tap devices and bridges. Additionally, we delve into the specifics of simulating serial interfaces to connect Qemu's emulated peripheral interfaces with VEROSIM's physics simulation, with an emphasis on the Universal Asynchronous Receiver-Transmitter (UART) communication protocol.

The integration of physics, network, and hardware emulation in a virtual testbed facilitates comprehensive Software in the Loop (SiL) testing, allowing for the simulation and verification of the overall system behavior under realistic conditions, without the need for physical hardware. This virtual testbed is useful as an intermediate step to a full hardware in the loop setup that provides most of the realism at a considerable discount, as no real hardware is required and real-time constraints are lifted.

979-8-3503-6966-3/23/$31.00 ©2023 IEEE

To demonstrate the effectiveness and versatility of the proposed virtual testbed, we present a case study involving a capsule approaching the International Space Station (ISS). The capsule's behavior is simulated within the testbed, incorporating various fault injection scenarios to evaluate the system's resilience under adverse conditions. Through this application, we highlight the value of our virtual testbed in accelerating the development and verification of complex cyber-physical systems while mitigating the risks associated with real-world testing.

The remainder of this paper is organized as follows: The requirements for the virtual testbed are derived from a motivational example in Section 2. Section 3 summarizes some related work in this area. Section 4 provides an overview of the virtual testbed architecture and its key components. Section 4.2.1 elaborates on the coupling of Qemu's network emulation with the integrated network simulator. Section 4.4 discusses the simulation of serial interfaces, focusing on UART. Section 5 presents the case study of the ISS approach.

## 2 MOTIVATING EXAMPLE

This work is motivated by the need for a virtual testbed that seamlessly integrates physics and network simulation with hardware emulation to enable comprehensive software-in-the-loop (SIL) testing to verify the overall system behavior of cyber-physical systems. We present a fictional scenario involving a capsule approaching the International Space Station (ISS) to illustrate the importance of such an integrated testbed, as depicted in Figure 1. In this scenario, the control algorithms responsible for the desired approach trajectory are distributed across two interconnected OBCs.

Figure 2 provides a schematic representation of the space capsule components relevant for the approach scenario. One OBC estimates the capsule's position and attitude relative to the ISS based on sensor readings received through a serial interface. The other OBC commands the thrusters via a separate serial interface, relying on the estimated position and attitude data transmitted through the on-board network. The control loop is closed by the mechanical coupling of sensors and actuators via the capsule's body.

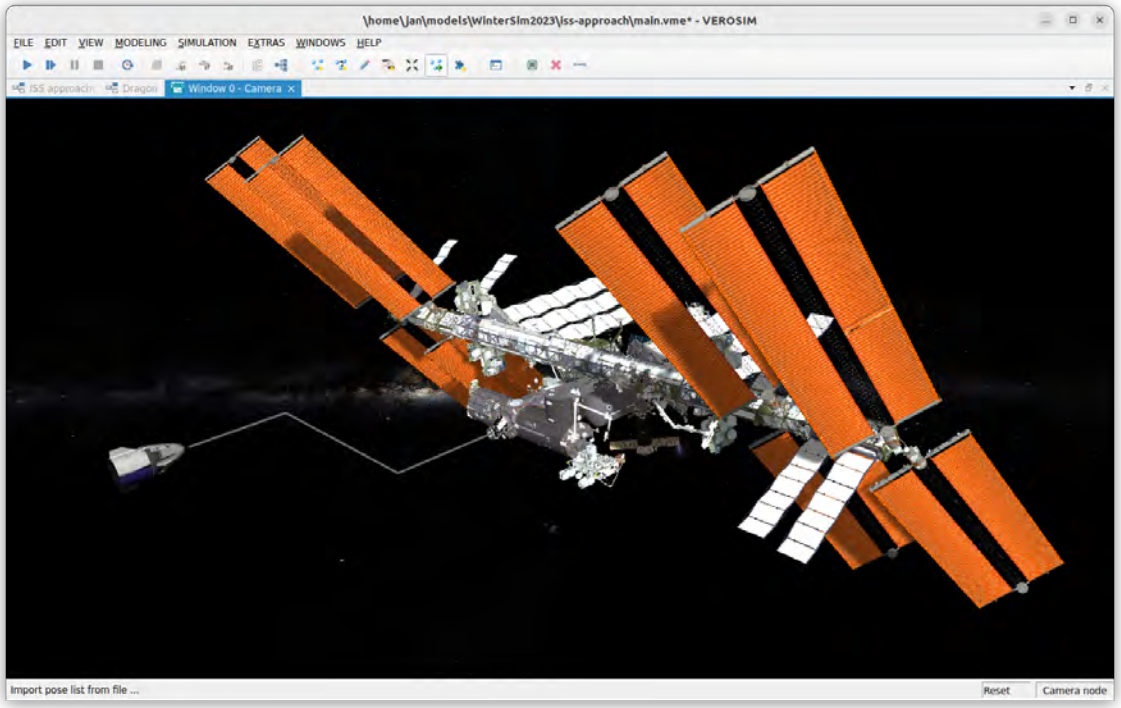


Figure 1: Render view of the application scenario: A capsule approaches the International Space Station (ISS). ISS model courtesy of NASA.

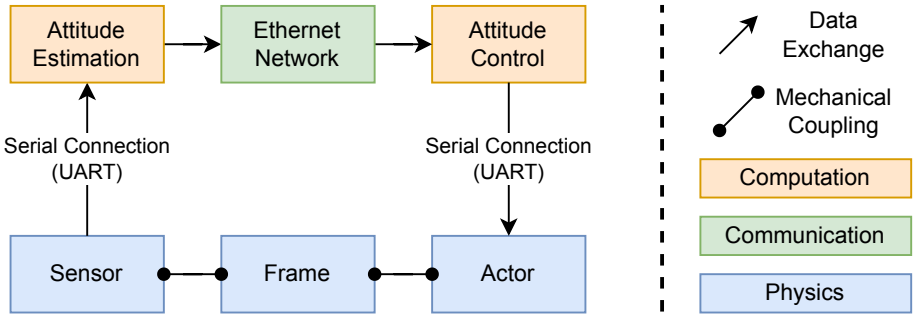


Figure 2: Schematic representation of the capsules components relevant for the approach and the software used to simulate them.

Ensuring the accurate and reliable operation of this distributed cyber-physical system demands a thorough verification of the control algorithms, including their interactions with hardware components and communication channels, as well as their performance under various physical conditions.

By developing a virtual testbed tailored to the requirements of this type of application, we aim to enable the efficient analysis, testing, and validation of control algorithms for spacecraft OBCs in a flexible and cost-effective manner, while ensuring that the interactions between software, hardware, and physical elements are accurately captured and evaluated.

## 3 RELATED WORK

For several years, the development and verification of cyber-physical systems (CPS) has been an active area of research, with numerous approaches and frameworks proposed to facilitate the design, simulation, and testing of such complex systems. The Ptolemy II project, co-simulation techniques and their relation to our proposed virtual testbed approach are some of the works we present in this section.

The Ptolemy II project (Eker et al. 2003) is a well-known framework for modeling and simulating heterogeneous systems, which shares common goals with our virtual testbed approach, such as providing a flexible environment for the analysis of cyber-physical systems. Ptolemy II employs an actor-oriented design with a focus on composing diverse models of computation (MoCs) within a single simulation environment.

Co-simulation techniques have gained considerable attention as a means to simulate cyber-physical systems by combining the strengths of various simulation tools and domains (Gomes et al. 2018). In a co-simulation setup, individual simulators or solvers, each specialized in a specific domain (e.g., control, communication, or physical processes), are coupled to simulate the overall system behavior. Co-simulation allows for the efficient handling of complex, multi-domain systems and fosters modularity, reusability, and scalability. The Functional Mock-up Interface (FMI) standard (Blochwitz et al. 2012) is a prominent example of a co-simulation framework widely used in the automotive and industrial automation sectors.

Unified simulation frameworks, such as Ptolemy II, offer several advantages, including simpler simulation setup and model management, as they consolidate the entire system representation within a single environment. However, these frameworks may lack the specialized capabilities of domain-specific simulators, which can result in reduced accuracy or performance for certain aspects of the system.

In contrast, co-simulation approaches leverage the expertise of multiple domain-specific simulators, providing more accurate and detailed simulations of individual components. While this can lead to improved overall simulation results, co-simulation setups can be more complex to manage and may require additional effort to ensure interoperability between the individual simulators.

Our virtual testbed approach can be seen as a compromise between these extremes. It integrates a significant amount of functionality in VEROSIM, covering network and physics simulations, while

delegating specialized tasks to Qemu for hardware emulation. This approach maintains a high degree of integration and simplicity in the simulation setup and model management, while still leveraging the unique capabilities of specialized tools like Qemu for accurate hardware emulation.

## 4 METHOD

The virtual testbed is composed of two components: A multi-domain simulation framework, VEROSIM, that acts as the main user interface and is responsible for network and physics simulation, and the virtualization platform Qemu that executes real software on emulated target hardware. To the user, the involvement of the virtualization platform is mostly transparent, as the entire simulation scenario including virtual machines, emulated network devices (see Section 4.2.2), serial interfaces (see Section 4.2.3) and their connections with other simulated components is modeled in VEROSIM's unified modeling language (Reitz et al. 2023). The virtual machines and related configurations are set up and torn down automatically, as specified in the model. While the network connections are used to establish connections between different virtual machines via the network simulation, the serial interfaces are used to communicate with the physics part of the simulation, by transmitting sensor readings or actuator commands. Data exchange between VEROSIM and Qemu is realized by different means, depending on the specific interface being simulated.

In the remainder of this section, we first introduce the concept of hybrid cables to abstract over these differing technical implementations. We then discuss details of the hardware and device emulation, network UART simulation, as well as their integration in the virtual testbed.

### 4.1 Hybrid cables

To ensure seamless development, testing, and verification, it is crucial to bridge the gap between simulations and real-world scenarios. To address this challenge, we introduce the concept of "hybrid cables" in our virtual testbed. Hybrid cables serve as a conduit between simulated components in the virtual environment and real-world hardware devices or networks, enabling a smooth transition between the simulated and the physical worlds.

Hybrid cables consist of two distinct ends, with one end connected to the model simulated components within the virtual testbed, and the other end interfacing with either real hardware devices or an actual network. This connection is made possible by using suitable adapters, which can be implemented in software or hardware, depending on the application requirements.

In the case of software adapters, hybrid cables can be connected to tap devices, as demonstrated in our case study. Tap devices provide a software-based solution that allows for the creation of virtual network interfaces, enabling the transmission of network packets between the simulated environment and the real world. For hardware adapters, hybrid cables can be connected to various types of interface converters, such as Controller Area Network (CAN) to USB adapters. These adapters allow for the translation of communication protocols between the simulated components and the physical devices, ensuring accurate data exchange and interoperability. This approach facilitates the seamless integration of simulated and real components, allowing engineers to verify system behavior in realistic conditions.

While the hybrid cables approach offers numerous benefits, it also imposes additional requirements on the simulated model components. These components need to be able to handle real network packets, which makes the simulation more realistic but also requires more development work. This added complexity is offset by the insights gained from evaluating the system in real-world conditions.

### 4.2 Hardware Emulation

Integrating hardware emulation in the virtual testbed allows the inclusion of virtual machines running real software in the simulation scenario. Having real software in the loop means that the effects of every implementation detail can be observed in the simulation. This is a major benefit compared to using more abstract software models, such as state machines or petri nets.

This integration requires the execution of emulated hardware to be synchronized with simulation time and a means to exchange data between the simulators. Methods for the synchronization of both simulators and the data exchange between emulated peripherals and their counterparts in the physics simulation have been investigated in prior work (Reitz et al. 2020).

The main improvements compared to this prior work is the integration of emulated network devices with the newly developed network simulation in VEROSIM and the new integration of emulated serial interfaces and their integration with the physics simulation using the hybrid cable concept. This allows a richer, more flexible integration. In both cases, Qemu's device emulation is used, which will be introduced in the next section.

### 4.2.1 Device Emulation

In most computing hardware, access to peripheral devices like disks, network cards or serial interfaces is mediated by special memory addresses. Writing to and reading from these addresses has particular side effects, like sending messages via a serial interface, initiating a read from disk or receiving messages from a network card. The emulator must reproduce these side effects to maintain the guest's illusion of running on real hardware.

Qemu separates the task of device emulation into two components, a device frontend and a device backend. Device frontends emulate the behavior of specific hardware devices, like a concrete network card model, from the guest's point of view. This means monitoring the special memory addresses for writes and reacting appropriately, as well as writing to other special memory addresses. Device backends offer different mechanisms to realize the expected side effects. In the case of disk emulation, the guest expects her writes to be persistent. This may be achieved by storing data in the host file system, assigning a partition of the host system's disk to the virtual machine or sending the data to remote storage over the network. Separation of device frontend and backend allows free combination of device presentation to the guest and implementation of the corresponding effects on the host.

Two device categories are particularly relevant to the simulation of cyber-physical systems: Network hardware and serial interfaces. Network hardware mediates the communication between computing nodes, and serial interfaces are typically used to interact with sensors and actuators. Available options for the relevant device backends and their integration into the virtual testbed will be discussed in the following.

### 4.2.2 Network Devices

Qemu offers a variety of network device backends for different use cases. They represent different options to transmit network messages sent from the guest to their destination and to pass received network messages back to the guest. Figure 3 depicts two common device backends: Virtual networks and unix domain sockets.

To connect two virtual machines via a virtual network, each emulated network card is attached to a tunnel adapter (TAP) device, which is in turn connected to a bridge device. This forms a virtual Ethernet connection between the virtual machines that is mediated by the host networking stack. The virtual network can be extended to include more virtual endpoints and real endpoints to grant access to the host network. The second option is to connect both machines using a unix domain socket. This does not require the setup of virtual network devices, but is limited to one-to-one connections. Section 4.3 details the coupling with the simulation frameworks network simulation, and Section 4.3.2 discusses the performance characteristics of both options and the effect of inserting a simulated network in the connection.

### 4.2.3 Serial Interfaces

Next to the integration of real software via this network interface, virtual machines can also be equipped with serial interfaces, like UART or I2C. There is a variety of character device backends available, like

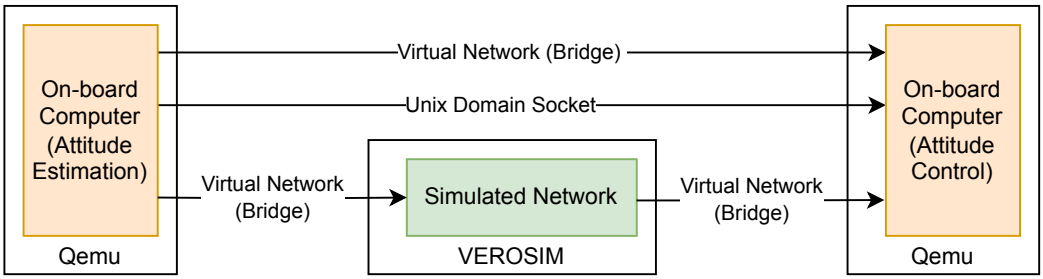


Figure 3: Three options to connect the virtualized on-board computers: Unix domain sockets, a virtual network, and a virtual network with an integrated network simulation.

`pipe`, `stdio` or `UDP`. A pragmatic choice for data exchange with the simulation framework are unix domain sockets, as they incur little overhead and are easy to work with.

### 4.3 Network Simulation

In this section, we discuss our approach to network simulation within the multi-domain simulation framework VEROSIM. Integrating network simulation into a 3D framework has the benefit of granting access to scene geometry and material properties. First, this enables a natural treatment of mobility by effectively tracking and modeling dynamic changes in network topologies due to movement. Second, the 3D framework allows the use of sophisticated physical radio wave propagation models that consider scene geometry and material properties, leading to a more accurate representation of signal propagation and interference, as highlighted by (Alfrink, Reitz, and Roßmann 2021). Additionally, having network and physics aspects within the same software and model simplifies the simulation setup and model management, streamlining the development and testing process.

In the following sections, we detail our network simulation implementation in VEROSIM and demonstrate how its integration with physics and hardware emulation contributes to the effective simulation and verification of overall system behavior in cyber-physical systems, including the simulation of fault scenarios.

#### 4.3.1 Network Emulation

To allow bridging the simulated network with real networks, a gateway to the host network has been implemented utilizing a tunnel adapter (TAP) device. This technique is commonly used by other network simulators, such as the OMNeT++ INET framework (Mészáros et al. 2019) or ns-3 (Fontes et al. 2015), to test simulated scenarios with real-world traffic. It is also similar to the mechanism Qemu employs to attach emulated network devices to the host network, as outlined in Section 4.2.2. An important use case of the feature regarding cyber-physical system simulation is the connection of virtual machines via a simulated network. Compared to direct connections of virtual machines as outlined in Section 4.2.2, this allows arbitrary network topologies, fine-grained control over the network's transmission characteristics, such as latency and throughput, as well as the injection of arbitrary faults. Figure 3 depicts the connection of two virtualized on-board computers via virtual networks with an integrated simulated network. The simulated network contains two TAP adapters that are attached to two separate TAP devices of the virtual network. Inside the simulation model, a simulated Ethernet cable connects both TAP adapters. The simulated Ethernet cable has configurable transmission properties such as latency and packet loss rate. The network topology is kept simple for demonstration purposes. However, it is possible to add Ethernet switches and additional cables. The feature set of the network simulation is small compared to other established network simulators, but includes basic implementations of Ethernet, ARP, IP and UDP, that are compatible with real networks. This means a fully simulated computation node, for example realized via VEROSIMs Python scripting interface, can communicate with other nodes on the simulated or real network.

### 4.3.2 Network Performance Evaluation

A network benchmark considering both latency and throughput shows the overhead created by guiding the network traffic through the network simulation compared to using Qemus standard network backends. The benchmark includes all three options to connect the virtual on-board computers depicted in Figure 3: Standard Qemu network connections using both virtual networking devices and unix domain sockets as network backend (Section 4.2.2), and a configuration where the network simulation is inserted into the virtual network. Two test cases are considered for the configuration with the simulated network. In one test case, the overhead of the simulated network is isolated by configuring ideal transmission properties in the simulated Ethernet cable with 0 ms latency and 0 % packet loss. A second test case shows the ability to influence transmission characteristics via configuration of the simulated network by observing the effects of 5 ms latency and 20 % packet loss in the simulated Ethernet cable.

Latency was evaluated by executing `ping` inside one of the virtual machines with the other's IP address. The reported results depicted in the upper part of Figure 4 are one way latencies, i.e. half of the round trip times reported by `ping`. Unix domain sockets offer the lowest latency, followed by the virtual network. Inserting the VEROSIM network simulation into the virtual network connecting the virtualized on-board computers causes a slight, consistent increases in latency compared to the configuration with only the virtual network. This suggests that the latency is not caused by the virtual network, but has a different origin. In any case, VEROSIM does not lead to a significant increase in latency, when the simulated Ethernet connection is parameterized with a latency of 0 ms. If the simulated Ethernet connection is parameterized to 5 ms latency, the measured latency increases accordingly.

The program `iperf3` was used to measure network throughput. It was launched in server mode on one virtualized on-board computer and in client mode on the other. The client attempts to send UDP packets at 2 Gbit/s, while the server measures the incoming packets.

The results in the lower part of Figure 4 show that the configuration with the virtual network achieves the highest throughput. Connecting two virtual machines with a domain socket leads to roughly 20 % less throughput. Inserting the simulated network with ideal transmission characteristics again has only a small effect on the measured network performance. Setting the packet loss rate to 20 % leads to a corresponding decline in throughput.

These results show, that inserting the network simulation in the test configuration leads to a minor increase in latency and a minor decrease in throughput, while giving fine-grained control over the characteristics of the simulated network. During the benchmark, the user interface was still completely responsive and the rendering smooth. One caveat is, that the simulation was running a fairly simple physics model during this benchmark and was thus able to run in real time. When simulating more complex models, such as large dynamics models or expensive sensor simulations, the results may vary.

## 4.4 UART Simulation

A simulation of the universal asynchronous receiver transmitter (UART) protocol is added to capture the effects of serial communication characteristics on the overall system behavior. UART is an asynchronous serial protocol, which only uses one wire for communication with the target device. The communication channel is characterized by the speed at which bits are transmitted (baud rate), as well as the number of data bits, parity bits, and stop bits used.

Most UART simulations model serial communication on the logic level, transmitting and sampling each bit individually using the configured baud rate (Norhuzaimin and Maimun 2005). In the context of cyber-physical systems simulation, however, the level of detail provided by bitwise simulation is usually not required. For this reason, a simplified model of serial communication is implemented that simulates throughput and buffer sizes. The characteristics of sender and receiver, such as baud rate, data bits, etc., are configurable, and data transmission only succeeds if they are compatible. Adding data to the transmitter's send buffer activates the transmission of this data in blocks of data of configurable size. The time required

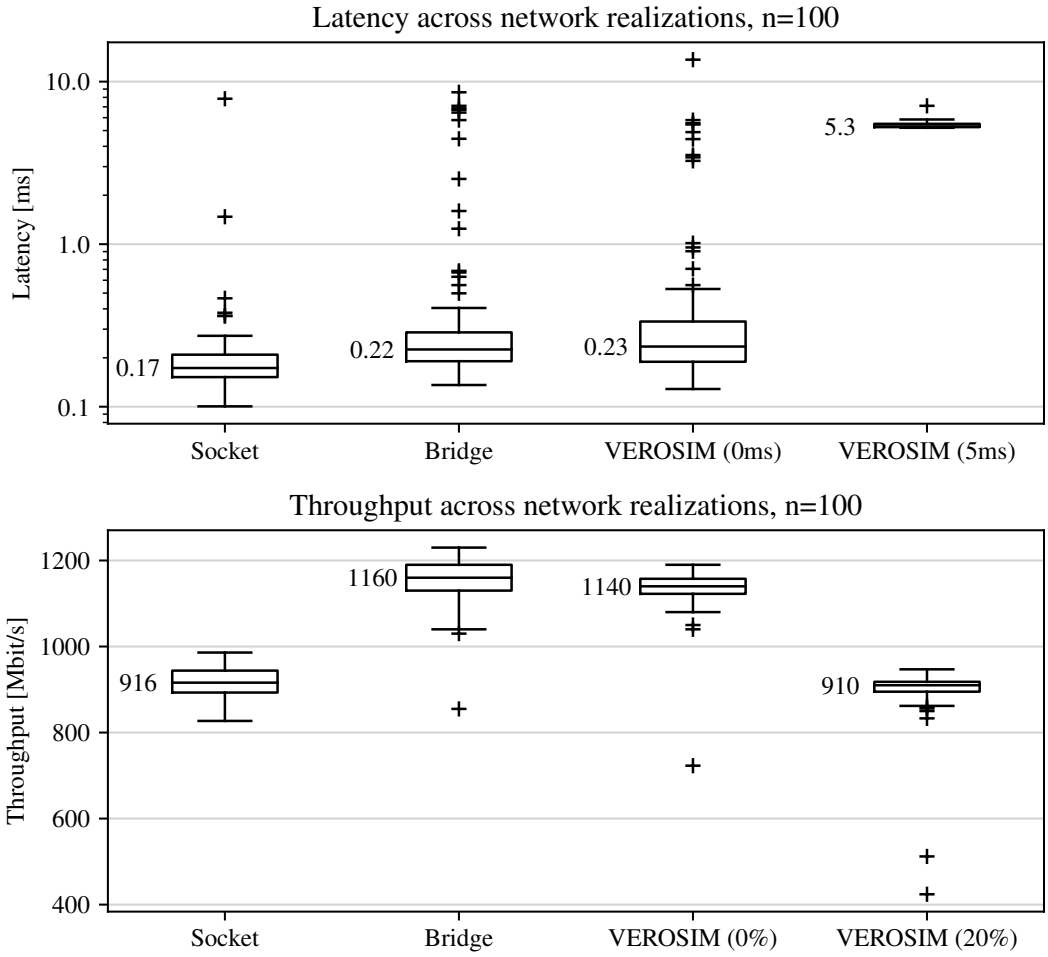


Figure 4: Network performance benchmark comparing latency and throughput of direct connections between virtual machines with different network emulation backends and connections via the network simulation with different simulated connection properties.

for the transmission of each block is calculated based on the connection's baud rate. At the destination, the data is inserted into a receive buffer. From there, it can either be read and processed within the simulation or be forwarded to a virtual machine, for example using a character device as described in Section 4.2.3. If the size limit of either buffer is exceeded, no more data is written to the buffer until there is space again.

This approach allows the use of serial interfaces in the virtual testbed. It captures the necessary effects to test if the system under test is limited by bandwidth or buffer sizes involved in serial communication, while being computationally inexpensive.

## 5 APPLICATION

The presented method is applied to an approach scenario of a capsule to the International Space Station (ISS), as described in Section 2. The architecture of the distributed attitude control, as depicted in Figure 2, is maintained in the simulation model. Figure 5 shows a schematic view of the simulation model, including the components required to facilitate data exchange between the simulators. Emulated network interfaces in Qemu are attached to the simulated network in VEROSIM via TAP adapters and virtual networking devices, as discussed in Section 4.3.1. The connection between the emulated serial interfaces (Section 4.2.3) and

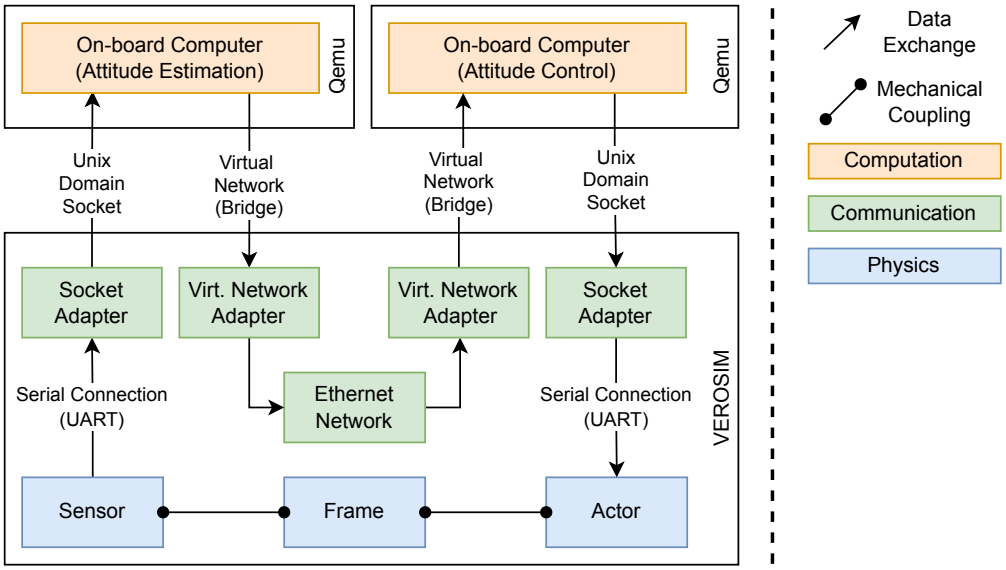


Figure 5: Setup of the simulation scenario including model components in the simulation tool VEROSIM, virtual networking devices and virtual machines.

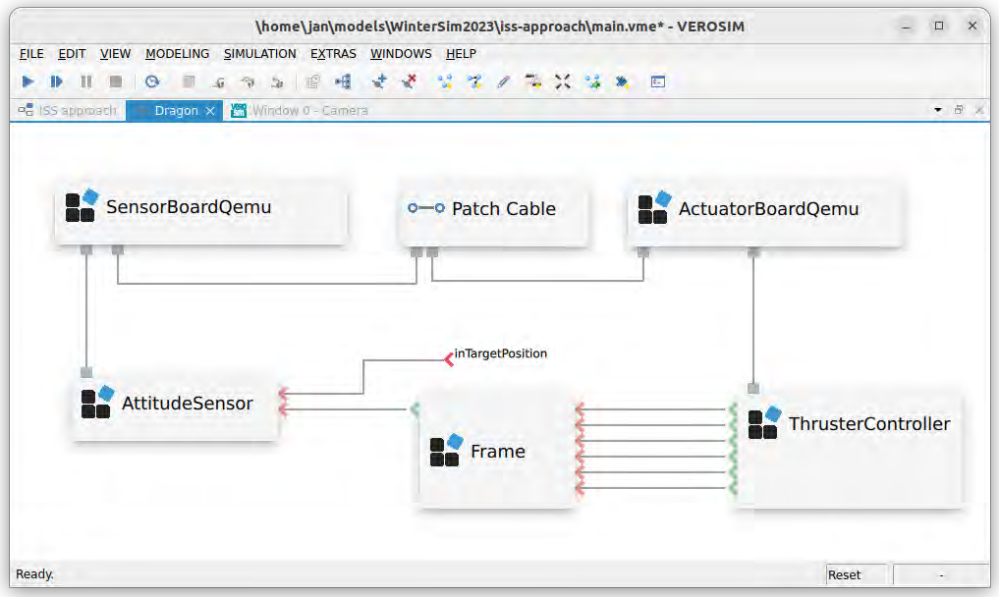


Figure 6: Block diagram view of the capsule model in VEROSIM.

the simulated sensor and thrusters via the UART simulation (Section 4.4) are realized using unix domain sockets.

Figure 6 is a VEROSIM screenshot of the block diagram view of this model. The structure of the system of interest is preserved, as all components and their connections are represented in a one-to-one relation to their real counterparts.

The development of this application began with prototyping the attitude estimation and controller software using VEROSIM's Python scripting interface and direct data exchange within the simulation. After creating a functioning prototype, the software was ported to C++ to be executed in virtual machines

or on the actual OBCs. During this translation from simple prototype scripts with access to all properties of the simulation to a completely separate C++ program, many issues related to data rates, buffering, serialization and other networking topics arose, both regarding the network connection and the serial connection. This is a feature, not a bug, since it is preferable to encounter these issues earlier in the development cycle. Furthermore, the joint simulation allows an iterative translation process, replacing prototypes with real components as development progresses.

## 6 EVALUATION

Figure 7 shows the capsule trajectory under different networking conditions. The blue trajectory represents a baseline of ideal networking conditions with no latency added by the network simulation and all packets being forwarded. With the combination of thrust limitations and the chosen control parameters, the capsule oscillates around the target trajectory and closes in on the rendezvous point.

The system's high inertia means that increased latency does not have a large effect on the resulting trajectory. Even a latency of 500 ms is barely noticeable. A similar assumption was held for packet loss. Since the connectionless and unreliable UDP protocol is used, packet loss should only lead to larger intervals between control values, which are not critical in this application, due to the capsules high inertia. The actual results of the trajectory with a 90 % packet loss rate are puzzling and led to the questioning of everything from various implementation details to the random number generator used to calculate the packet dropping. The real explanation, however, lies in the virtual machines Linux networking stack sending periodical Address Resolution Protocol (ARP) requests for IP addresses it has already resolved. If the sender does not receive a response, it will stop sending UDP packets containing the estimated attitude value until a valid ARP reply is received. The low probability of two sequential successful transmissions: $(1-0.9)^2 = 0.01$, combined with an ARP polling rate of $1\,\mathrm{s}^{-1}$ leads to long stretches without communication. To verify this explanation, another trajectory with packet loss rate 90 % is depicted, where ARP packets were never dropped. This trajectory is in line with initial expectations. While frustrating, observing such unexpected behavior in the safe confines of a simulation during development is invaluable.

Figure 8 depicts the same scenario with two phases of simulated connection loss lasting 10 s visualized in gray. During the connection losses, the thrusters are firing at a constant rate since no new set points are received, causing the capsule to deviate from the target trajectory. There is enough time for the controller to

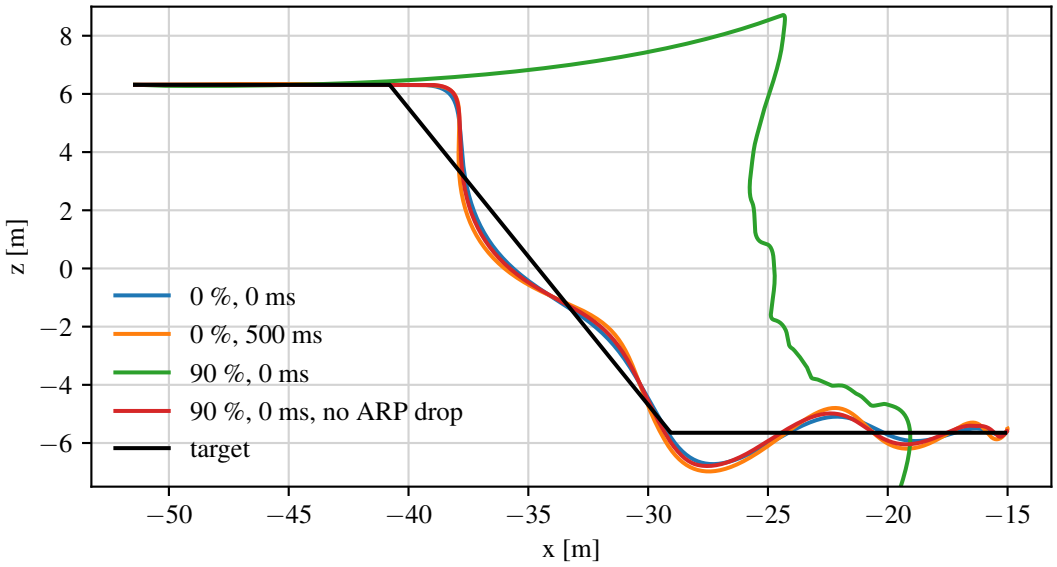


Figure 7: Approach trajectory under different network conditions. (packet loss % | latency ms)

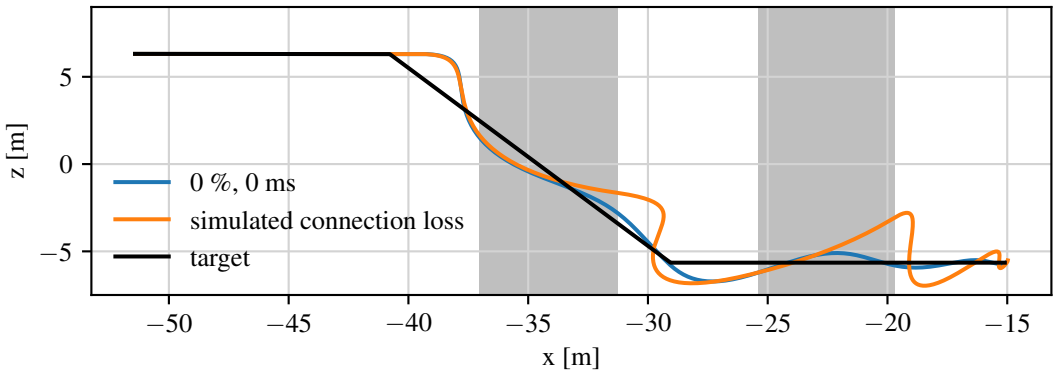


Figure 8: Approach trajectory with simulated connection loss.

compensate for the deviation after connection is reestablished and to return to the target trajectory. However, assuming a linear relationship between cumulative thruster force and fuel consumption, the detour incurs a 242 % increase in fuel consumption.

This application shows the impact of low-level details on the overall system behavior. In this case, the configuration of the ARP implementation can cause a manifold increase in fuel consumption and even catastrophic mission failure. The employed virtual testbed approach has proved to be a useful tool for the identification of issues that span across multiple disciplines and the verification of overall system behavior.

## 7 CONCLUSION

The main contribution of this paper is the integration of 3D physics simulation, hardware emulation and network simulation in a virtual testbed for cyber-physical systems. The technical realization consists of the multi-domain simulation framework VEROSIM and the virtualization platform Qemu. Data exchange between these software components is realized using virtual network devices and unix domain sockets. A network performance benchmark shows that guiding network traffic between two virtual machine instances through the VEROSIM network simulation incurs only slight overhead but gives fine-grained control over the transmission characteristics. The virtual testbed was applied to an ISS approach scenario. An analysis of the distributed trajectory controller of the approaching capsule considering nominal and adverse network conditions in the form of increased latency, packet loss rates and total communication loss revealed an unexpected issue of long stretches without communication under high packet loss conditions due to a detail of the Linux implementation of the address resolution protocol that could cause catastrophic outcomes. This result shows the impact of low-level implementation details on the overall system behavior and highlights the importance of integrated simulation approaches. To this end, the presented virtual testbed is a useful tool for the identification of such issues and, more broadly, for the verification of the overall system behavior throughout the life cycle of the cyber-physical system. Currently, the complexity of the network and physics models is limited to what the virtual testbed can simulate in real time. Future work includes performant synchronization between Qemu and VEROSIM to overcome this limitation.

## REFERENCES

Alfrink, M., J. Reitz, and J. Roßmann. 2021. “Improving Ray Tracing Based Radio Propagation Model Performance Using Spatial Acceleration Structures”. In *Proceedings of the 17th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, Q2SWinet ’21, 25–32. New York, NY, USA: Association for Computing Machinery: Association for Computing Machinery.

Bellard, F. 2005. “QEMU, a Fast and Portable Dynamic Translator”. In *Proceedings of the USENIX Annual Technical Conference*, 41–46. Anaheim, California: USENIX Association.

Blochwitz, T., M. Otter, J. Akesson, M. Arnold, C. Clauß, H. Elmqvist, M. Friedrich, A. Junghanns, J. Mauss, D. Neumerkel, H. Olsson, and A. Viel. 2012. "Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models". In *Proceedings of the 9th International MODELICA Conference*, edited by M. Otter and D. Zimmer, 173–184. Modelica Association: Linköping University Electronic Press.

Eker, J., J. W. Janneck, E. A. Lee, J. Liu, X. Liu, J. Ludvig, S. Neuendorffer, S. Sachs, and Y. Xiong. 2003. "Taming Heterogeneity - The Ptolemy Approach". *Proceedings of the IEEE* 91(1):127–143.

Fontes, H., R. Campos, and M. Ricardo. 2015. "Improving ns-3 emulation support in real-world networking scenarios". In *SIMUTOOLS 2015 - 8th EAI International Conference on Simulation Tools and Techniques*. European Alliance for Innovation: ICST.

Gomes, C., C. Thule, D. Broman, P. G. Larsen, and H. Vangheluwe. 2018. "Co-Simulation: A Survey". *ACM Computing Surveys* 51(3):1–33.

Mészáros, L., A. Varga, and M. Kirsche. 2019. *INET Framework*, 55–106. Springer Cham.

Norhuzaimin, J., and H. Maimun. 2005. "The Design of High Speed UART". In *2005 Asia-Pacific Conference on Applied Electromagnetics*, 5 pp.–. IEEE.

Reitz, J., U. Dahmen, T. Osterloh, and J. Rossmann. 2023. "Systems Engineering and Simulation: Towards a Unified Methodology for Developing Cyber-Physical Systems". In *SysCon 2023 - 17th Annual IEEE International Systems Conference, Proceedings*. IEEE: Institute of Electrical and Electronics Engineers Inc.

Reitz, J., A. Gugenheimer, and J. Rossmann. 2020, 12. "Virtual Hardware in the Loop: Hybrid Simulation of Dynamic Systems with a Virtualization Platform". *Proceedings - Winter Simulation Conference* 2020-Decem:1027–1038.

Rossmann, J., M. Schluse, and R. Waspe. 2013. "Combining Supervisory Control, Object-oriented Petri-Nets and 3D Simulation for Hybrid Simulation Systems using a Flexible Meta Data Approach". In *Proceedings of the 3rd International Conference on Simulation and Modeling Methodologies, Technologies and Applications*, 15–23. INSTICC: SciTePress - Science and and Technology Publications.

## AUTHOR BIOGRAPHIES

**JAN REITZ** is a researcher at the Institute for Man-Machine Interaction at RWTH Aachen University. He holds a B.Sc. and M.Sc. in Mechanical and Process Engineering from TU Darmstadt. His research interest is in modeling and simulation of cyber-physical system, and distributed systems. His email address is reitz@mmi.rwth-aachen.de.

**DAVID BÖKEN** is a researcher at the Institute for Man-Machine Interaction at RWTH Aachen University. He holds a B.Sc. and M.Sc. in Electrical Engineering, Information Technology and Computer Engineering from RWTH Aachen University. His research focus is on the human interaction with distributed system in the context of dataspaces and IoT. His email address is boeken@mmi.rwth-aachen.de.

**JÜRGEN ROßMANN** is a professor with the faculty of electrical engineering at RWTH Aachen University and head of the Institute for Man-Machine Interaction. After graduating from the Universities of Dortmund and Bochum, he worked as a research assistant, then as a group leader and finally as a department head at the Institute of Robotics Research at the University of Dortmund. His work in the fields of space and industrial robotics, automation technology and virtual reality has been awarded more than 15 different prizes. He has been a visiting professor for robotics and computer graphics at the University of Southern California since 1998. Furthermore, he is a member of the board of Directors of the Dortmund Institute for Research and Transfer (RIF) as well as managing director of VEROSIM GmbH. His email address is rossmann@mmi.rwth-aachen.de.