Secure Distributed Co-Simulation over Wide Area Networks

Kristoffer Norling[1], David Broman[1], Peter Fritzson[1], Alexander Siemers[2], Dag Fritzson[2]

[1)] Department of Computer and Information Science, Linköping University
SE-581 83 Linköping, Sweden
{x06krino,davbr,petfr}@ida.liu.se

[2)] SKF Engineering Research Centre
MDC, RKs-2, SE-415 50 Göteborg, Sweden
alexander.siemers@skf.com
dag.fritzson@skf.com

Abstract

Modeling and simulation often require different tools for specialized purposes, which increase the motivation to use co-simulation. Since physical models often are describing enterprises' primary know-how, there is a need for a sound approach to securely perform modeling and simulation. This paper discusses different possibilities from a security perspective, with focus on secure distributed co-simulation over wide area networks (WANs), using transmission line modeling (TLM). An approach is outlined and performance is evaluated both in a simulated WAN environment, and for a real encrypted co-simulation between Sweden and Australia. It is concluded that several parameters affect the total simulation time, where especially the network delay (latency) has a significant impact.

Keywords: Modeling, Co-Simulation, Transmission Line Modeling, Security, Data communication.

1. Introduction

The interest for modeling and simulation of complex physical systems, such as aircrafts and trains, has dramatically increased during the last decades. There exist both commercial modeling and simulation environments for specific domains, such as Adams [9] for mechanical systems, and specialized environments for certain application areas, e.g., SKF's BEAST [14,15] dedicated for bearing simulations. Moreover, new standardized acausal modeling languages for mixed-domains, e.g., Modelica [8] and VHDL-AMS [4], are gaining popularity in various applications, leading to a wide variety of models and specialized tool environments within an enterprise. These models and components are often dependent on each other and commonly need to be simulated together.

One technique that has proven to be both stable and efficient for simulating existing models is *co-simulation*, where sub-models are coupled together using *transmission line modeling (TLM)* [3,5,6,10]. The technique makes use of physically motivated time delays, enabling both numerically robust simulations and better performance using parallel processing.

Since modeling and simulation is becoming a common activity within enterprises, large amount of money and knowledge are invested into such models. Hence, models are becoming critical business assets describing primary know-how.

Within larger enterprises, different departments can be spread out over of the world, have special modeling competences, and model different parts of systems. Usually, there are defined confidentiality levels within an organization, requiring models to be protected against unauthorized disclosure. Furthermore, models need to be protected from modification by mistake and still being available for large scale reuse. Even if confidentiality issues with sub-contractors are today normally regulated by contracts, the need to protect and keep control of critical business assets is still vital.

Hence, in a co-simulation environment, it is important to have a sound approach for *secure modeling and simulation*, i.e., to create, modify, distribute, and simulate models in an acceptable manner according to the enterprise's security policy.

1.1 Approaches to Secure Modeling and Simulation

Information security can be divided into three aspects[1]:

- *Confidentiality*: protection against unauthorized disclosure of information.
- *Integrity*: protection against unauthorized creation, modification, or deletion of information.
- *Availability*: the assurance that authorized entities have access to correct information when needed.

[1] Even if most practitioners in the field of information security recognize these three aspects as fundamental, there is no exact border between e.g., robustness and availability. Here we treat robustness of the system as being part of the availability of simulation and model information.

The importance of these three areas for secure modeling and simulation will be outlined in the following section.

The obvious first phase in modeling and co-simulation is the modeling phase, which can be divided into three steps, each requiring different expertise [10,12]:

- Modeling of sub-models in specialized environments, e.g., Adams and BEAST models.
- Encapsulate sub-models and define interfaces.
- Design a *meta-model*, where the different sub-models are integrated and connected to each other.

It is of great importance to perform these steps in a secure manner. However, in this paper we have chosen to focus on how the models are *securely distributed and simulated*, leaving secure modeling as future research.

We have identified two general possible approaches for secure distribution and simulation:

- *Centralized simulation with secure model exchange*. The models are centralized and simulated at one location. The models must be encrypted to avoid unauthorized disclosure of the model content.
- *Secure distributed co-simulation.* The models are kept within their departments/companies and simulated locally. Simulation data is exchanged between the locations during simulation, making use of co-simulation technologies.

Both these approaches raise several questions regarding security aspects of confidentiality, integrity and availability, as well as practical simulation feasibility.

The former centralized approach uses traditional simulation techniques, where all models are locally available. To provide confidentiality and avoid disclosure of model content, the models need to be encrypted. This sounds initially as a sound approach, but even if the model was encrypted when sent to the other locations, it still needs to be decrypted at the centralized location. This means that the simulation environment needs to know the encryption key. Hence, the information is only *practically confidential*, i.e., it might be hard to get the model in clear text, but not theoretically impossible[2].

A second alternative to model encryption can be *model obfuscating*, i.e., to rearrange the models structure making it unreadable, without changing the model's behavior. Another variant is to distribute the models in another format, e.g., as binary executables.

A certain level of integrity protection, e.g., to achieve need-to-know access (least privilege), can be used by applying message authentication codes (MAC) on the models. However, the same problematic issues regarding keys are unfortunately applicable here as well. Moreover, the centralized approach cannot control the number of times a model is used or that it is not illegally redistributed.

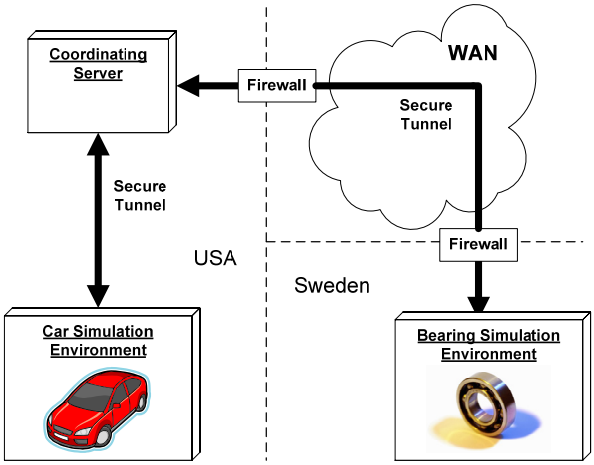

Figure 1. A scenario of secure distributed co-simulation distributed over long distances.

Figure 1 outlines a scenario of the second approach of distributed co-simulation, where a car model is simulated at a location in the USA and the bearing model is simulated in Sweden. To make this scenario possible, a co-simulation technique such as the one based on Transmission Line Modeling (TLM) can be used. Models are not sent between the different locations. Instead data describing states of the models' interfaces are exchanged between the locations during simulation, using a wide area network (WAN), e.g., the Internet. This scenario leads to several security observations:

- *Confidentiality*: The models are never exchanged between the parties and are therefore never exposed to the other party, i.e., confidentiality is kept using a black-box view of the model. If the TLM data sent between the simulation nodes is encrypted, unauthorized disclosure of the traffic is avoided from third parties. Furthermore, the permission to reuse models for simulation can be controlled, since the models never leave the original location.
- *Integrity:* Unauthorized modification or changes by mistake of models are avoided due to the fact that models are never sent between the locations. Note however that the meta-model needs to be defined and maintained at one location. Integrity protection of simulation traffic can be protected using standard MAC technique.

[2] In cryptography, it is never impossible to decrypt data without a key, but it can be *computationally very hard*. E.g., using a brute-force approach, it might take hundreds of years to decrypt a specific model.

- *Availability*. Co-simulation using TLM has been shown to provide numerically robust results. However, if the data communication link is not robust or is too slow, and data is not delivered in time, the simulation can be slowed down or terminated. Hence, performance, robustness, and total simulation time are critical factors concerning distributed co-simulation.

Both approaches have pros and cons. A centralized approach raises several problematic issues regarding true confidentiality and integrity of models. On the other hand, centralization can guarantee better communication networks and may therefore result in better performance.

The distributed approach is appealing due to the fact that the black-box characteristics give both better integrity and confidentiality properties of models. In this approach, computational resources from different locations can also be shared and used in a distributed manner. However, the extra overhead needed for protection of data sent during simulation and the cost of transmitting data traffic over large distances can naturally affect performance of the total simulation time.

1.2 Challenges and Contributions

We have chosen to focus on the second approach of *secure distributed co-simulation*, due to its promising properties of confidentiality and integrity. Hence, this paper will primarily deal with aspects of performance and robustness, with the following questions:

- Is it even practically possible to co-simulate stiff-bearing models in Sweden together with other mechanical models, which are simulated as far away as the USA or Australia?
- If it is possible, how much longer time does it take to simulate?
- Which parameters and factors affect the total simulation time? Which dependencies exist between these parameters? Are they linear or are there certain breakpoints?

The main contribution of this work is the described approach of secure distributed co-simulation together with conclusion draw from analysis of an experimental simulation test.

1.3 Paper Outline

The paper is structured as follows. Section 2 introduces the fundamental theory of Transmission Line Modeling (TLM) for co-simulation and the basic concepts of data communication. Parameters and factors that may affect the total simulation time are discussed. Section 3 outlines the experimental setup used for generating simulation data from a model reflecting the scenario described in the introduction. The physical model, simulation framework, deployment structure, and simulation tools are described. Section 4 presents the results from the experiment followed by analysis of how different parameters affect the duration of the total simulation time. Finally, section 5 states conclusions of the work.

2. Parameters Affecting the Total Simulation Time

In a distributed co-simulation environment, several parameters and factors can affect the total simulation time. First, we will consider the equations stating transmission line modeling. Second, different factors of the data communication link will be discussed.

2.1 Transmission Line Modeling

The theory of transmission line modeling (TLM) has evolved from the telegraph equations, which concern transmission of electrical signals over long wires.

The TLM method can also be used in other domains, such as mechanical systems where force and velocity are affecting a system. The main motivation to use TLM in such a system is that it describes physically relevant time delays. These delays allow different parts of the system, which are separated by TLM interfaces, to be simulated independently of each other's time steps. Hence, this technique enables the subsystems to be simulated in parallel. Moreover, since exchange of data is needed only between well-defined interfaces, it is also possible to co-simulate between different simulation environments and tools. The latter fact is used in SKF's co-simulation framework [10,12,13].

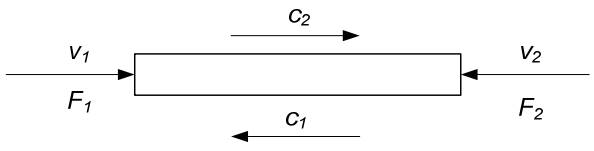

Figure 2. Delay line with wave variables c_1, c_2, velocity v_1, v_2, and reaction forces F_1, F_2.

Figure 2 outlines the main variables involved in a transmission line in a mechanical context. The velocity and reaction forces on each side of the line affect the wave variables c_1 and c_2.

The equations involved in the TLM connection are as follows [5,10]:

$$c_1(t) = F_2(t - T_{TLM}) + Z_c v_2(t - T_{TLM})$$
$$c_2(t) = F_1(t - T_{TLM}) + Z_c v_1(t - T_{TLM})$$
$$F_1(t) = Z_c v_1(t) + c_1(t)$$
$$F_2(t) = Z_c v_2(t) + c_2(t)$$

Here c_1 and c_2 denote the force waves, F_1 and F_2 reaction forces and v_1 and v_2 velocities. There are two parameters specified in the model: T_{TLM} indicating the delay time through the line and Z_c that is the characteristic impedance. These parameters must be assigned values, which are within physically acceptable boundaries. See [5,6,10] for a more detailed discussions about the physical aspects of the model.

From the equations above, we can clearly see that only old values of F_2 is needed to calculate F_1, and only old values of F_1 is needed to calculate F_2. The allowed length of this delay depends on the parameter T_{TLM}. Larger values of T_{TLM}, will allow the simulations to take larger time steps, which may result in shorter simulation time. Hence, T_{TLM} is an important parameter affecting the total simulation time.

2.2 Data communication

During the co-simulation a continuous flow of data packages containing TLM information are transmitted between the computation nodes. Before the data is transmitted over the WAN, it is being encrypted and message authentication codes (MAC) are created for integrity checks. When the data packet is received at the other node, it is being decrypted and forwarded to the simulation software.

There are two fundamental measurable characteristics of network performance [11].

- *Bandwidth* - the number of bits that can be transmitted over the network during a specific period of time. Normal measurement is in Bits/s.
- *Delay (or latency)* - the time period it takes for a very small message to be sent over the network. Delay is measured strictly in time, e.g., number of milliseconds.

A network always has these two characteristics, even though they can change over time, depending on routing and traffic load. This variation can lead to another important phenomenon called *jitter*, which is the variation of the delay for different network packets in a data stream. In the rest of the paper B_{WAN} denotes the bandwidth for the simulation environment and T_{WAN} the delay between the nodes.

Note that in this paper we will discuss two different time scales. The first one is the simulated time corresponding to the modeled physical process. Integration time steps, time stamps sent within the TLM framework, and the T_{TLM} parameter correspond to this time scale. The second time scale belongs to the real-time of the computation of the simulation result. The total simulation time and delays in the network, i.e., T_{WAN} belong to this scale.

When measuring delays in a system, the time is often referred to as the round-trip-time (RTT), which is the time it takes for the packet to travel to the destination and back. The delay of the system can be approximated to *RTT/2*.

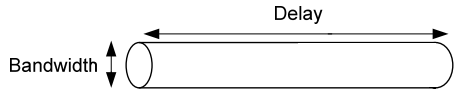

Figure 3. The delay × bandwidth product P.

Another central concept is *the delay × bandwidth product*, depicted in Figure 3. The formula is as follows:

$$P = B_{WAN} * T_{WAN}$$

The bandwidth can be seen as the diameter of a pipe, and the delay as the length of the pipe. Hence, the P represents the maximal natural queue of data located in the network.

Besides the TLM equations and the delay / bandwidth discussion above, there are of course several other factors that can affect the outcome. One such important factor is the load and balanced computation power among the nodes. Another factor is the integration step size / tolerance level used on the simulation nodes. Other potential requirements, such as encryption protocols and algorithms used, can both affect T_{WAN} and B_{WAN}. If the bandwidth is critical, lossless data compression techniques can be used to increase the bandwidth [1]. However, note that delay cannot be improved by data compression.

3. Experimental Setup

This section provides a detailed description of the environments surrounding the experiment, the simulation framework, and software used in the experiment.

3.1 Meta-Models and Components

A *component* is a model instance created in a specialized modeling and simulation tool (e.g., Adams or BEAST). A *meta-model* defines the interconnections between two or more components.

The objective of the experiment is to measure how secure communication over large distances will affect the meta-model simulation in terms of total simulation time. To model the scenario described in Figure 1, a highly simplified model is used instead of a real car model. We are interested in how the data communication factors, T_{WAN} (the delay) and B_{WAN} (the band-

width), together with the TLM delay T_{TLM} affect the simulation time.

The meta-model used in the experiment is a double pendulum. As shown in Figure 4, the pendulum is constructed from three different components, two shafts, which both have TLM connection to a bearing.

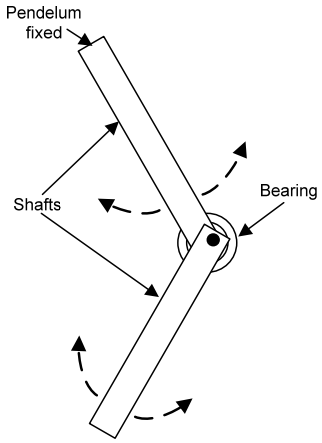

Figure 4. The bearing-shaft model to be simulated.

In terms of simulation time and resources, the two shaft components represent the less demanding components in our meta-model.

3.2 Simulation Framework

To conduct the experiment, we have used a centralized TLM co-simulation application referred to as the TLM manager. Figure 5 presents an overview of the TLM co-simulation framework.

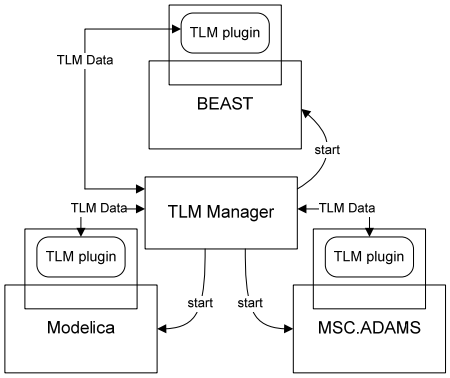

Figure 5. SKF's TLM system framework.

The TLM manager reads a meta-model specification defining the simulation components and the TLM interfaces in the meta-model.

The simulation components are simulated in a specialized simulation environment, such as Adams, BEAST, or Modelica. In order to make the framework functional, the specialized simulation environments need to be incorporated into the framework. The TLM manager and the specialized environment communicate through TLM plug-in's, as depicted in Figure 5.

During simulation, the simulated components interact by sending time-stamped data and delayed position and orientation data. The TLM data is transmitted to relevant node via the TLM manager. For a more detailed description of the framework, see [10].

3.3 Deployment Structure

The static deployment structure is depicted in Figure 6.

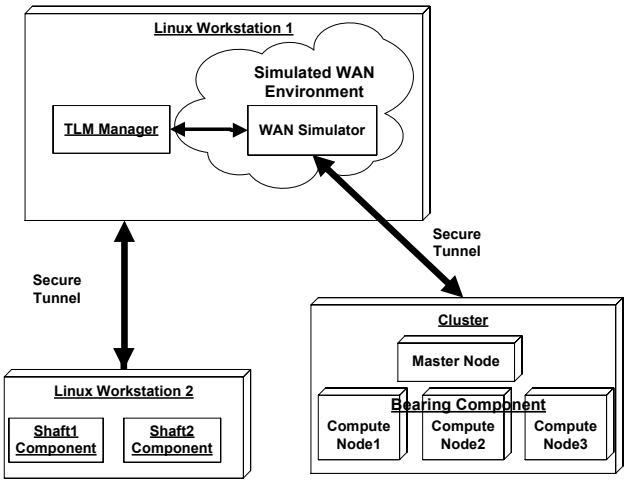

Figure 6. The deployment structure of the static structure of the system.

Two Linux workstations and a computer cluster are used for simulation of the components. The simulation is started from the workstation labeled Linux Workstation 1; this is also where the TLM-Manager and WAN simulator are instantiated.

In order to evaluate what impact delay and bandwidth in the WAN environment have on the simulation it is desirable to have full control of these parameters. This control is obtained through the WAN simulator. The WAN simulator is a software service that intercepts all data sent between the cluster and Linux Workstation 1. It will then apply the desired data communication factors and pass on the data to the correct destination. A more detailed overview of the WAN simulator will be given in the end of this section.

The shaft components are simulated using BEAST. These simulations are run on the workstation labeled Linux Workstation 2. The bearing component is simulated on the cluster using BEAST as well. The two tunnels are ensuring secure communication between the TLM manager and Linux Workstation 2 and between the TLM manager and the cluster. In the experimental setup SSH version 2 (SSHv2)[7] is used to create the secure tunnels. Another strategy would be to incorporate the security layer within the application. With such an approach TLS [2] may be a sound alterative.

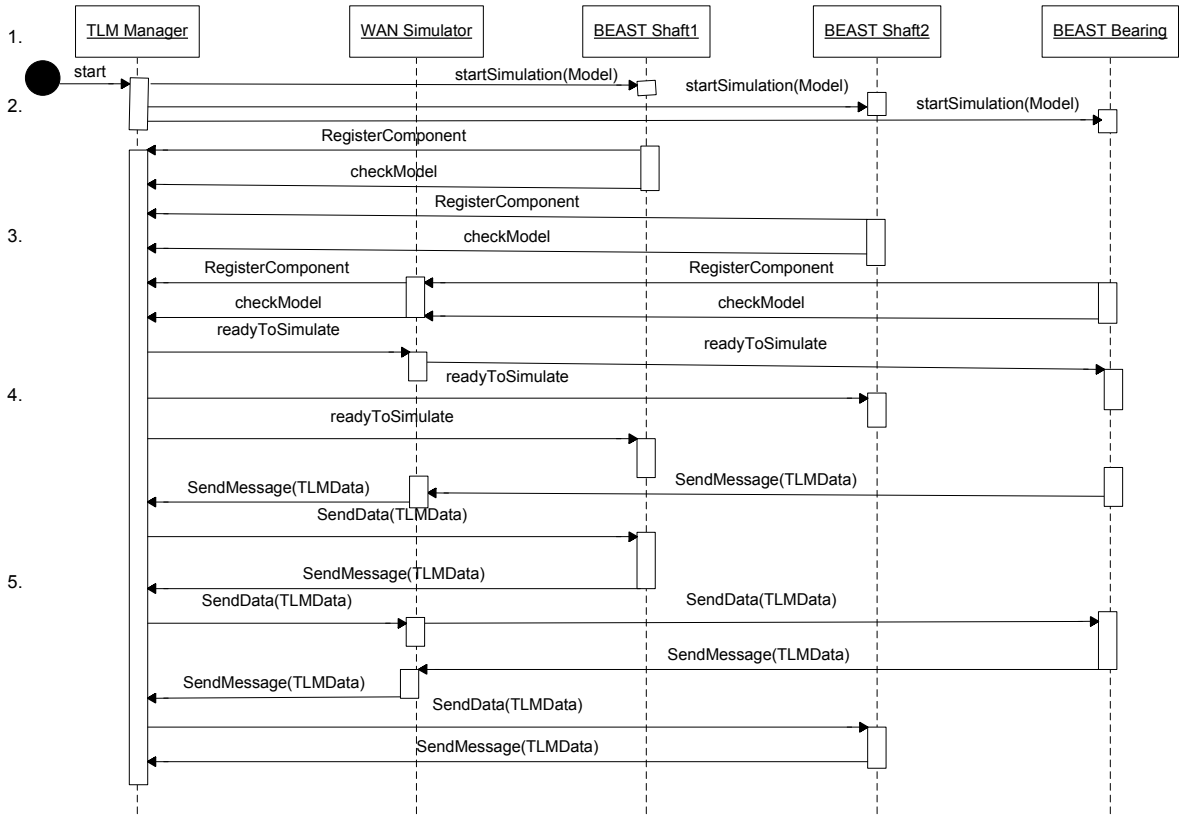

Figure 7. Sequence diagram showing basic information flow in the co-simulation environment.

3.4 Dynamic System Behavior

This section describes the interaction between the simulation components and the TLM manager. The sequence diagram in Figure 7 can be described as follows:

1. There are five different objects involved in the interaction during the simulation. To the right, there are the three simulation components, *BEAST Shaft1*, *BEAST Shaft2,* and *BEAST bearing*. The *TLM manager* is handling the data exchange between the components. Finally, the *WAN simulator* is intercepting and forwarding all calls between the bearing simulation component and the TLM manager.

2. The meta-model simulation is initiated by a script. The script will start the TLM manager on the local machine. It will also start the specialized simulation environments of all the simulation components.

3. Each simulation component has to register itself to the TLM manager. The component will also register all of its TLM interfaces. Once this has been accepted by the manager the simulation component will send a check model request to the manager. This is a request asking if the entire meta-model is ready to be simulated.

4. Once all simulation components and TLM interfaces are accounted for, the manager will reply to the simulation components that the meta-model is ready to be simulated. At this point the actual simulation will begin.

5. The simulation components will regularly send data for its TLM interfaces to the manager. The manager will then pass the data to the appropriate destinations. At the bottom of the diagram we can see how the bearing component through the TLM manager sends some TLM data directed to the shaft1 component. The WAN simulator intercepts this data, holds it for a while, and then forwards it to the TLM manager. The manager forwards the data to the shaft1 component that after some simulation time sends new data to the manager. This communication is not intercepted by the WAN simulator since the data is not passing through our simulated WAN environment. The manager will receive the new data from shaft1 and forward it to the bearing component. This message, from the manager to the bearing, will once again be intercepted by the WAN simulator, and then forwarded to the bearing component. This type of data exchange will continue until the simulation has finished.

For a more detailed specification of the interaction in the communication protocol, see [10].

3.5 WAN Simulator

The WAN simulator is a specific application designed for this experiment with the purpose of controlling the data communication (T_{WAN} and B_{WAN}) between the bearing simulation component and the TLM manager. The simulator captures all data sent between the TLM manager and the bearing component. The simulator consists of two queues, as depicted in Figure 8. One queue holds the data sent from the simulation component to the manager and one queue holds the data sent from the manager to the simulation component. The WAN simulator reads and writes data in the queues concurrently.

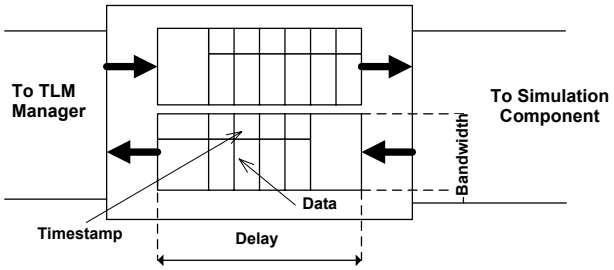

Figure 8. Conceptual outline of the WAN simulator.

To control the delay, the WAN simulator will, when capturing a data message, put a timestamp on it, before it is added to the queue. This data message will not be released from the queue until it has been held for the entire duration of the delay time.

For example, the simulation component sends some data to the TLM manager. The WAN simulator captures it and stamps it with the time of the capture. The message together with the timestamp is added to the last position in the queue of data waiting to be forwarded to the TLM manager. When the data reaches the first position in the queue it is next in line to be forwarded to the manager. The WAN simulator will then perform a check of the timestamp on the data. If the condition

$$T_{WAN} \leq (currentTime - timestamp)$$

is true, the data will be released from the queue and forwarded to the TLM manager. Else, the data will be held until enough time has passed to fulfill the condition. The second queue is working in the same way, but in the opposite direction.

Bandwidth may be controlled with the aid of the delay $\times$ bandwidth product, P, described in Section 2. The queues may hold up to P bits of data at any given time without risk of exceeding the desired bandwidth restrictions in respective communication direction. When a queue is filled, further data that wish to pass the WAN simulator will be delayed until data has been released from the queue and space has been made available.

4. Experiment Results and Analysis

In the following section we will present the experimental results produced by performing several test simulations in the experimental setup. The result is followed up by discussions and analysis of how different parameters affect the overall simulation time.

4.1 Experiment Results

The experiment was divided into two parts, where the first part used the experimental setup with WAN simulator according to Section 3. In the second part, co-simulation was performed over the Internet between Sweden and Australia.

Part 1 of the experiment was implemented by performing four different test sequences, where the data communication delay T_{WAN} was varied for each sequence. Figure 9 shows the four sequences for different values of T_{TLM}.

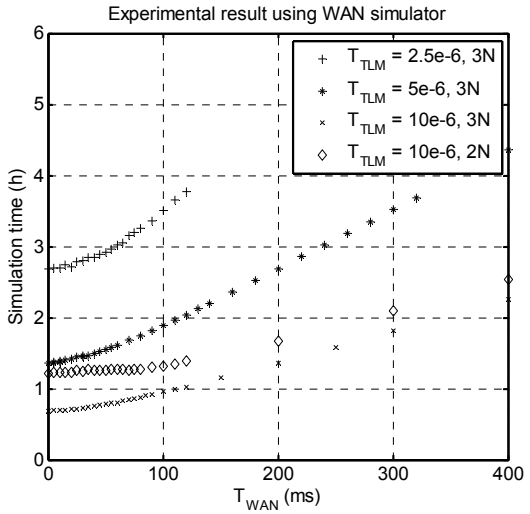

Figure 9. Plot of total simulation time in relation to the network delay T_{WAN}.

In the first three sequences, 3 computation nodes in the cluster (see Figure 6) were used, while the fourth sequence used only 2 nodes (denoted 3N and 2N in the figure). In addition to the values shown in this diagram, simulations were performed for T_{WAN} = 500 ms and T_{WAN} = 1000 ms for some of the sequences. These values are not shown in the diagram due to illustration reasons, but will be used in the analysis in the next section.

Part 2 of the experiment was performed without the WAN simulator, i.e., the same setup as depicted in Figure 6 was used, except that the TLM manager communicates directly to the cluster via an encrypted tunnel. Three computation nodes were used in the cluster for these experiments. The results are given in Table 1.

Type of simulation	T_{TLM} = 2.5e-6	T_{TLM} = 5e-6	T_{TLM} = 10e-6
LAN, No Encryption	120 min	62 min	31 min
LAN, Encryption	160 min	82 min	41 min
Increased time due to encryption	33 %	32 %	32 %
Round-trip simulation to University of Queensland, Australia	495 min	277 min	133 min
Increased time due to delays to Australia (round-trip)	312 %	344 %	332 %

Table 1. Simulation time when executed at the local area network (LAN) in Sweden, both with and without SSH tunnel encryption enabled. Round-trip simulation to and from Australia, including SSH encryption.

The table shows the increased time needed in percent using an encryption tunnel (SSH) compared to performing the simulation without it.

When executing the simulation via Australia, both the shafts and the bearing were simulated in Sweden, but the traffic between the cluster and the TLM manager was transmitted via a tunnel from Sweden to Australia and then back again. Hence, the experiment is actually performed for a distance twice as long as Australia-Sweden. Note also that the simulation time for just one way cannot be calculated simply by dividing in half.

While performing the simulations, the data throughput in the WAN simulator was being monitored and measured. Since the WAN simulator captures all data between the TLM manager and the bearing simulation component, it was measured that the data throughput did not normally exceed 35 Kbit/s during the TLM data exchange. However, when the simulations were initiated, a small peak in throughput of about 140 Kbit/s was noticed.

In the simulations performed in this experiment, no major robustness problems were discovered and all started simulation jobs were finished without unexpected termination.

All in all, the simulation experiment results presented in this section required approximately 250 hours of simulation time.

4.2 Discussion and Analysis

The main purpose of this work is to investigate if it is possible to co-simulate over long distances and which parameters that affect the total simulation time. In Table 1 it was shown that the simulation was indeed possible to perform, even for very long distances. We will now analyze and discuss the affected parameters and then finally compare the experimental data obtained using the WAN simulator with the simulation times resulting from the simulation via Australia.

Recall the diagram given in Figure 9 in the previous section. For small values of T_{WAN}, a non-linear behavior can be seen, while for larger values the curves have a more linear characteristic.

Consider Figure 10, where the values are plotted for T_{TLM} = 5e-6 and T_{TLM} = 10e-6, where $T_{WAN} \leq 250$ ms .

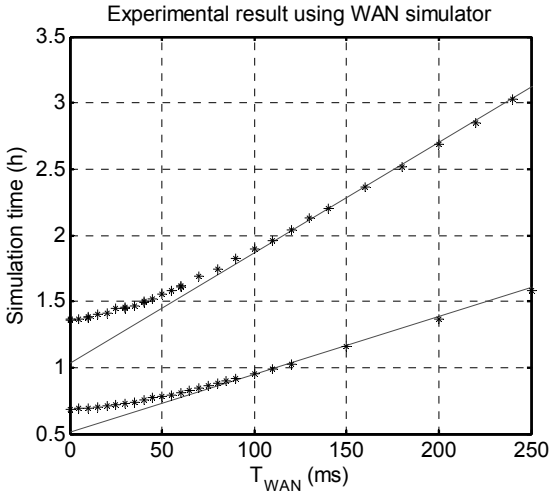

Figure 10. Simulation time with linear regression estimations. T_{TLM} = 5e-6 (the upper curve). T_{TLM} = 10e-6 (the lower curve). Both sequences are using 3 cluster nodes.

Linear regression using the least square method is used to adjust a line for $T_{WAN} \geq 80$ms. Both measured values and the adjusted curves are plotted in the diagram. Note that all measured values above 80ms are used for finding the curve, including measurements 500 ms and 1000 ms.

In both curves, we can imagine a smooth breakpoint somewhere around 75ms, where the derivative becomes fixed. What can cause this breakpoint? One reasonable cause is that to the right of the breakpoint, one of the computation nodes is idle for some time while waiting on TLM data input, i.e., the delay causes the bottleneck of the system to be switched from one of the computation nodes to the delay value in the communication link. For this reason, we call this breakpoint the bottleneck breakpoint, denoted T_{BBP}. Hence, from the data shown so far, we clearly see that T_{WAN}, affects the simulation time significantly, especially above T_{BBP}. The approximated formulas for all experiment sequences are given in Figure 11:

$$(1) \quad f_{2.5e-6,3N}(t) = 7964{,}4 + 46{,}8t$$
$$(2) \quad f_{5e-6,3N}(t) = 3735{,}5 + 30{,}0t$$
$$(3) \quad f_{10e-6,3N}(t) = 1845{,}1 + 15{,}7t$$
$$(4) \quad f_{10e-6,2N}(t) = 2911{,}0 + 15{,}9t$$

Figure 11. Linear approximations after using the least square method on measured simulation data. *f(t)* states simulation time in seconds and *t* represents T_{WAN} in ms.

All formulas in Figure 11 were approximated for T_{WAN} larger or equal to 80ms, except for the last case where $T_{WAN} \geq$ 110ms, since the linear characteristic started at larger values. One interesting observation is that changing the computational power for the bearing calculation changes the total simulation time, but not the derivative for the linear part of the curve (15,7 ≈ 15,9).

Another important parameter in data communication is the bandwidth in the communication link. Does the limitations of the bandwidth affect the total simulation time? The WAN simulator tool has the capability to simulate a smaller bandwidth. However, for this model, we noted that only a small bandwidth of about 35Kbit/s was needed. Since leased lines normally have the capacity of several MBit/s, this throughput requirement can be negligible. However, for larger models with many TLM-interfaces, the bandwidth factor may be important. To draw any further conclusions about how larger models affect the bandwidth, more experiments must be conducted, which is suggested as future research.

There are of course several other factors that may affect the total simulation time, which we have not measured in this experiment. One such parameter is the time step or tolerance level of the numerical integration routine for the simulation.

In the experiments, SSH was used to protect the TLM data between the nodes. The encryption and hash computations do indeed affect the delay in the system. As we can see in Table 1, the effect is not negligible (around 32% increased simulation time). We have not made any comparisons in this experiment what the effect would be when applying different encryption algorithms, even if there could be expected to be differences. Note however that if the co-simulation is performed on a trusted WAN, e.g., leased private lines, it might be acceptable to disable data encryption to improve performance.

To be able to compare the simulated delays of T_{WAN} with delays in real world networks, a number of measurements were performed over the Internet using the standard `ping` application. The tests were performed both using a connection at Linköping University (LIU), and a best effort connection from Stockholm, Sweden. Since we know that the simulation time grows linear to T_{WAN}, we can approximate the simulation time for different cities. The measured delays and estimated simulation times are presented in Table 2.

For values below $T_{BBP} \approx$ 75ms, estimations are taken from measured data and not the linear approximation. However, can we really expect that these approximations correspond to real distributed simulation? Since the experiments with the WAN simulator were made in an idealized environment, it can be expected that these results are optimistic.

University, Location	T_{WAN} **(ms) Lkp**	T_{WAN} **(ms) Stockholm**	**Simulation time (min)**
University of Trier, Germany	21	20	85
University of Cambridge, UK	19	19	85
MIT, Massachusetts, USA	57	56	96
Stanford, California, USA	96	87	108
University of Tokyo, Japan	142	143	134
University of Queensland, Australia	170	178	149

Table 2. Measured communication delays between Sweden and different locations all over the world. The optimistic approximated simulation times, using formula (2) in Figure 11, are given for T_{TLM} = 5e-6.

In the second part of the experiment, as given in Table 1, a real simulation via Australia was performed. A comparison between measured and approximated results is depicted in Table 3.

	T_{WAN}	T_{TLM} = **2.5e-6**	T_{TLM} = **5e-6**	T_{TLM} = **10e-6**
Ping RTT	348 ms	*404 min*	*236 min*	*122 min*
Max, RTT, SSH	470 ms	*499 min*	*297 min*	*154 min*
Min, RTT, SSH	340 ms	*398 min*	*232 min*	*120 min*
Mean, RTT, SSH	405 ms	*473 min*	*265 min*	*137 min*
Measured, Australia	-	495 min	277 min	133 min
Difference Measured to Ping RTT	-	-18,3 %	-14,7 %	-8,4%
Difference Measured to Mean, RTT, SSH	-	-4,5%	-4,4 %	2,8%

Table 3. A comparison between approximated simulation time and real measured simulation time.

Note that all times are round-trip-times and not one-way delays. Simulation times given in italic indicate an approximated value, using (1)-(4) in Figure 11. Row 5 corresponds to the experiment in Table 1.

It was discovered that the actual delay during the simulation varied from 340 ms to 470 ms (row 2 and 3), indicating a certain jitter in the communication[3].

In the table we can see that the ping RTT gives an optimistic value, resulting in an 18% to 8% too short simulation time. However, if we calculate the mean (row 4), a closer approximation is achieved.

Using formula (2) in Figure 11, together with the measured simulation time of the round-trip simulation in Table 1, we can compute the total simulation time it would take to simulate the shafts in Australia and the bearing in Sweden, i.e., use the delay

[3] Note that these delays show the RTT for the TCP packets of the SSH tunnel, which does not include the delay of the actual encryption/decryption.

instead of the RTT. For T_{WAN} = 5e-6, an approximated simulation time is ≈ 170 min; a 174% increase in time compared to an unencrypted local simulation that took 62 minutes.

5. Conclusions

We have in this paper discussed different aspects of *secure modeling and simulation*. Focus was put on secure distribution and simulation and both a centralized and a decentralized approach were discussed and compared from a security perspective. The decentralized approach was argued to have benefits regarding confidentiality and integrity, while giving open questions about performance and robustness.

An experiment was performed where a simple double pendulum with a bearing was co-simulated between Sweden and Australia. No robustness issue was discovered during simulation, and the simulation time was increased from approximately 62 min to 170 min (≈ 174% time increase), compared to a local simulation without encryption of data traffic.

To investigate which parameters that affect the total simulation-time, four series of tests were performed using a simulated environment of the wide area network (WAN). The main findings can be summarized as follows:

- The *network bandwidth* between simulation nodes has little effect, since throughput of TLM data is small.
- The *network delay (latency)* between simulation nodes has great impact on the total simulation time. The growth is linear after a certain breakpoint. Consideration must be taken to *jitter* (delay variations over time).
- The *TLM delay* of the model affects the simulation time significantly. Larger delay gives shorter simulation time, since the solver is allowed to take longer time steps.

Finally, we would like to conclude that secure co-simulation over long distances seems to be both a practical and possible solution for secure distribution and simulation of models within, and potentially between, enterprises.

6. Acknowledgements

This research was funded by CUGS (National Graduate School in Computer Science), by SKF, and by Vinnova under the NETPROG Safe and Secure Modeling and Simulation on the GRID project.

References

[1] David Broman. (2001). "Lossless Data Compression - Methods for Achieving Better Performance in a Wireless VPN". Master thesis. LiTH-ISY-EX-3159. Linköping University. Sweden.

[2] T. Dierks and E. Rescorla. (2006). "The Transport Layer Security (TLS) Protocol. Version 1.1". RFC 4346.

[3] Dag Fritzson, Jonas Ståhl, and Iakov Nakhimovski. (2007). "Transmission line co-simulation of rolling bearing applications". In Proceedings of the 48th Conference on Simulation and Modelling (SIMS'07). Göteborg. Sweden. Linköping University Electronic Press.

[4] IEEE 1706.1 Working Group. (1999). "IEEE Std 1076.1-1999, IEEE Standard VHDL Analog and Mixed-Signal Extensions". IEEE Press. USA.

[5] Petter Krus. (1999). "Modelling of Mechanical Systems Using Rigid Bodies and Transmission Line Joints". Transactions of ASME. Journal of Dynamic Systems Measurement and Control. December 1999.

[6] Petter Krus, Arne Jansson, Jan-Ove Palmberg, Kenneth Weddfelt. (1990). "Distributed Simulation of Hydromechanical Systems". Presented at Third Bath International Fluid Power Workshop, Bath, UK 1990.

[7] S. Lehtinen and C. Lonvick. Ed., (2006). "The Secure Shell (SSH) Protocol Assigned Numbers". RFC 4250.

[8] Modelica Association. (2007). "Modelica - A Unified Object-Oriented Language for Physical Systems Modeling - Language Specification Version 3.0". Available from: www.modelica.org.

[9] MSC Software. (2007). "MD Adams - overview". http://www.mscsoftware.com/.Last Access:Oct 3, 2007

[10] Iakov Nakhimovski. (2006). "Contribution to the Modeling and Simulation of Mechanical Systems with Detailed Contact Analysis". PhD Thesis. Linköping University. Sweden.

[11] Larry L. Peterson and Bruce S. Davie. (2000). "Computer Networks - A Systems Approach". Sec. Ed. ISBN 1558605770. Morgan Kaufmann Publ. USA.

[12] Alexander Siemers and Dag Fritzson. (2007). "A Meta-Modeling Environment for Mechanical System Co-Simulation". In Proceedings of the 48th Conference on Simulation and Modelling (SIMS'07). Göteborg. Sweden. Linköping University Electronic Press.

[13] Alexander Siemers, Iakov Nakhimovski, and Dag Fritzson. (2005). "Meta-modelling of Mechanical Systems with Transmission Line Joints in Modelica". In Proceedings of the Fourth International Modelica Conference. Pages 177-182. Hamburg. Germany.

[14] Lars Erik Stacke and Dag Fritzson. (2001). "Dynamic behaviour of rolling bearings: simulations and experiments". Proceedings of the Institution of Mechanical Engineers, Part J: Journal of Engineering Tribology. Volume 215. Number 6. Pages 499-508. Professional Engineering Publishing

[15] Lars Erik Stacke, Dag Fritzson, and Patrik Nordling. (1999). "BEAST—a rolling bearing simulation tool". Proceedings of the Institution of Mechanical Engineers, Part K: Journal of Multi-body Dynamics. Volume 213. Number 2. Pages 63-71. Professional Engineering Publishing