

An Extensible Simulation Framework for Critical Infrastructure Security

Technical Report: TUB-DAI 09/09-1

Joël Chinnow, Rainer Bye, Stephan Schmidt
Karsten Bsufka, Seyit Ahmet Çamtepe, Şahin Albayrak

E-Mail: {joel.chinnow, rainer.bye, stephan.schmidt, karsten.bsufka,
ahmet.camtepe,sahin.albayrak} @dai-labor.de

September 29, 2009



*DAI-Labor der Technischen Universität Berlin
Prof. Dr.-Ing. habil. Sahin Albayrak
Technische Universität Berlin / DAI-Labor
Institut für Wirtschaftsinformatik und Quantitative Methoden
Fachgebiet Agententechnologien in betrieblichen Anwendungen und der Telekommunikation
Sekretariat TEL 14
Ernst-Reuter-Platz 7
10587 Berlin
Telefon: (030) 314 74000
Telefax: (030) 314 74003
E-mail: Sekretariat@dai-labor.de
WWW: <http://www.dai-labor.de>*

Abstract

We introduce the Network Security Simulator (NeSSi²), an open source discrete event-based network simulator. It incorporates a variety of features relevant to network security distinguishing it from general-purpose network simulators. Compared to the predecessor NeSSi, it was extended with a three-tier plugin architecture and a generic network model to shift its focus towards simulation framework for critical infrastructures. We demonstrate the gained adaptability by different use cases.

Contents

1	Motivation	5
1.1	Organization	5
2	<i>NeSSi</i>²	6
2.1	Plugin Structure	6
2.2	Separation of Simulation and Presentation	7
2.3	Recording	7
2.4	Network Model	8
3	Worm spread in multi-AS topologies	8
3.1	Worm Mode	8
3.2	Topology Generation	9
3.3	Worm Application	9
3.4	Mitigation System	10
3.5	Simulation Setup	10
3.6	Case study results	11
4	DSL Access Network Link Utilization	12
5	Interdependency Analysis of CIs	13
6	Conclusions	14
7	Acknowledgments	14

List of Figures

1	The <i>Worm Mode</i> extension of <i>NeSSi</i> ²	9
2	Signature generation with a local preference worm	11
3	Performance Measurement of DSL network in <i>NeSSi</i> ²	12
4	Proposed CI interdependency analysis with <i>NeSSi</i> ²	13

1 Motivation

Critical Infrastructures (CI) play a vital role in modern society. Our daily lives become increasingly dependent on CIs and the services they provide. The wide application of information and communication technologies for CIs, in turn, has created a massive dependence on Critical Information Infrastructure (CII). The cyber attack on Estonia in 2007, where several web sites important for public life were attacked, is a good example in this context [13]. However, CIIs are vulnerable to intentional attacks and accidental failures, which we refer to as incidents. In this regard, the analysis of vulnerabilities or attacks in telecommunication networks plays a vital role for perpetuating service of interdependent CI.

Critical Information Infrastructures and dependent CIs form very complex systems. Therefore, measuring the impact of incidents on performance and availability of CIs is a challenging task. In most cases, experiments with a production system are risky or infeasible, and results from a small test-bed cannot be extrapolated. Mathematical models offer an alternative, but analytical solutions are hard to find and can often not be computed efficiently. However, simulation offers the means to evaluate them. Thus, a tool is needed to reflect the model constraints and perform the required simulations.

In this paper we present *NeSSi²*, the open source *Network Security Simulator*¹, successor of NeSSi introduced by Bye *et al.* [4]. The focus of NeSSi was the evaluation of net-centric intrusion detection systems in mid-sized telecommunication networks. In contrast, *NeSSi²* enhances the prior work by introducing exhaustive attack modeling on the one hand and providing a generic extension mechanism and data model on the other. In this fashion, apart from general purpose network simulation, *NeSSi²* is also prepared for discrete event-based simulation of other (interdependent) Critical Infrastructures.

1.1 Organization

First, we introduce the new plugin concept for *NeSSi²* to extend functionality without changing the simulation core itself. Here, we show three possible extension mechanisms imposing different levels of abstraction: *application*, *device* and *network* level. In this context, we also present the new *NeSSi²* data model which now allows the simulation of arbitrary topology types such as power grids.

Second, we point out extensibility and attack modeling with concrete case-studies: the first case-study presents a profound modeling of computer worm spread and network-based counter measures. As an example of network-level abstraction, topologies of multiple autonomous systems (AS) are realized. The next case-study shows *NeSSi²* as a performance analysis tool for customer access networks. This is the basis for a real-time Decision Support System. In the last case-study, we outline the application of *NeSSi²* for an upcoming work in modeling and simulation of interdependent Critical Infrastructures. Relevant related work is presented in the case study sections.

¹<http://www.nessi2.de>

2 *NeSSi*²

*NeSSi*² is a discrete event based network simulator with a focus on security-based research. It was published under the Apache License at the end of 2008. Realized scenarios are among others a cooperative artificial immune system [15], influence of monitor deployment [19] and a study which analyzed the interdependency of detector generation in distributed environments [3]. This section highlights the improvements of *NeSSi*² towards a general purpose simulation tool for CI.

2.1 Plugin Structure

Initially, *NeSSi*² was focused on fixed line network simulations of one AS. To broaden the use cases, we realized a generic plugin concept. It allows an easy extension of *NeSSi*² without changing the simulation core or the user interface. This also facilitates dynamic updates of *NeSSi*² itself as well as the integration of extensions. We offer three different interfaces for extensions: *Applications*, *Layer Handler* (device-level) and *Modes* (network-level).

Application Plugins: Applications offer a simple way of integrating third-party application-level protocols into *NeSSi*². For this purpose, the user needs to implement a predefined interface, which is needed to initialize or restart simulations between multiple runs.

Applications make use of the *NeSSi*² TCP/IP stack. Therefore the API provides methods for opening TCP (server) sockets, similar to the normal Java socket API, as well as methods for sending and receiving UDP packets. To realize protocol timer functionality, we offer methods for registering events as well as callback functions to handle them. Furthermore, applications can make use of the logging facility (see Section 2.3) to evaluate the examined measures.

We differentiate between two types, normal applications and promiscuous mode applications: the former receives only packets addressed to them, whereas the latter additionally provide direct access to the network interfaces. This allows the realization of intrusion detection or prevention systems.

Realized applications are integrated into *NeSSi*² and can be deployed using the graphical user interface, either by manually assigning them to network elements (devices and links) or distributing them automatically based on specified distribution parameters. Additionally, it is possible to define new device types with a customized user interface appearance, additional properties and a composition of predefined applications.

Layer Handler Plugins: A device generates traffic according to executed applications. But, the underlying infrastructure, i.e. the layer stack, is mandatory to enable necessary networking capabilities. In the predecessor of *NeSSi*², we have implemented a TCP/IP-based layer stack as described by Bye *et al.* [4]. Due to shortcomings regarding the flexibility and extensibility of the protocol stack, we introduced the Layer Handler to *NeSSi*².

The Layer Handler is a unique container for each device where layers and protocols are instantiated and administrated via fixed interfaces. This enables a flexible combination of new protocols and layers as well as their combinations for evaluation. In addition, the Layer Handler serves also as central data base for device characteristics as

well as reference to the simulation core for event handling and status information like current tick. In this regard, also meta-information, e.g. number of nodes, random web server IPs etc. is available.

Mode Plugins: Sometime network structures need a different kind of handling. Therefore a generic concept was implemented which supports the integration of modes (the term refers in this context to the initialization and handling of topologies). It interferes more than the other two plugin mechanisms with *NeSSi*², but allows the greatest changes as it directly influence the simulation core.

*NeSSi*² modes are instantiated during the start of the simulation core. The interface contains besides the life cycle, (re-) starting and stopping of simulations, a method for execution of ticks (discrete time units). Instead of programming a complete new handling, it is possible to extend existing handler classes with the required features. The mode used for a simulation can be defined in the user interface as fully classified class name. This is used to load it with the Java reflection API during the initialization.

2.2 Separation of Simulation and Presentation

A simulation tool benefits from a well designed user interface. It is possible to create small topologies in a fast way and larger (auto generated) topologies can be examined and adapted to specific needs. Furthermore, device or link based statistics of finished simulations can be examined in their context.

However, in the previous *NeSSi* version, the user interface was hardwired into the simulation core, negatively affecting performance and modularity. Therefore we split *NeSSi*² in two parts: a backend which handles the simulations itself and the user interface for network creation and evaluation of results. Decoupling of the user interface considerably increased the performance which is critical for carrying out large scale simulations. An additional benefit of the modular structure is the optional use in distributed environments, where a single user interface terminal can be used to control multiple running simulations.

2.3 Recording

The old *NeSSi* version had only supported the recording of attack specific events into a database. This was sufficient for the past use cases, but unsatisfactory for our generic extensions. Therefore we used *log4j*, a standard JAVA logging engine, to shift *NeSSi*² towards a general purpose simulator. *Log4j* can be configured with the help of appender. We focused mainly on three different types: With the default database appender, it is possible to log different data types and the occurrence of events, global or bound to a specific network element (device or link). Besides the advantage that the data is stored persistent, this allows examination of data in the user interface and results can easily shared between users. Otherwise a file appender can be used for generating data in a *gnuplot*² compatible format. This increases the performance and thus the required runtime, but circumvents the user interface-based evaluation. A third appender is used

² <http://www.gnuplot.info>

to log parts of the network traffic as pcap file, which is helpful for debugging protocol implementations with tools like *wireshark*³.

The *NeSSi*² user interface comes with a default visualization and evaluation module based on jFreeChart⁴. If the recording is configured to use a database, finished and even running simulations can be examined graphically in a dedicated *statistics view*. Providing a wide variety of different chart types (bar, graph, cumulative etc), it is highly customizable to the user's needs.

2.4 Network Model

Our previous network model was bound to IP networks. As most CI (e.g. pipelines, power networks, etc) are non-IP networks, we generalized our network model to generic nodes and edges. They include only the properties required to display them in the user interface (e.g. position, name, etc). For specific scenarios they can be extended with the Java Generics concept. This allows the creation of plugins for the respective topology. Furthermore, these models can be combined to simulate interdependent CIs. Additionally, we offer common graph theoretical algorithms.

With the developed extensions, *NeSSi*² can be used as a simulation tool for the analysis of different CI scenarios. In the following sections we substantiate the benefits of our improvements through exemplary *NeSSi*² extensions.

3 Worm spread in multi-AS topologies

The first case study is based on a diploma thesis [6]. We want to evaluate how effective the spread of computer worms can be mitigated through network-based countermeasures. The main focus here was the time constraint for a successful mitigation. First we describe the realized *NeSSi*² enhancement: the *Worm Mode*, a topology generator, a generic worm model and counter-measures. Second, we describe the simulation setup and explain the resulting measurements.

3.1 Worm Mode

In order to be able to simulate multi-AS topologies without scalability problems, we developed a *Worm Mode* as a *NeSSi*²-*Mode*. Inside an access network, the connections from the broadband remote access router (BB-RAR) to the end systems are point-to-point and thus negligible for network-centric counter measures. Hence it is sufficient to have a single entity which contains the combined worm models for all the administrated hosts. We realized the abstracted topology by extending *NeSSi*² with a new router type. It handles traffic like normal router, but additionally administers a *net block*. Within this net block, the end systems are described as a list containing an IP address, the instantiated worm models, a *susceptible* and an *infected* flag. When a net block router (NBR) receives a worm packet, it has to lookup if the IP is used and susceptible. In this case, a worm model generating infected traffic is instantiated. The combined worm

³ <http://www.wireshark.org/>

⁴ <http://www.jfree.org/jfreechart/>

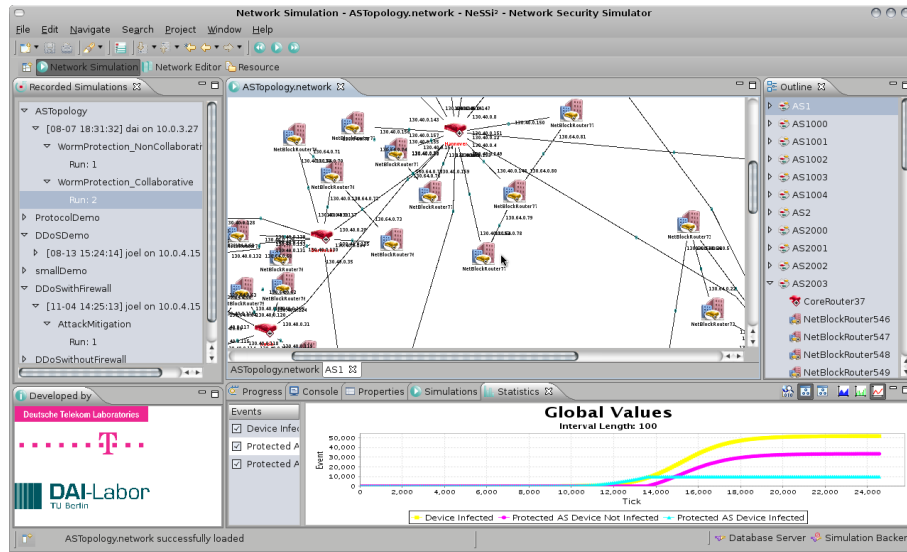


Figure 1: The *Worm Mode* extension of *NeSSI*²

traffic of all contaminated hosts in this net block will be sent via the NBR into the AS. In order to allow simulations at a more fine-grained resolution (i.e. examination of intra-AS infections), it is desirable to have multiple NBRs in one AS. Figure 1 shows the inspection of a single finished simulation run with the graphical user interface of *NeSSI*².

3.2 Topology Generation

We implemented a generic converter in order to handle different types of AS structures. This converter parses a given topology file and creates a corresponding *NeSSI*² data structure. The resulting file format represents connections between two AS as a line with both AS numbers, separated by a tab. This simple format allows the use of existing AS topology generators to create the topology files. Furthermore, it is possible to use real-world data. A viable option to obtain AS topologies is the CAIDA website⁵. The resulting data structure is a *NeSSI*² network model where AS are represented by subnets. Inter AS connections are directly taken from the AS topology file, and are represented by an AS border router. Furthermore, every AS contains several NBR. The API offers two ways of determining the exact number of NBRs in a subnet; it can be set to a fixed value or initialized by a randomization routine.

3.3 Worm Application

A computer worm can be modeled as a simple extended finite state machine (EFSM). In our simulations, we implemented a generic model based on a configurable local preference scanning algorithm, which can also realize random scanning. The worm is continuously scanning with different ranges for the PRNG. It varies only the last octet of the address for n_1 scans, the last two for n_2 scans, the last three for n_3 scans and n_4 random scans. Additionally, it is possible to define a maximal range for an octet with $m_i = x$.

⁵ <http://www.caida.org/data/active/as-relationships/index.xml>

As an example, worms on a device with the IP address 130.1.2.3 and $m_3 = 10$ will scan between 130.0.0.0 and 130.10.255.255. This is necessary since only a fraction of the total IP space is simulated. Based on this scanning scheme, it is possible to adapt the worm to the AS topology under attack.

3.4 Mitigation System

A mitigation system consists of two parts: On the one hand, a detection method is needed. Algorithms can be based on different features like error messages [2], connection attempt frequencies [12], dark address space [1], address dispersion [5] or even only payload [16]. Additionally, it is a practical option to use a combination of different approaches [10, 17]. On the other hand, it is necessary to use an automated containment as manual intervention is too slow for fast spreading worms like SQLslammer. The possible approaches can be classified in two groups: First, malicious connections can be blocked or rate-limited based on a list of infected IP addresses. Second, with the help of signatures it is possible to block only malicious connections.

We have chosen an error message-based system as this offers a reliable detection of scanning worms when the amount of susceptible hosts is much less than the total sum. It works in the following way: if a host exceeds a threshold of n_{cf} failed connection attempts, it will be marked as suspicious. Afterwards, its network traffic will be cloned and sent to a signature generation server. The time needed to generate a signature is t_{sg} . In order to avoid the deployment of an incorrect signature, a certain amount n_s of identical signatures, belonging to different hosts, needs to be generated to trigger the deployment. Subsequently, the signature will be deployed to the BB-RAR. They are able to match the deployed signatures and discard UDP packets or abort TCP connections that belong to a worm. The time needed for the deployment is t_{sd} , which enables us to express the percentage of protected hosts as:

$$p_{prot} = \frac{1 - I(t_d + t_{sg} + t_{sd})}{S} \quad (1)$$

where t_d is dependent on n_s and can be obtained from the simulation data.

3.5 Simulation Setup

The topology we finally used consists of a core formed by six transit-AS, which represent larger ISPs. Each of them has five or six connections to stub AS, resulting in a total size of 41 AS. As every AS utilizes a /16 net, the available IP space is larger than 2.6 million addresses, 1.9 percent of which are susceptible after the vulnerability model initialization process. This leads to a final number of over 50000 infected hosts. To adapt the worm to the used network topology, we used $n_1 = 1$, $n_2 = 3$, $n_3 = 3$, $n_4 = 0$ and 256 for all m_i .

$$\delta(n, \alpha) = t_{n-1, 1-\alpha/2} \sqrt{\frac{S^2(n)}{n}} \quad (2)$$

We used equation 2 with the point in time where half of the susceptible population is infected, 0.15 for the relative error γ and 0.2 for α for run-length control. This means

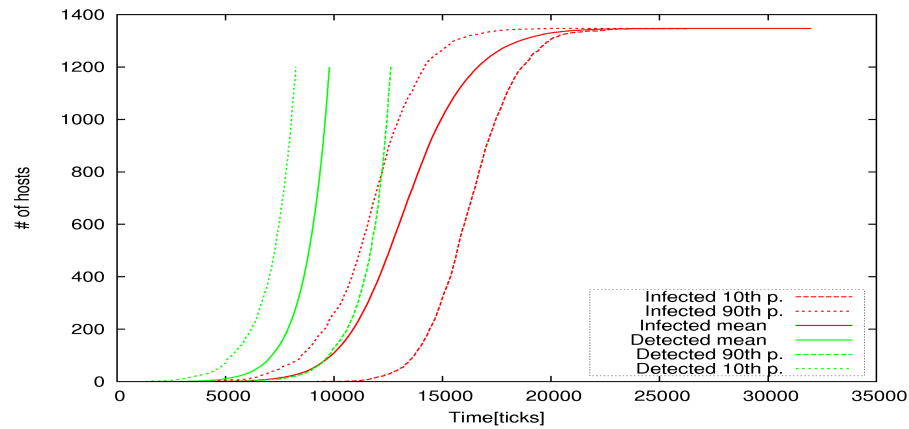


Figure 2: Signature generation with a local preference worm

we stopped the simulation after the quotient of $\delta(n, \alpha)$ and $|\bar{X}(n)|$ was smaller than $\frac{\gamma}{1+\gamma}$.

3.6 Case study results

Figure 2 shows the detection and infection results for selected protection values; in this context, the number of infections is limited to a single AS. The green line represents the amount of detected scanning hosts with an upper bound fixed at 1200. The overlapping areas formed by the detection and infection percentiles are attributed to the stochastic nature of the worm propagation. The rightmost percentile represents the slowest overall detection speed and naturally corresponds to the slower worm spreading. Seen on a single run basis, the detection is always triggered sufficiently early. With the use of equation 1 it is possible to appraise the efficiency of the signature-based approach more detailed. On the one hand, it is possible to determine p_{prot} for a specific system with given parameter. On the other hand, we can get an estimation for requirements needed to protect a certain percentage of hosts by transform equation 1.

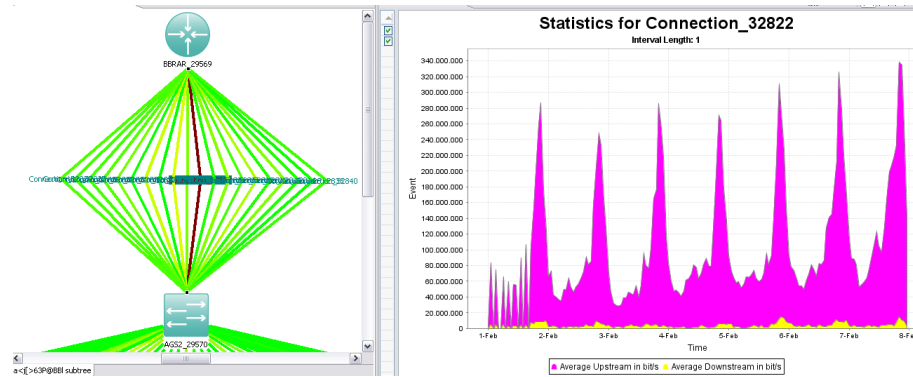


Figure 3: On the left, a part of a DSL access network is shown. Links with a color range from green to yellow signal a link utilization of less than 50 % (green lower percentages, yellow higher percentages), links with utilization above 50 % are displayed in red. On the right side the measured bandwidth for a selected link (red on the left side) is shown.

4 DSL Access Network Link Utilization

Traffic management and optimization usually is looked at from an IP network perspective, e.g. [8, 9, 20], but these optimizations do not affect the PPP over Ethernet (PPPoE) sessions that are established between a DSL customer and a BB-RAR. This means that congestions may occur already before any IP-based network optimization approaches will have an effect.

In a current project *NeSSI²* is used to simulate the actual traffic flows in DSL access networks, see Figure 3, and to test the effects of alternate traffic distributions on the current network infrastructure. In case any of these simulations detect traffic congestions, *NeSSI²* extensions generate warnings and propose actions to neutralize the congestion, e.g. reassign customer traffic to alternate links or to increase bandwidth on links or add additional links.

The default operation mode of *NeSSI²* is, that networks are manually created and traffic is simulated on an IP packet level. This of course is unsuitable for the tasks described above. The *NeSSI²* plugin concept allows changing the traffic model from a packet-oriented model to a fluid traffic model [11, 14]. Another more complex change was needed in regard to the network generation, since it is not practically to regenerate a DSL access network with several thousands of nodes by hand. To cope with this problem a specialized agent was developed, which is capable to parse existing topology information of the DSL access network and translate it to *NeSSI²* network files. In addition to this agent, a user interface plugin was created that allows fetching these generated network files and displaying them in *NeSSI²*. Additionally, this agent is responsible for translating actual metric information from the DSL access network to *NeSSI²* simulation events. These events are used to simulate (replay) the current state of the DSL access network and will be used as input values for computing traffic distribution functions for each DSLAM node in the network. To analyze the network under different conditions than the observed ones, a *NeSSI²* application will be created that allows the modification of existing traffic distribution functions for a DSLAM. In addition it will be possible to reassign customer traffic to different links within the DSL access network.

Based on the available topology information, it is also possible to identify weak

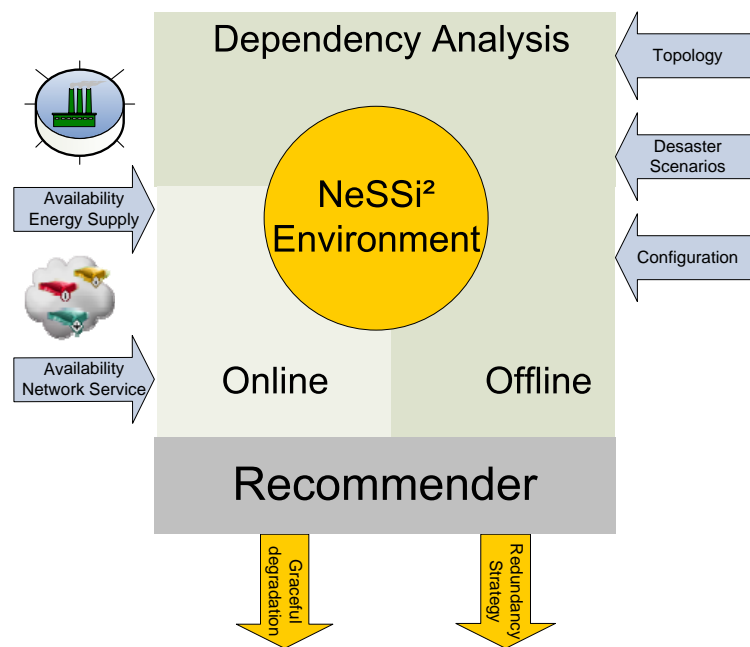


Figure 4: Conceptual view of a proposed framework for analyzing CI interdependencies based on *NeSSi²*

points in the infrastructure, which are vulnerable to denial of service attacks or physical attacks. This risk analysis and the automatic generation of alternate topologies with build-in redundancy are out-of-scope of our current research, but will be very likely part of upcoming research activities, such as the one described in the following use case scenario.

5 Interdependency Analysis of CIs

In an upcoming project, *NeSSi²* will be used for modeling and analysis of interdependencies between different Critical Infrastructures, namely electricity generation and telecommunication. In this regard, the European IRRIS project (Integrated Risk Reduction of Information-based Infrastructure Systems) ⁶ focusing on modeling and evaluation of interdependencies between Critical Infrastructures represents related work. Here, the *federated simulation environment SimCip* (Simulation for Critical Infrastructure Protection) has been developed. As a proof-of-concept, the special purpose simulators ns-2⁷ for telecommunication and SINCAL⁸ for electricity networks have been integrated. The combination of existing tools is also carried out by Duflos *et al.* under the term overlay simulation [7]. In general, Rinaldi *et al.* discuss different categories of interdependencies and failures and line out the challenges of their modeling and simulation [18].

The application of a dedicated environment like *NeSSi²*, in contrast to federated simulation has its merits as well: it is easier to set up scenarios using one graphical inter-

⁶<http://www.irriis.org/>

⁷<http://www.isi.edu/nsnam/ns/>

⁸http://www.simtec-gmbh.at/sites_en/sincal.asp

face and develop with a single API. This circumvents also the redundant adaption of topologies for different simulators. Additionally, we expect, by using the *NeSSi²* extension mechanisms, a highly customized simulation approach for improved performance needed for a large amount of replications. This is essential for statistically well-founded results.

Figure 4 depicts the conceptual architecture of the intended solution based on *NeSSi²* for the upcoming project. *NeSSi²* will be the central component of a framework for modeling, analysis, simulation, cooperation and evaluation. This system receives input parameters in form of reference network topologies of electricity as well as telecommunication networks. In addition, service-level configurations, i.e. prioritization of communication-services or customers, will be processed. Possible disaster scenarios enable the “what-if”-scenario evaluation. In addition, we consider online availability information of electricity network as well as telecommunication network as beneficial input to take the dynamics of the Critical Infrastructures into account. The expected output of such a system is considered helpful for two target groups: first, redundancy strategies are recommended for network planning department and second, the graceful degradation mechanism supports the operative decision makers in case of emergencies.

6 Conclusions

We realized a generic plugin structure, which allows an easy adaption of *NeSSi²* to various CI scenarios. The plugin concept was evaluated with the implemented *Worm Mode* and the access network optimization scenario. For the future, we plan to enhance the simulation capabilities of *NeSSi²*. To broaden the use cases, we want to publish additional plugins like the afore mentioned extension for power networks as well as for wireless networks.

7 Acknowledgments

This work is part of a project funded by Deutsche Telekom Laboratories. We additionally appreciate the work of the whole *NeSSi²* team as well as the fruitful discussion within our whole security research group.

References

- [1] S. Antonatos, K. Anagnostakis, and E. Markatos. Honey@home: a new approach to large-scale threat monitoring. In *WORM '07: Proceedings of the 2007 ACM workshop on Recurring malware*, pages 38–45, New York, NY, USA, 2007. ACM. 3.4
- [2] G. Bakos and V. Berk. Early detection of internet worm activity by metering ICMP destination unreachable messages. In *In Proceedings of the SPIE Aerosense*, pages 33–42, 2002. 3.4

- [3] R. Bye, K. Luther, S. A. Çamtepe, T. Alpcan, Şahin Albayrak, and B. Yener. Decentralized detector generation in cooperative intrusion detection systems. In S. Masuzawa, Toshimitsu; Tixeuil, editor, *Stabilization, Safety, and Security of Distributed Systems 9th International Symposium, SSS 2007 Paris, France, November 14-16, 2007 Proceedings*, Lecture Notes in Computer Science , Vol. 4838. Springer, 2007. 2
- [4] R. Bye, S. Schmidt, K. Luther, and S. Albayrak. Application-level simulation for network security. In *Proceedings of the First International Conference on Simulation Tools and Techniques for Communications, Networks and Systems*, 2008. 1, 2.1
- [5] M. Cai, K. Hwang, Y.-K. Kwok, S. Song, and Y. Chen. Collaborative internet worm containment. *IEEE Security and Privacy*, 03(3):25–33, 2005. 3.4
- [6] J. Chinnow. Simulation and Evaluation of a Network-Based Computer Worm Counter Measure. Diploma thesis, Technical University of Berlin, 2008. 3
- [7] S. Duflos, A. A. Diallo, and G. L. Grand. An Overlay Simulator for Interdependent Critical Information Infrastructures. In *Proceedings of the 2nd International Conference on Dependability of Computer Systems*, pages 27–34, June 2007. 5
- [8] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True. Deriving traffic demands for operational ip networks: methodology and experience. *IEEE/ACM Transactions on Networking*, 9(3):265 – 279, Jun 2001. 4
- [9] G. Goth. Traffic management becoming high-priority problem. *IEEE Internet Computing*, 12(6):6 – 8, Nov. 2008. 4
- [10] J. Jung, V. Paxson, A. Berger, and H. Balakrishnan. Fast portscan detection using sequential hypothesis testing. In *Proceedings IEEE Symposium on Security and Privacy, 2004*, pages 211–225, 9-12 May 2004. 3.4
- [11] C. Kiddle, R. Simmonds, C. Williamson, and B. Unger. Hybrid packet/fluid flow network simulation. In *Proceedings Seventeenth Workshop on Parallel and Distributed Simulation, 2003. (PADS 2003)*, pages 143–152, 2003. 4
- [12] B. Kim, S. Bahk, and H. Kim. Fdf: Frequency detection-based filtering of scanning worms. In *Communications, 2006 IEEE International Conference on*, volume 5, pages 2124–2129, June 2006. 3.4
- [13] M. Lesk. The new front line: Estonia under cyberassault. *IEEE Security and Privacy*, 5(4):76–79, 2007. 1
- [14] B. Liu, D. Figueiredo, Y. Guo, J. Kurose, and D. Towsley. A study of networks simulation efficiency: Fluid simulation vs. packet-level simulation. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1244–1253, 2001. 4

- [15] K. Luther, R. Bye, T. Alpcan, S. Albayrak, and A. Müller. A Cooperative Approach for Intrusion Detection. In *IEEE International Conference on Communications (ICC 2007)*. IEEE, 2007. 2
- [16] M. V. Mahoney. Network traffic anomaly detection based on packet bytes. In *SAC '03: Proceedings of the 2003 ACM symposium on Applied computing*, pages 346–350, New York, NY, USA, 2003. ACM. 3.4
- [17] S. Ranjan, S. Shah, A. Nucci, M. Munafa, R. Cruz, and S. Muthukrishnan. Downtitcher: Effective worm detection and containment in the internet core. *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 2541–2545, May 2007. 3.4
- [18] S. Rinaldi. Modeling and simulating critical infrastructures and their interdependencies. pages 8 pp.–, Jan. 2004. 5
- [19] S. Schmidt, T. Alpcan, S. Albayrak, and A. Müller. A Monitor Placement Game for Intrusion Detection. In *Proc. of CRITIS, 2nd International Workshop on Critical Information Infrastructures Security*, volume 5141 of *Lecture Notes in Computer Science*. Springer, 2007. 2
- [20] T. Ye, H. Kaur, S. Kalyanaraman, and M. Yuksel. Large-scale network parameter configuration using an on-line simulation framework. *IEEE/ACM Transactions on Networking*, 16(4):777–790, Aug 2008. PMNO. 4