Simulating Online Regulation in Distributed Techno-socio-economic Systems

Evangelos Pournaras, Ben-Elias Brandt, Jose Espejo-Uribe, Manish Thapa, Dirk Helbing

Professorship of Computational Social Science
ETH Zurich, Zurich, Switzerland
{*epournaras,bbrandt,josee,mthapa,dhelbing*}*@ethz.ch*

Abstract

The introduction of ICT in techno-socio-economic systems, such as Smart Grids, traffic management, food supply chains and others, transforms the role of simulation as a scientific method for studying these complex systems. The scientific focus and challenge in simulations move from understanding system complexity to actually prototyping online and distributed regulatory mechanisms for supporting system operations. Existing simulation tools are not designed to address the challenges of this new reality, however, simulation is all about capturing reality at an adequate level of detail. This paper fills this gap by introducing a Java-based distributed simulation framework for inter-connected and inter-dependent techno-socio-economic system: SFINA, the Simulation Framework for Intelligent Network Adaptations. Three layers outline the design approach of SFINA: (i) integration of domain knowledge and dynamics that govern various techno-socio-economic systems, (ii) system modeling with dynamic flow networks represented by temporal directed weighted graphs and (ii) simulation of generic regulation models, policies and mechanisms applicable in several domains. SFINA aims at minimizing the fragmentation and discrepancies between different simulation communities by allowing the interoperability of SFINA with several other existing domain backends. The coupling of two such backends with SFINA is illustrated in the domain of Smart Grids with a simulation scenario on cascading failures. The same model of cascading failures is developed once and evaluated with both MATPOWER and InterPSS backends without changing a single line of application code. Results provide a proof-of-concept for the high modularity and reconfigurability of SFINA and puts the foundations of a new generation of simulation tools that prototype and validate online decentralized regulation in techno-socio-economic systems.

Keywords: simulation, framework, techno-socio-economic system, flow network, agent, distributed system, cascading failure, inter-dependent networks

1. Introduction

Simulation has traditionally been one of the cornerstone scientific methods for understanding the complexity of various technical, social or economic systems. However, the introduction of cutting-edge ICT technologies[1] in such systems has brought fundamental changes in their design and real-time operation with disruptive implications for the current simulation approaches.

[1]Such technologies include 'Internet of Things', 'Big Data' and other.

Preprint submitted to Technological Forecasting and Social Change *November 22, 2015*

On the one hand, techno-socio-economic systems are nowadays highly inter-connected, inter-dependent and distributed [1, 2]. For example, human mobility influences traffic systems, traffic systems rely on communication networks that also support the operations of Smart Grids, which at the end energize all other aforementioned systems. A data-driven multi-disciplinary approach is required to understand these interactions and inter-dependencies. On the other hand, the regulation of such complex inter-connected systems evolves to online, automated and decentralized control systems running intelligent software mechanisms. Existing simulation tools cannot anymore imitate the real-world operations of techno-socio-economic systems as these are nowadays of a total different nature. Alternative distributed simulation tools, of a fundamentally different design approach, are required by researchers, engineers, policy-makers and operators to build and run decentralized regulation mechanisms for inter-dependent techno-socio-economic systems.

This paper contributes such an alternative distributed simulation tool: *SFINA, the Simulation Framework for Intelligent Network Adaptations*. SFINA aims at bridging the gap of the highly fragmented work on simulation between different scientific disciplines. This is achieved by splitting the simulation complexity in three levels that form the conceptual layers of SFINA. The first layer integrates domain knowledge and dynamics from various domains. Existing simulation tools can interoperate with SFINA. The second layer provides a modeling abstraction using domain-independent flow networks represented by temporal directed weighted graphs. The third layer supports the prototyping and evaluation of regulation models, policies and mechanisms over the underlying flow networks. Therefore simulations are validated using different coupled backends of the same domain or even across multiple domains.

This paper also contributes a framework realization that shows an application scenario and provides a proof-of-concept for the modularity, reconfigurability and scalability of SFINA compared to related work. The application scenario concerns the modeling of cascading failures in Smart Grids. Cascading failures are complex phenomena to understand and challenging to prevent or mitigate. They have traditionally been a source of societal costs, disorder and chaos. Online regulation of techo-socio-economic systems results in new threats of reliability and system robustness [1, 3, 4]. This paper shows that the same model of cascading failures can be simulated using the power flow analysis from two different domain backends. Simulation performance varies and results about the impact of cascading failures are not always in symphony confirming the requirement for a multi-perspective modular simulation in future work.

This paper is organized as follows: Section 2 provides an overview of the SFINA framework. Section 3 illustrates the architecture of SFINA in detail. Section 4 introduces a realization of SFINA with an application scenario on cascading failures in Smart Grids. Section 5 compares SFINA with related work. Finally, Section 6 concludes this paper and outlines future work.

2. Overview

There is often a significant fragmentation in the data formats, models and tools used by academic and industrial communities when studying and optimizing techno-socio-economic systems such as transportation systems, financial markets or infrastructural networks, i.e. power, gas and water networks. This means that several communities use or develop their own data formats, models and tools that do not allow a more universal evaluation and comparison of models and mechanisms. This phenomenon results in discrepancies of findings or very customized solutions with limited applicability.

Motivated by this limiting status-quo, this paper introduces SFINA, the *Simulation Framework for Intelligent Network Adaptations*. SFINA represents complex techno-socio-economic systems as temporal flow networks modeled by dynamic directed weighted graphs. In contrast to static undirected unweighted graphs that only show a snapshot of interactions, a temporal flow network encompasses both structural and dynamical aspects of most techno-socio-economic systems. The core operation of SFINA is the flow analysis that computes the flow in a network given its physical characteristics in an application domain. The grant objective of SFINA is to provide development toolkits to build and evaluate generic and modular flow regulation mechanisms applicable in different flow analysis models and, even application domains.

SFINA is outlined in three layers: (i) *Domain knowledge and dynamics* concern real-world data and physical laws that govern techno-socio-economics systems. (ii) *Flow networks* are an abstraction of domain knowledge and dynamics. (iii) *Regulation models, policies and mechanisms* are generic domain-independent implementations by the users of the SFINA software.

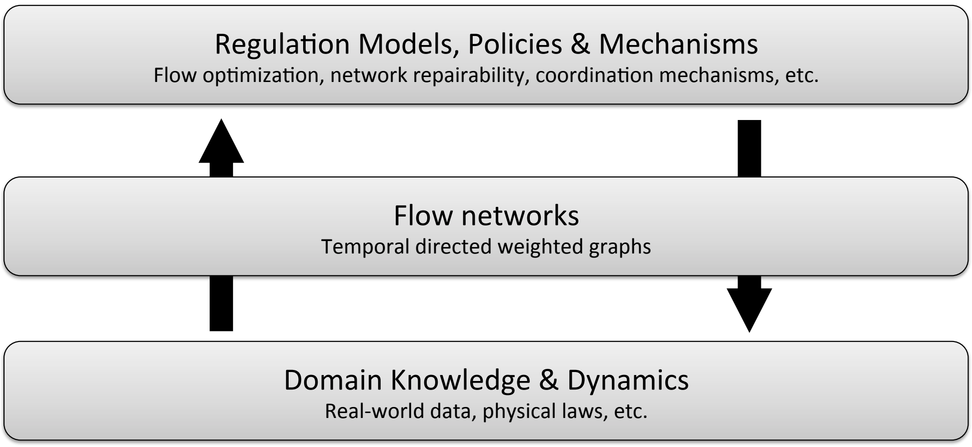


Figure 1: The three main layers of the SFINA framework.

The three layers of SFINA enable an integrated analysis and prototyping of techno-socio-economic systems with two novel capabilities:

1. Support of flow analysis and regulatory models for systems of the same domain.
2. Support of flow analysis and regulatory models for systems of several independent or interdependent domains.

These capabilities result in the following advantage: An implemented flow regulation model can be evaluated with different integrated tools of the same domain but also among different domains. For example a flow regulation mechanism for mitigating cascading failures can be evaluated with multiple integrated tools of a certain domain, e.g. power networks. One tool may support the power flow analysis and another tool the transient stability (Capability 1). Moreover, the same flow regulation mechanism can be evaluated in another domain as well, e.g. gas and water networks (Capability 2).

SFINA is by design a decentralized multi-agent system and it is an open-source implementation in Java. A SFINA user writes once an application and evaluates it by integrating different tools of the same or different application domain without changing a single line of code in its application.

3. The SFINA Architecture

The three layers of Figure 1 are realized with the architecture of Figure 2. SFINA consists of 6 components: (i) the *file system*, (ii) the *Protopeer toolkit*, (iii) the *flow analyzer*, (iv) the *flow network*, (v) the *simulation agent* and (vi) the *applications*. The file system loads all required information for the simulation. Input data can be reloaded during simulation and output data are written to files during simulation for further post analysis. Protopeer is a distributed prototyping toolkit that provides networking, scheduling, logging and deployment services to the overall SFINA framework. The flow analyzer computes the distribution of the flow in the network according to the selected domain backend. The flow network contains and manages information about the nodes, the links and their topology. The simulation agent orchestrates all other components by scheduling operations and executing simulation events. Finally, applications expand the functionality of the simulation agent and implement policies, models and mechanisms for the flow regulation.

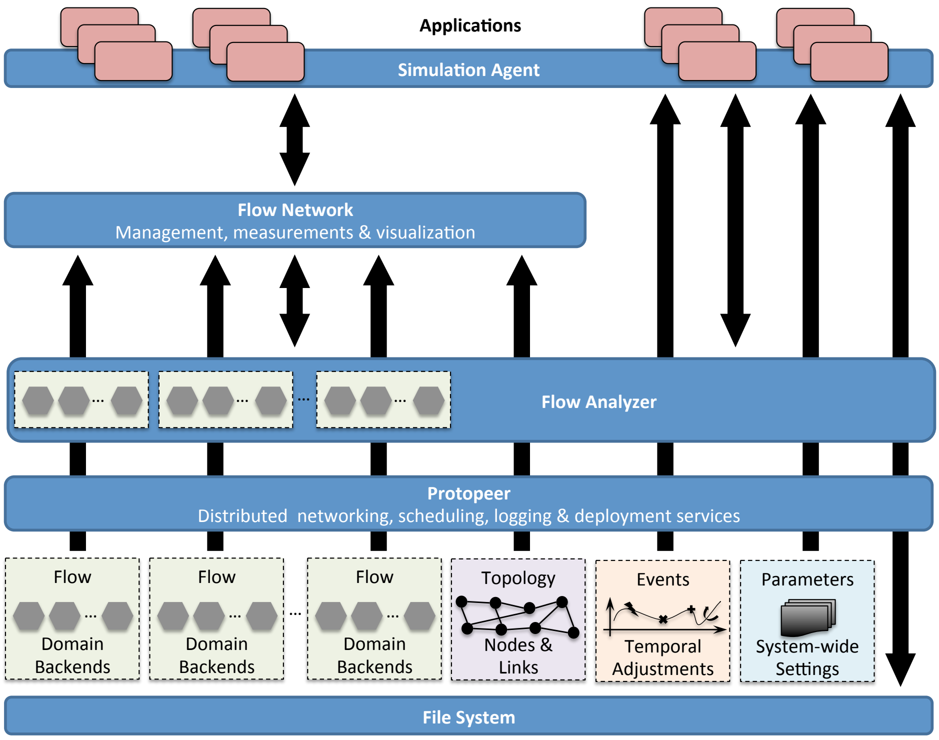


Figure 2: An outline of the SFINA framework.

The execution of a simulation consists of the following runtime periods: (i) *system*, (ii) *bootstrapping*, (iii) *simulation*, (iv) *step* and (v) *iteration*. Figure 3 outlines the time management of the SFINA framework.

The system runtime is the total execution period of the SFINA simulation software. It is measured with a Protopeer clock in simulation or live deployment mode. System runtime consists

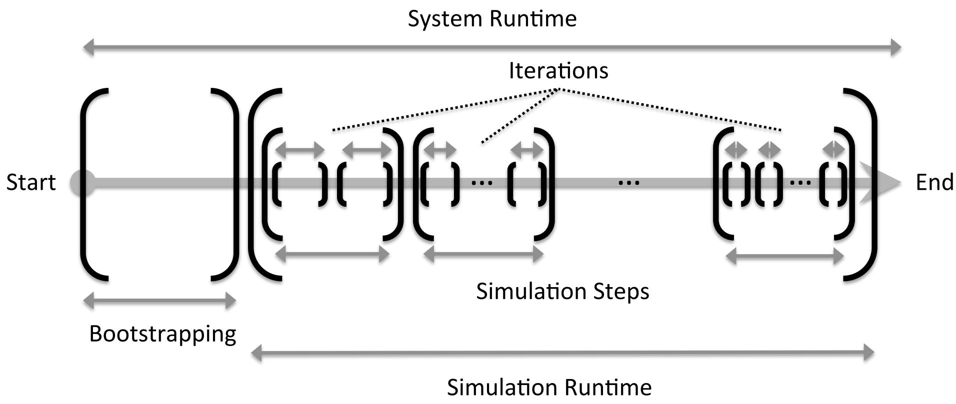


Figure 3: Time management in SFINA

of the bootstrapping runtime and simulation runtime. During bootstrapping, the simulation is initialized by loading the required input information for the simulation. The duration of the bootstrapping period is chosen by the user. Simulation runtime is equal to the system runtime minus the bootstrapping runtime. It consists of discrete simulation steps of equal length during which measurements are logged, events are executed and flow analysis is performed. The main operations of applications are also executed in each of the simulations steps. Finally, each simulation step optionally defines a number of iterations that model sequences of evolving events. For example, Section 4 shows that the failure of a power line in a power network may result in a cascading failure during which several other lines consecutively fail. The evolution of a cascading failure can be modeled and studied with the iterations of SFINA.

An event in SFINA is the adjustment of flow, topology or system parameters during simulation runtime. An event can be *static* or *dynamic*. Static events are encoded offline and manually by the user. They are loaded in the simulation by the file system. Dynamic events are created in an automated fashion during simulation by the application logic.

A SFINA application can run single-threaded or multi-threaded in a single machine. It can also run in a cluster or even be deployed in a real-world operational system such as Planetlab[2] or a cloud computing infrastructure [5]. Protopeer supports these deployments options [6]. Distributed simulation scenarios may include computationally intensive simulations, modeling of inter-dependent flow networks or even multi-domain simulations.

3.1. File system

SFINA relies on a stuctured file system for (i) a user-friendly dynamic loading of all the required data and (ii) exporting the simulation output during simulation runtime. Four file types are loaded in system runtime with the following information:

1. *Flow*: It contains domain-dependent information about the physical characteristics of the network that govern how flow is distributed. For example, for a power system this file may contain the values of line impedance, node voltages, etc. For traffic systems it may contain physical characteristics of vehicles, traffic lights or road intersections.

[2] Available at `https://www.planet-lab.org` (last accessed: November 2015)

2. *Network*: It contains domain-independent information about the topology, meaning which nodes the links connect.
3. *Events*: It encodes changes of information about a node, link or system parameter at a certain simulation step. For example the removal of a line at a certain step can be encoded by a static event.
4. *Parameters*: It contains global system-wide parameters of the simulation such as which domain backend is selected for the simulation.

Flow and network file types can be reloaded and exported at any simulation step, in contrast to the events and parameters that are loaded only once at the initialization phase. At this development stage, the SFINA file format concerns comma-separated text files for easier use and readability of the data used in the simulation. SFINA contains extensible utilities that convert back and forth other file formats[3] to SFINA format so that the user can integrate state-of-the-art datasets generated from various domain backends. The data loaders of the file system are modular and can be extended to parse future data types required in simulations.

3.2. The Protopeer toolkit

SFINA is prototyped as a distributed system by design using the interfaces of the Protopeer library [6]. Figure 4 illustrates an outline of the Protopeer architecture. The core simulation agent extends the functionality of the 'peerlet component'. Therefore, all SFINA applications written on top of the core simulation agent have a network API with which they can exchange messages in real-world networking environments. Moreover, SFINA applications have access to measurement and logging services that can be used for a post-analysis of the simulation results. The core simulation agent inherits the scheduling functionality of Protopeer that reduces low-level programmatic effort required by developers of applications.

In summary, low level complexity such as scheduling, networking, logging and measurements is hidden from developers of the SFINA applications that can focus entirely on the development of flow regulation functionality. Although the current SFINA implementation is supported by Protopeer library, the design of SFINA does not depend on Protopeer. SFINA interfaces allow the replacement of Protopeer by another backend if this becomes a requirement in the future.

3.3. Flow analyzer

Flow analysis is the calculation of the flow in the lines and nodes of the flow network given the physical characteristics of the studied domain and perturbations introduced by the executed events. A flow analysis is performed at every simulation step and follows the execution of static and dynamic events. The output of a flow analysis can be used to tailor decision-making and future regulatory actions on the flow network.

A flow analysis highly depends on the domain-specific physical characteristics of the studied network and the physical laws that govern such a network. SFINA provides to applications a generic interface for flow analysis that is agnostic of the actual implementation of the adopted domain backend. It is the simulation agent that makes the acquired instantiation of the flow analysis based on the selected domain backend adopted in the simulation. In this way, the application developer does not need to implement flow analysis algorithms but can use several existing ones implemented at different domain backends.

[3] Such formats include the MAT-File and the IEEE Common Data Format.

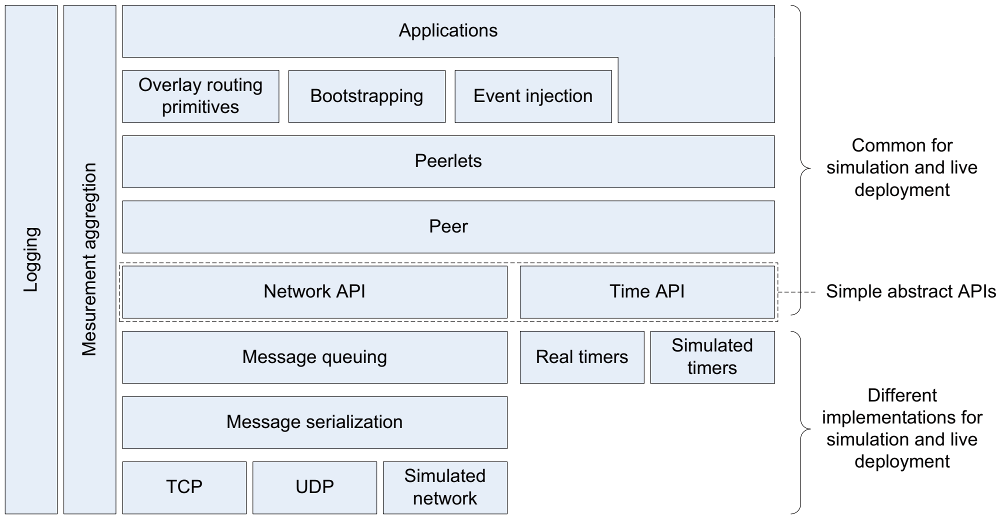


Figure 4: An outline of the Protopeer architecture as illustrated in earlier work [6]. SFINA inherits all services and prototyped functionality of Protopeer and therefore it can be deployed and run as a fully decentralized networked system.

3.4. Flow network

A flow network in SFINA consists of a set of uniquely identified nodes and directed links that contain the flow information loaded from the respective file type. A node contains a set of incoming and outgoing links. Respectively, a link contains a starting node and ending node. A network is connected when there is a path between any of the nodes, otherwise the network is disconnected to graph components referred to as *islands*.

The flow network manages the topology. Nodes and links have two statuses: (i) *activated/deactivated* and (ii) *connected/disconnected*. The former status mandates whether a node or link is operational during the simulation. Events may determine this status. The latter status denotes whether a node or link is adjacent to a link or node respectively that actually determines the topology of the flow network.

Nodes and links have a (i) *flow* and (ii) *capacity*. The information type of flow and capacity is determined by a system parameter and should be information contained in the nodes and links. The flow type represents a resource distributed in the network and studied in the performed simulation. The capacity type determines which quantity can make a node or link overloaded that may result in activation/deactivation and, therefore, connection/disconnection from the topology.

The management of the topology is hidden from applications. This means that a generic interface allows adding/removing nodes and links. The update of the graph occurs in the background without programmatic effort by developers. A flow network contains general-purpose topological and graph spectra metrics such as the node degree, closeness centrality, degree centrality, degree distribution, clustering coefficient, shortest path and other. Some of this metrics are calculated with the support of the JGrapht library[4]. Metrics are logged in the background and are

[4]Available at `http://jgrapht.org` (last accessed: November 2015)

available to the researcher for post analysis. Moreover, the flow network provides the opportunity of a real-time visualization of the temporal flow network by integrating tools such as Gephi[5].

3.5. Simulation agent

The simulation agent orchestrates all other components of SFINA during system runtime. It is responsible for the execution of all scheduled events and the calculation of the network status that is the topology and the distributed flow. For each measurement type, simulation step, node and link, registered measurements by the framework or by applications are computed, stored and logged. Topological metrics computed within the flow network can also be measurements that are seamlessly performed. At the end of each simulation step, the flow and network information are exported by the file system in the respective SFINA file format.

The above agent tasks form the *active state* of the simulation agent and it is executed periodically at every simulation step. During bootstrapping, the simulation agent initializes the domain backend, loads all required information from the file system and creates the static events. It is actually the simulation agent that encodes all supported domain backends. A call for a flow analysis by an application is routed to the right domain backend for execution. A simulation agent also has a *passive state* that listens for external events received during simulation by other agents that may participate in the simulation.

3.6. Applications

A SFINA application is either a usage scenario of the simulation agent or an extension of the simulation agent with additional prototyped functionality.

In the first case, minimal or no development effort is required. A SFINA user undertakes the following actions:

1. Feeding all required input data in the file system as shown in Section 3.1.
2. Running a simulation scenario as instructed by the input data in the file system.

In the second case, development effort is put into extending or overriding the inherited functionality of a simulation agent. This may include one or more of the following actions:

1. Performing, logging and analyzing new measurements.
2. Implementing policies, models and mechanisms for flow regulation.
3. Adjusting or re-implementing existing complex core functionality of the simulation agent.

The design approach and interfaces of SFINA support the first two actions. The possibility of the third action occurring should be minimized by either adopting more effective software engineering practices at the application-level or updating the core simulation framework of SFINA to support more complex and tailored functionality.

Upgrading the SFINA framework to support a new domain backend requires the following actions:

1. Making flow input data available in the SFINA format. This can be done by either using one of the supporting conversion utilities of SFINA or by building a customized conversion tool for this purpose.
2. Making available a domain backend library with interfaces for performing flow analysis and other supported operations.

Moreover, an application logic can be split into multiple communicating agents by using the generic interfaces and modular approach of SFINA.

[5] Available at `http://gephi.github.io` (last accessed: November 2015)

4. Framework Realization and Evaluation

This paper provides an empirical proof-of-concept for SFINA by illustrating a framework realization including the following:

- The integration and evaluation of two different backends from the domain of power systems.
- The implementation of a model for cascading failures.
- The implementation of a benchmark for the performance evaluation of the model.
- The network visualization of power networks under cascading failures.
- Simulation scenarios for testing and evaluating the model of cascading failures.

Two state-of-the-art domain backends are plugged into SFINA: (i) *MATPOWER* and (ii) *InterPSS*. MATPOWER[6] is an open-source MATLAB simulation package for power flow and optimal power flow computational problems [7]. It is one of the most common tools used in the power domain. MATPOWER simulation code is procedural and has a scripting-style. In contrast, InterPSS[7] is a power system simulation software that follows the object-oriented and component-based software paradigm [8]. The goal of the integration is to perform DC and AC power flow analysis on which the model of cascading failures relies on.

The integration of MATPOWER in SFINA is achieved with the support of the Matlabcontrol API[8]. The use of MATPOWER requires the installation of MATLAB. The conversion utilities of SFINA can convert the MAT-Files of MATPOWER to the the SFINA file format. It is straightforward to integrate other MATLAB-based backends in a similar fashion, such as backends for water and gas networks [9, 10]. The integration of InterPSS is straightforward as it mainly requires the inclusion of the free Java libraries. However, performing the desired system calls to InterPSS backend can be complex. Its source code is not open and knowledge of the implemented algorithms may be required in some cases[9]. The generic interfaces of the flow analyzer hide this complexity from the SFINA users and application developers that only need to make a choice of the backend with which the simulation runs.

The model of cascading failures is implemented by reusing and extending the functionality of the simulation agent. The model captures both DC and AC power flows, it meets the generation limits and performs load-shedding to match supply and demand. Load-shedding is repeated 15 times at maximum, after which the system results in a blackout. Most operations performed are actual calls to the simulation agent. It is only the algorithmic logic that is implemented in the extended agent. In each iteration of the algorithm, power flow analysis is performed to compute the state of the cascading failure. The model implementation is totally agnostic of which domain backend, MATPOWER or InterPSS, performs the power flow analysis.

Cascading failures are evaluated in five case networks[10]: (i) *case-30*, (ii) *case-57*, (iii) *case-118*, (iv) *case-2383* and (v) *case-2736*. Two measurements quantify the impact of the cascading

[6] Available at `http://www.pserc.cornell.edu//matpower/` (last accessed: November 2015)

[7] Available at `http://www.interpss.com` (last accessed: November 2015)

[8] Available at `https://code.google.com/p/matlabcontrol/` (last accessed: November 2015)

[9] These challenges are overcome using the comprehensive tutorials of InterPSS and direct communication with the InterPSS support team.

[10] Available at `http://www.pserc.cornell.edu/matpower/docs/ref/matpower5.0/menu5.0.html` (last accessed: November 2015)

failure: (i) *line losses* and (ii) *power losses*. Line losses are the ratio of lines trimmed during the cascading failure. Power losses equal one minus the power served after the cascading failure over the power served before the cascading failure. These measurements are performed and logged within a developed benchmark agent that extends the functionality of the simulation agent. The performed measurements can be reused beyond cascading failure models. This paper focuses on simulating and quantifying the impact of cascading failures. Prevention and mitigation strategies, such as flow regulation with smart transformers [11], are out of the scope of this paper and subject of ongoing work. Experiments are performed on a MacBookAir 4.2 with 1.8 GHz CPU, 4 GB RAM and Matlab R2015a installed. A prototype for live deployments in the Brutus and Euler cluster[11] of ETH Zurich is developed as well.

The integration of Gephi in SFINA is used to visualize the impact of cascading failures. Figure 5 illustrates two power networks, the case-118 and the case-2383 before and after a cascading failure. Several events that each removes a power line, trigger the cascading failure. The links colored red indicate the removed power lines. The size of the nodes and the thickness of the lines indicates the amount of power they serve. The missing links before and after the cascading failure refer to the power lines trimmed during the cascading failure.

Figure 6 illustrates the power losses computed with MATPOWER and InterPSS when a proportion of lines are incrementally removed. Both DC and AC flow analysis is shown for each domain backend for case-30. Experiments are repeated 10 times. In Figure 6a, the removed lines are random but fixed between the performed experiments, in contrast to Figure 6b that shows the results for totally random line removals . The results of Figure 6a show that the power losses under cascading failures are on average 15.09% higher for AC compared to DC. Both MATPOWER and InterPSS compute power flow distributions with the same power losses.

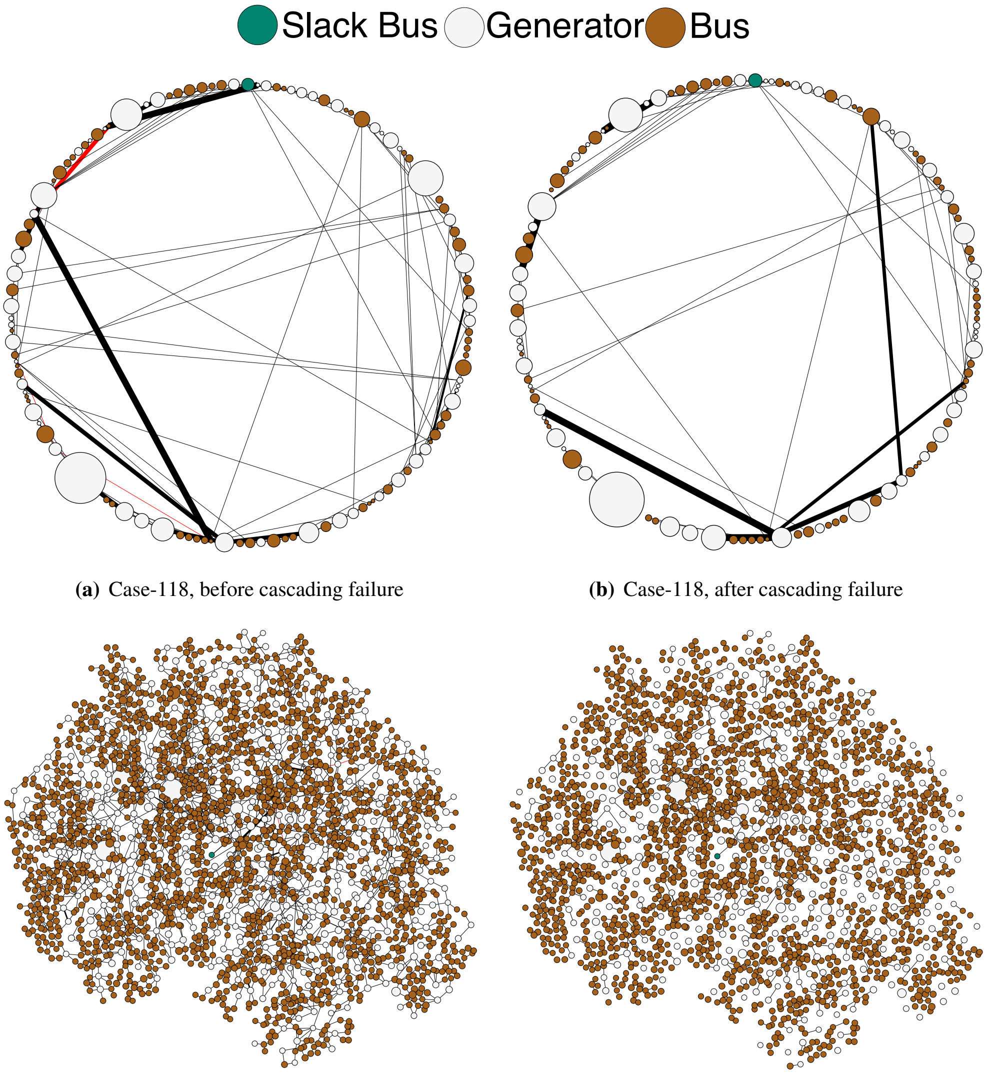
Figure 7 complements Figure 6 by showing the power losses when a proportion of the capacity in the lines is incrementally decreased. In contrast to Figure 7a, Figure 7b restores the trimmed lines at every capacity reduction. Results show the same phase transition in both cases. Both MATPOWER and InterPSS compute power flow distributions with the same power losses.

In large-scale networks, the two domain backends result in different outcomes. Figure 8a shows that for DC in case-118, MATPOWER results in 1.29% higher power losses than InterPSS. Under AC power flow, InterPSS does not converge and results in a blackout, in contrast to MATPOWER that computes average power losses of 19.5%. Case-2736 in Figure 8b cannot converge for both MATPOWER and InterPSS in AC power flow due to the violation of the generator limits. For a DC power flow analysis, both MATPOWER and InterPSS result in 46.61% of average power losses.

Figure 9 illustrates the simulation speed of SFINA with each domain backend. Overall, InterPSS is 93.71% and 88.63% faster than MATPOWER in flow analysis for case-30 and case-57. This is because of the external calls to MATLAB by the JVM. However, InterPSS has a high initialization overhead in the total runtime that is especially observable in low total runtimes such as the one of case-30. Simulation in case-30 is on average 89.13% and 72.7% faster than case-57 for MATPOWER and InterPSS respectively.

Figure 10 shows in more detail the processing overhead of SFINA under cascading failures in case-30, triggered by line removals and capacity reduction. Experiments are repeated 10 times and with random removed lines but fixed among experiments. Figure 10a shows an overhead with several fluctuation but without highly distinguished changes as the number of removed lines increases. The average number of iterations is 1.05. The average simulation time is 359.47, 542.89,

[11] Available at `http://brutuswiki.ethz.ch/brutus/Brutus_wiki` (last accessed: November 2015)



(a) Case-118, before cascading failure

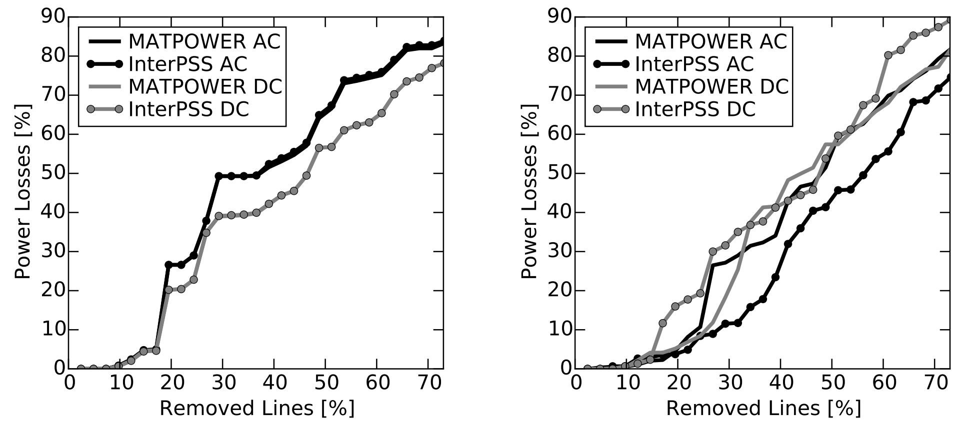
(b) Case-118, after cascading failure

(c) Case-2383, before cascading failure

(d) Case-2383, after cascading failure

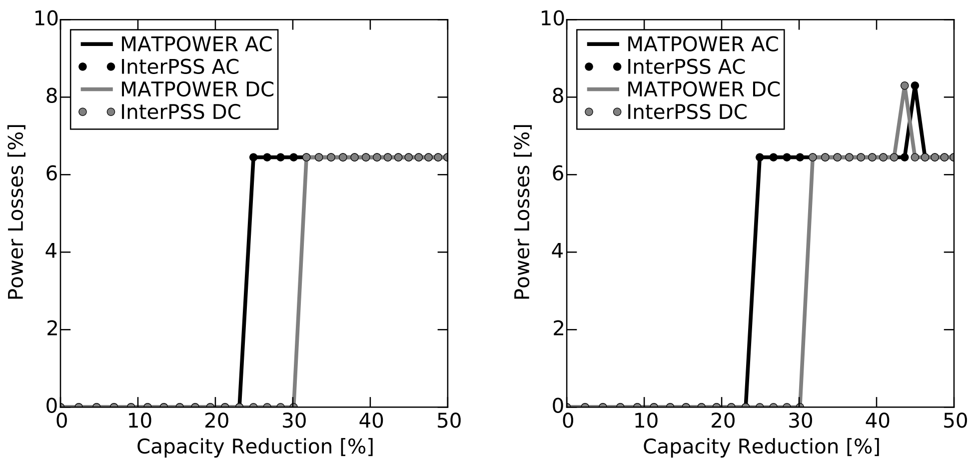
Figure 5: Gephi visualization of cascading failures. Before the cascading failure, the removed lines are marked as red. The size of the nodes and the thickness of the lines indicates the amount of power they serve. For better visual representation, line directions are removed.

287.97 and 207.41 ms for MATPOWER AC, MATPOWER DC, InterPSS AC and InterPSS DC. These results are expected given the nearly linear increase of power losses observed in Figure 6. In contrast, Figure 10b is in line with the phase transitions observed in Figure 7. When capacity reduction overpasses 19.0%, the average number of iterations increase from 1 to 4.53, resulting in an overall increase of the simulation time for both domain backends and DC/AC power flow models. The average simulation time is 542.91, 548.17, 582.97 and 320.35 ms for MATPOWER



(a) Random removed lines, fixed among experiments. **(b)** Random removed lines among experiments.

Figure 6: Power losses under cascading failures trigerred by line removals in case-30.

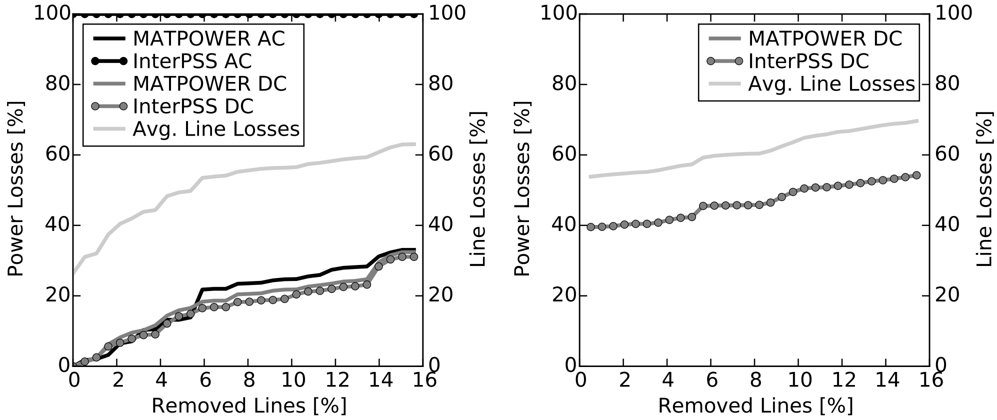


(a) Without line restoration. **(b)** With line restoration

Figure 7: Power losses under cascading failures trigerred by capacity reduction in case-30.

AC, MATPOWER DC, InterPSS AC and InterPSS DC. The respective numbers before and after phase transition are 150.56, 164.28, 225.72, 84.66 and 773.63, 751.10, 799.16, 455.41.

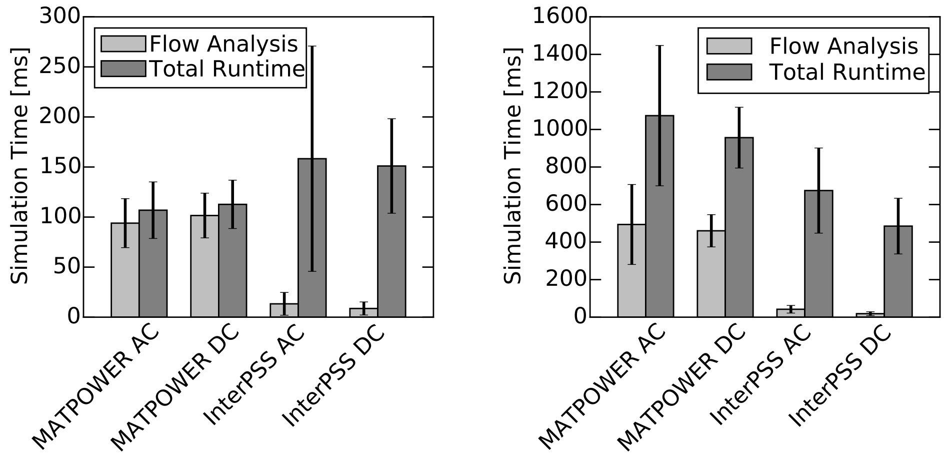
These measurements show that both MATPOWER and InterPSS communities can use their software within SFINA to study the same model of cascading failures. The model itself can be validated in more depth by inter-changing backends as illustrated in this section. The cascading failure model does not always converge for both domain backends. This is subject of discussion and further explanations in regards to the implemented flow analysis models in the two domain backends and the model of cascading failures.



(a) Case-118

(b) Case-2736

Figure 8: Power and line losses under cascading failures trigerred by line removals in larger-scale networks.



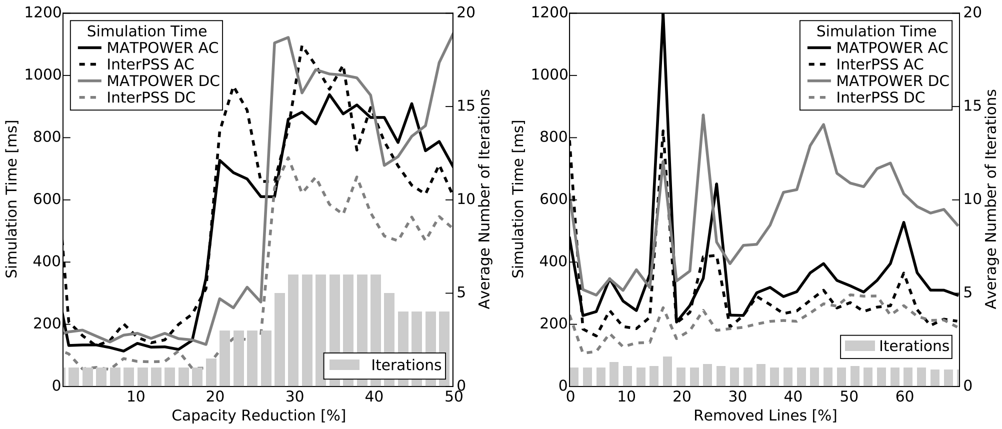
(a) Case-30

(b) Case-57

Figure 9: Simulation speed of MATPOWER and InterPSS under cascading failures with DC and AC power flows.

5. Comparison with Related Work

Some earlier work [12, 13, 14, 15] stretches the role of software technologies and standards to make simulation tools more generic and applicable in different applications. Although this paper supports this claim, technology by itself cannot tackle real-world challenges. The architecture and components of SFINA discussed in this paper are, to a high extent, technology-independent and that is how they are illustrated. It is the actual concepts, system design and component synergies that structure the proposed framework for the simulation of self-regulating techno-



(a) Cascading failure via line removals. **(b)** Cascading failure via capacity reduction.

Figure 10: Simulation speed and number of iteration under cascading failures in case-30.

socio-economic systems.

Most simulation frameworks focus on a single domain. For example, in power systems and beyond MATPOWER or InterPSS, GridSpice [16] is an open source cloud-based simulation package for the smart grid. It incorporates code from MATPOWER and GridLAB-D[12]. Its simulator wrappers allow users to plug in new tools to run separate distribution and transmission system simulations. This capability shares similarities with the support of multiple backends by SFINA, however, GridSpice focuses entirely on power systems and does not have generic semantics for flow networks. Moreover, GridSpice relies on Amazon Web Services, for a cluster deployment, in contrast to SFINA that can be deployed to any networked infrastructure. Other simulation tools for power systems include the PowerSystems[13] of Modelica. A step further move co-simulation approaches that model both power and communication networks [17, 18, 19, 20]. Simulation systems in other application domains include knowledge flows [21], evacuation systems [22] or traffic systems [23].

There is also a plethora of simulation tools that perform simulation of computer networks, physical or overlay networks [24, 25, 26, 27]. Such tools are reviewed extensively in earlier work [28]. Only a few of these tools model networks as temporal directed weighted graphs. Simpler representation of static networks are usually employed. Other generic simulation tools include agent-based ones [29, 30], social simulators [31, 32], and simulators of social networks such as Hashkat[14]. These tools focus on understanding system complexity, in contrast to SFINA that goes a step further to simulate mechanisms for the self-regulation of several techno-socio-economic systems.

Related work on cascading failures in power grids and beyond is limited to modeling instead of the modular and reconfigurable simulation of these complex phenomena [3, 33]. MATCASC [34]

[12]Available at `http://www.gridlabd.org` (last accessed: November 2015)
[13]Available at `https://github.com/modelica/PowerSystems` (last accessed: November 2015)
[14]Available at `http://hashkat.org` (last accessed: November 2015)

is a MATLAB-based tool that simulates cascading line outages in power grids. Authors focus on the simulation of cascading failures and not on their control or mitigation. They provide valuable metrics for quantifying the system robustness under cascading failures. However, MATCASC does not integrate AC power flow analysis and linearizes active power flow equations with several assumptions that may not always hold as shown in Section 4. It exclusively uses the tolerance parameter for estimating the line capacities. In contrast, SFINA does not rely on a commercial tool but instead integrates domain backends such as MATPOWER and InterPSS to provide state-of-the-art DC and AC power flow analyses with further options for solving optimal power flow problems. SFINA also simulates line capacities with both tolerance parameter and line ratings. Events can easily encode any static and dynamic line removal scenarios.

In conclusion, related work on the simulation of techno-socio-economic systems lies on the spectrum between highly flexible and generic simulation frameworks to highly customized software tools tailored to simulate system scenarios within a specific domain. The former may result in complex unintuitive framework realizations. The latter have limited practical use and applicability, especially when the nowadays techno-socio-economic systems become more inter-connected, inter-dependent and multi-perspective. SFINA bridges this gap by allowing the integration of multiple domain backends whose domain knowledge and dynamics are abstracted by dynamic flow networks represented as temporal directed weighted graphs. This network abstraction brings solid theoretical fundamental knowledge on complex networks into a highly empirical and experimental context. The applicability of regulation models, policies and mechanisms, such as preventing or mitigating cascading failures, can be simulated and evaluated within the proposed framework. SFINA does not aim to replace existing simulation tools, rather to form a unifying umbrella over a significant but highly fragmented work on the simulation of techno-socio-economic systems.

6. Conclusion and Future Work

This paper concludes that SFINA is a modular, reconfigurable and scalable simulation framework capable of prototyping online decentralized regulation for techno-socio-economic systems. Experimental evaluation of two domain backends in a challenging computational scenario of cascading failures in power grids indicates the benefits in the design approach of SFINA. Rather than coming to replace existing simulation tools, SFINA aims at minimizing the fragmentation and discrepancies between simulation communities. Its ultimate objective is to respond to nowadays challenges on how to regulate highly inter-connected and inter-dependent techno-socio-economic systems as a result of the pervasive ICT technologies in several societal sectors.

The further support of other domain backends, the showcase of application scenarios and the execution of SFINA in real-world distributed networks are part of future work. Moreover, compliance to interoperability standards [13] and open data formats are subject of ongoing work. Synergies with several simulation communities are a priority for the wide adoption of SFINA.

Acknowledgments

This research is part of the SFINA project funded by the Professorship of Computational Social Science, ETH Zurich, Zurich, Switzerland. The authors would like to thank the support team of InterPSS for their tips in using InterPSS.

[1] S. V. Buldyrev, R. Parshani, G. Paul, H. E. Stanley, S. Havlin, Catastrophic cascade of failures in interdependent networks, Nature 464 (7291) (2010) 1025–1028.

[2] E. Pournaras, Multi-level reconfigurable self-organization in overlay services, Ph.D. thesis, Delft University of Technology (March 2013).
[3] M. Papic, K. Bell, Y. Chen, I. Dobson, L. Fonte, E. Haq, P. Hines, D. Kirschen, X. Luo, S. Miller, N. Samaan, M. Vaiman, M. Varghese, P. Zhang, Survey of tools for risk assessment of cascading outages, in: Power and Energy Society General Meeting, 2011 IEEE, 2011, pp. 1–9.
[4] E. Pournaras, M. Vasirani, R. Kooij, K. Aberer, Decentralized planning of energy demand for the management of robustness and discomfort, Industrial Informatics, IEEE Transactions on 10 (4) (2014) 2280–2289.
[5] G. Sakellari, G. Loukas, A survey of mathematical models, simulation approaches and testbeds used for research in cloud computing, Simulation Modelling Practice and Theory 39 (2013) 92–103.
[6] W. Galuba, K. Aberer, Z. Despotovic, W. Kellerer, Protopeer: a p2p toolkit bridging the gap between simulation and live deployement, in: Proceedings of the 2nd International Conference on Simulation Tools and Techniques, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009, p. 60.
[7] R. D. Zimmerman, C. E. Murillo-Sánchez, R. J. Thomas, Matpower's extensible optimal power flow architecture, in: Power & Energy Society General Meeting, 2009. PES'09. IEEE, IEEE, 2009, pp. 1–7.
[8] M. Zhou, S. Zhou, Internet, open-source and power system simulation, in: Power Engineering Society General Meeting, 2007. IEEE, 2007, pp. 1–5.
[9] L. MatijaÅevi, I. Dejanovi, D. Spoja, A water network optimization using matlaba case study, Resources, Conservation and Recycling 54 (12) (2010) 1362 – 1367.
[10] T. Koch, B. Hiller, M. E. Pfetsch, L. Schewe, Evaluating gas network capacities, Vol. 21, SIAM, 2015.
[11] E. Pournaras, M. Yao, R. Ambrosio, M. Warnier, Organizational Control Reconfigurations for a Robust Smart Power Grid, in: Internet of Things and Inter-cooperative Computational Technologies for Collective Intelligence, Vol. 460 of Studies in Computational Intelligence, Springer-Verlag, 2012, Ch. 8, pp. 189–206.
[12] K. Al-Zoubi, G. Wainer, Rise: A general simulation interoperability middleware container, Journal of Parallel and Distributed Computing 73 (5) (2013) 580 – 594.
[13] K. Morse, M. Lightner, R. Little, B. Lutz, R. Scrudder, Enabling simulation interoperability, Computer 39 (1) (2006) 115–117.
[14] A. Tolk, J. A. Muguira, The levels of conceptual interoperability model, in: Proceedings of the 2003 Fall Simulation Interoperability Workshop, Vol. 7, Citeseer, 2003.
[15] A. Tolk, J. Pullen, Using web services and data mediation/storage services to enable command and control to simulation interoperability, in: Distributed Simulation and Real-Time Applications, 2005. DS-RT 2005 Proceedings. Ninth IEEE International Symposium on, 2005, pp. 27–34.
[16] K. Anderson, J. Du, A. Narayan, A. El Gamal, Gridspice: A distributed simulation platform for the smart grid, Industrial Informatics, IEEE Transactions on 10 (4) (2014) 2354–2363.
[17] H. Lin, S. Sambamoorthy, S. Shukla, J. Thorp, L. Mili, Power system and communication network co-simulation for smart grid applications, in: Innovative Smart Grid Technologies (ISGT), 2011 IEEE PES, 2011, pp. 1–6.
[18] H. Lin, S. Veda, S. Shukla, L. Mili, J. Thorp, Geco: Global event-driven co-simulation framework for interconnected power system and communication network, Smart Grid, IEEE Transactions on 3 (3) (2012) 1444–1456.
[19] V. Liberatore, A. Al-Hammouri, Smart grid communication and co-simulation, in: Energytech, 2011 IEEE, 2011, pp. 1–5.
[20] E. etinkaya, D. Broyles, A. Dandekar, S. Srinivasan, J. Sterbenz, Modelling communication network challenges for future internet resilience, survivability, and disruption tolerance: a simulation-based approach, Telecommunication Systems 52 (2) (2013) 751–766.
[21] H. Zhuge, Knowledge flow network planning and simulation, Decision Support Systems 42 (2) (2006) 571 – 592.
[22] N. Dimakis, A. Filippoupolitis, E. Gelenbe, Distributed building evacuation simulator for smart emergency management, The Computer Journal (2010) bxq012.
[23] Q. Yang, H. N. Koutsopoulos, A microscopic traffic simulator for evaluation of dynamic traffic management systems, Transportation Research Part C: Emerging Technologies 4 (3) (1996) 113–129.
[24] H. Casanova, A. Legrand, M. Quinson, Simgrid: A generic framework for large-scale distributed experiments, in: Computer Modeling and Simulation, 2008. UKSIM 2008. Tenth International Conference on, 2008, pp. 126–131.
[25] I. Baumgart, B. Heep, S. Krause, Oversim: A flexible overlay network simulation framework, in: IEEE Global Internet Symposium, 2007, IEEE, 2007, pp. 79–84.
[26] A. Montresor, M. Jelasity, Peersim: A scalable p2p simulator, in: Peer-to-Peer Computing, 2009. P2P'09. IEEE Ninth International Conference on, IEEE, 2009, pp. 99–100.
[27] G. Riley, T. Henderson, The ns-3 network simulator, in: K. Wehrle, M. GneÅ, J. Gross (Eds.), Modeling and Tools for Network Simulation, Springer Berlin Heidelberg, 2010, pp. 15–34.
[28] L. Breslau, D. Estrin, H. Yu, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, et al., Advances in network simulation, Computer (5) (2000) 59–67.
[29] S. F. Railsback, S. L. Lytinen, S. K. Jackson, Agent-based simulation platforms: Review and development recommendations, Simulation 82 (9) (2006) 609–623.

[30] O. Obst, M. Rollmann, Spark a generic simulator for physical multi-agent simulations, in: G. Lindemann, J. Denzinger, I. Timm, R. Unland (Eds.), Multiagent System Technologies, Vol. 3187 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2004, pp. 243–257.

[31] P. Terna, Simulation tools for social scientists: Building agent based models with swarm, Journal of artificial societies and social simulation 1 (2) (1998) 1–12.

[32] R. Suleiman, K. G. Troitzsch, N. Gilbert, Tools and techniques for social science simulation, Springer Science & Business Media, 2012.

[33] D. Hirshleifer, S. Hong Teoh, Herd behaviour and cascading in capital markets: A review and synthesis, European Financial Management 9 (1) (2003) 25–66.

[34] Y. Koç, T. Verma, N. Araujo, M. Warnier, et al., Matcasc: A tool to analyse cascading line outages in power grids, in: Intelligent Energy Systems (IWIES), 2013 IEEE International Workshop on, IEEE, 2013, pp. 143–148.