Prototyping Self-managed Interdependent Networks – Self-healing Synergies against Cascading Failures

Evangelos Pournaras, Mark Ballandies, Dinesh Acharya, Manish Thapa and Ben-Elias Brandt
Professorship of Computational Social Science
ETH Zurich, Zurich, Switzerland
{epournaras,bmark,acharyad,manish.thapa,bbrandt}@ethz.ch

ABSTRACT

The interconnection of networks between several techno-socio-economic sectors such as energy, transport, and communication, questions the manageability and resilience of the digital society. System interdependencies alter the fundamental dynamics that govern isolated systems, which can unexpectedly trigger catastrophic instabilities such as cascading failures. This paper envisions a general-purpose, yet simple prototyping of self-management software systems that can turn system interdependencies from a cause of instability to an opportunity for higher resilience. Such prototyping proves to be challenging given the highly interdisciplinary scope of interdependent networks. Different system dynamics and organizational constraints such as the distributed nature of interdependent networks or the autonomy and authority of system operators over their controlled infrastructure perplex the design for a general prototyping approach, which earlier work has not yet addressed. This paper contributes such a modular design solution implemented as an open source software extension of SFINA, the *Simulation Framework for Intelligent Network Adaptations*. The applicability of the software artifact is demonstrated with the introduction of a novel self-healing mechanism for interdependent power networks, which optimizes power flow exchanges between a damaged and a healer network to mitigate power cascading failures. Results show a significant decrease in the damage spread by self-healing synergies, while the degree of interconnectivity between the power networks indicates a tradeoff between links survivability and load served. The contributions of this paper aspire to bring closer several research communities working on modeling and simulation of different domains with an economic and societal impact on the resilience of real-world interdependent networks.

CCS CONCEPTS

• **Networks** → **Network properties**; **Network structure**; **Network reliability**; **Network manageability**; **Cyber-physical networks**; • **Computing methodologies** → **Modeling and simulation**; • **Software and its engineering** → *Object oriented frameworks*; *Integrated and visual development environments*; *Software libraries and repositories*; *Software prototyping*; *Reusability*;

KEYWORDS

self-management, self-healing, modeling, simulation, interdependent networks, multiplex networks, cascading failure, distributed system, smart grid

ACM Reference Format:
Evangelos Pournaras, Mark Ballandies, Dinesh Acharya, Manish Thapa and Ben-Elias Brandt. 2018. Prototyping Self-managed Interdependent Networks – Self-healing Synergies against Cascading Failures. In *SEAMS '18: SEAMS '18: 13th International Symposium on Software Engineering for Adaptive and Self-Managing Systems , May 28–29, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3194133.3194148

1 INTRODUCTION

The increasing interconnectivity of digital techno-socio-economic systems puts at risk the resilience of society [14]: power networks rely on communication networks for their control operations, while communication networks are energized by the former ones and they are vulnerable to cyber-attacks. Transport networks support maintenance operations of power and water networks. The latter ones though feed power generation turbines. All these interdependencies challenge the manageability of unexpected events, e.g. system failures. Their dynamics over interdependent and multiplex[1] networks are fundamentally different and more complex than the ones of isolated networks [1, 21]: cascading failures can coevolve and spread from one network to the other causing unforeseen economic damages and social unrest of an unprecedented impact [2, 14].

This paper studies how to design a domain-independent software prototyping toolkit for modeling and simulation of self-management systems that serve interdependent networks. In this context, a self-management system can be a coordination mechanism that regulates the flow exchange between interdependent networks such that the utility of both networks is maximized. For instance, self-healing of two interdependent power networks can be the result of managing a smart exchange of power flows.

[1]Multiplex networks share the same nodes in different layers [12], whereas interdependent networks allow connections between the nodes of different networks. The latter provides a more general reference model in the context of this paper. Therefore, for the sake of simplicity, the rest of this paper refers to the studied networks as interdependent.

SEAMS '18, May 28–29, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5715-9/18/05…$15.00
https://doi.org/10.1145/3194133.3194148

Motivated by the limitations of related work [9, 31] reviewed in this paper, the set of requirements for prototyping self-management systems are the following: System dynamics from several application domains are supported in a modular and extensible way. A self-management mechanism is implemented once and applied in different networks of the same or different type. Both a micro (bottom-up) and macro (top-down) level design are supported, meaning the flexibility to model individual network components as well as the network at an operational level. The simulation design is aligned with the distributed nature of interdependent networks observed in reality [41], meaning that networks remain autonomous entities under the authority of their system operators, while coordination is performed via distributed computational models, which are the actual realizations of self-management mechanisms. Finally, such a toolkit should be designed for all involved communities and as such, impact requires an open source software artifact accompanied with documentation, a graphical user interface (GUI) as well as demonstration and application scenarios.

This paper introduces a novel software extension of SFINA, the *Simulation Framework for Intelligent Network Adaptations* [35], that meets the aforementioned requirements. The applicability of the framework to prototype a novel self-healing mechanism for interdependent power networks that undergo cascading failures is evaluated experimentally. The self-healing process performs load reduction in the damaged network. The reduced load can be transferred via an interlink to the second healer network that plays the role of a 'reservoir'. The damaged network can recover the tranferred load from the healer network using a different interlink to redistribute more efficiently the power flow. Given the challenge of computing analytical equations for cascading failures under nonliner AC power flows, this process is modeled as a particle swarm optimization, inspired by a battery model of earlier work [29]. The optimization can be used for system analysis, operational planning and, in some cases, for online regulation. An $m-1$ contingency analysis [20, 22, 39] with an AC power flow distribution over four reference power networks shows that self-healing decreases the damage spread, power cascade and damage correlation, with the former and latter ones showing a significant decrease. Moreover, by increasing the interdependencies, i.e. the number of interlinks between the power networks, the links survivability decreases, while the served load increases.

The contributions of this paper are outlined as follows:

(1) A generic model to prototype self-management systems for interdependent and multiplex networks.
(2) The extension of the SFINA simulation framework with a modular implementation of the proposed model.
(3) An open source documented software artifact for use by a broad range of communities.
(4) A novel self-healing mechanism against cascading failures in interdependent power networks.
(5) An overview of the literature on modeling and simulation for interdependent networks.

This paper is organized as follows: Section 2 introduces a modeling and simulation approach to prototype self-management systems for interdependent networks. Section 3 illustrates the software engineering of this approach as an extension of the SFINA simulation framework. Section 4 shows how interdependent power networks can self-heal from cascading failures and Section 4 illustrates the experimental evaluation of this case study. Section 6 positions this work to a broader context and makes qualitative comparisons with other modeling and simulation tools reviewed in literature. Finally, Section 7 concludes this paper and outlines future work.

2 SELF-MANAGEMENT OF INTERDEPENDENT NETWORKS

This paper studies *flow networks* as *temporal directed weighted graphs* that model at a fine-grained level several complex infrastructural networks such as water/gas networks, power grids, airline, transport and telecommunication networks. *Temporal* networks [16] model operational changes that rapidly occur in the aforementioned systems. *Directed* networks denote the exact distribution of the flows, while *weighted* networks provide information about the flow values as well as information about the physical characteristics of the nodes and links, i.e. capacities. Flows may represent electrical power, information exchanged or moving vehicles on roads. The domain dynamics of a flow network govern how flows are distributed in the network, for instance the dynamics of the Kirchoff's physical law govern the distribution of power flows over power lines. Optimization [43] and multi-agent simulations [17] are common techniques to compute network flow distributions.

Under uncertainties and perturbations, for instance infrastructural cyber-attacks, maintenance operations or extreme weather events, flow distribution can be disturbed resulting in high costs, low quality of service and infrastructural damages [27]. This is the case for power blackouts or traffic congestion. System operators can apply adaptive healing and repair strategies to prevent or mitigate such a system unrest. The design and prototyping of automated, adaptive and online self-management mechanisms is the focus of this paper. More specifically, this paper studies how software can be engineered for such mechanisms when flow networks of the same and/or different types become interdependent. In principle, exchanging flows between two or more stable interdependent networks plays the role of a negative externality and can cause network instability and disturbances. The goal of a self-management mechanism for inter-dependent networks is to turn these negative externalities into positive ones using distributed computational intelligence. This rationale is inspired by earlier evidence for the improvement of system reliability using an optimal interconnectivity level [2]. For instance, when a power grid undergoes a cascading failure, power lines become overloaded and the served load in the network is reduced by the blackout. Interconnecting an overloaded node with a neighboring power grid to export power, while the excess power in the second network is imported back at nodes with no flows due to the blackout, is an example of a self-healing action that a self-management mechanism can perform.

Figure 1 illustrates the proposed modeling and simulation approach to prototype self-management mechanisms for interdependent networks. Physical networks are considered as autonomous entities, which is usually the case in reality: System operators manage their network infrastrure independently for security, safety and economic reasons [41]. For instance, the system operators of a railway network infrastructure rely on the system operators of

the power grid to energize their infrastructure, however, railway operators do not have direct control over the physical assets of the power network and as a result coordination relies on manual sharing of operational plans and schedules. This is also a common practice for networks of the same type. For instance, national power grids run their own nodal or zonal power markets [4], while they can exchange power via special power lines referred to as *flow gates* to reduce production costs, penetrate renewables and improve reliability, i.e. operating reserves [23]. Policies such as the elimination of nuclear power production in Germany may require an increase of electricity input from France, which raises concerns[2] about the outsourcing of future energy security in Germany.

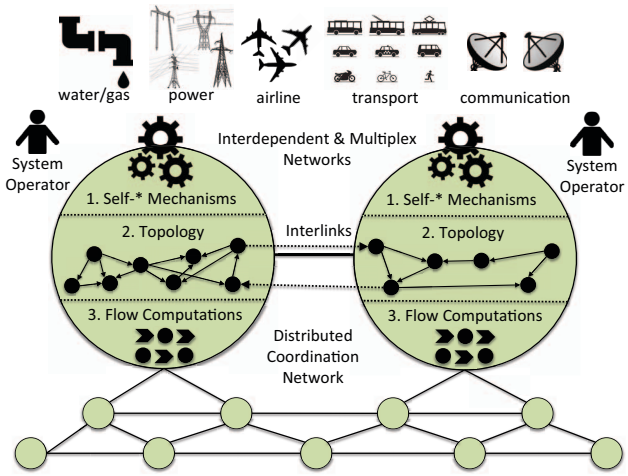


Figure 1: Self-management of interdependent networks using a distributed coordination network. This paper contributes a modeling and simulation toolkit for evaluation and prototyping of self-management systems.

Given these constraints and challenges, the proposed model introduces a *distributed coordination network* over which system operators can prototype automated self-management mechanisms for their interdependent infrastructural networks. In this coordination network, the *nodes* are the system operators, who have under their authority the control operations of their infrastructural network. The *links* are communication channels that manage the underlying inter-dependencies between their infrastructural networks. The topology of the coordination network represents the overall operational interdependencies and as such the coordination topology provides a holistic modeling and simulation approach for highly complex multifaceted application domains, for instance, smart cities.

Note that in contrast to the actual *physical* flow networks that reside under each node of the distributed coordination network, the coordination network is a *virtual* (logical) network realized by a software artifact that this paper contributes. The software can run in distributed computational resources of system operators and communication can be performed over the Internet or another communication infrastructure dedicated for this purpose.

3 A SOFTWARE ARTIFACT FOR INTERDEPENDENT NETWORKS

This section illustrates the SFINA framework as well as its software artifact extension that supports the prototyping of self-management mechanisms for interdependent networks.

3.1 The SFINA framework

The proposed model is implemented within SFINA[3], the *Simulation Framework for Intelligent Network Adaptations* [35]. SFINA is a novel modeling and simulation framework that supports the prototyping of domain-independent regulatory and management systems for flow networks. SFINA is an open source software implemented in Java under the GPL-2.0 license and is accompanied with a GUI[4].

By using SFINA, a system developer along with system operators and policy-makers can prototype a self-management system as a SFINA application once and evaluate it in different domains, i.e. power/gas/water/transport networks, without changing the application logic. This is achieved during the modeling process by separating the flow dynamics from the network topology. The computed flows over temporal directed weighted graphs can govern different laws. SFINA is designed to minimize the fragmentation and discrepancies between different simulation communities by allowing the interoperability of SFINA with several other existing domain backends. Figure 2 provides an overview of the SFINA open source software backbone supported at this project phase.

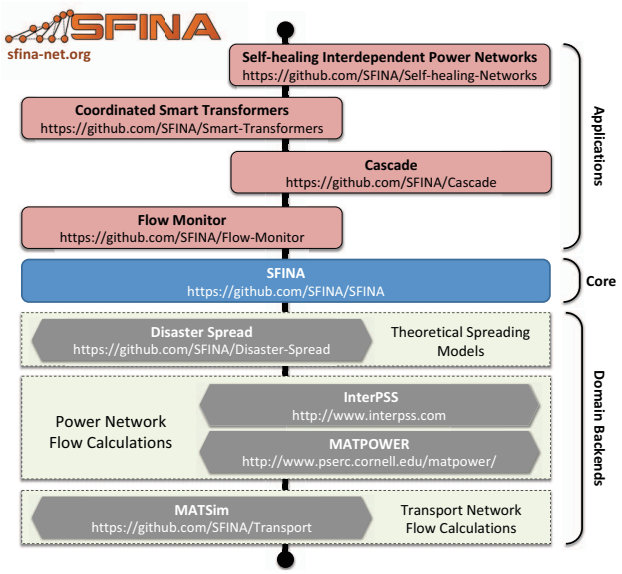


Figure 2: The current SFINA software backbone.

[2] Available at https://breakingenergy.com/2015/06/12/is-germany-outsourcing-its-future-energy-security/ (last access: March 2018)

[3] Avalaible at http://sfina-net.org and https://github.com/SFINA/ (last access: March 2018)

[4] Avalaible athttps://github.com/SFINA/GUI (last access: March 2018)

SFINA supports four *domain backends* for network flow calculations: (i) a theoretical disaster spread model [3], (ii) the MATPOWER [43] and (iii) the InterPSS [42] simulation packages for power networks as well as (iv) the MATSim[5] [17] framework for agent-based transport simulations. Implemented *applications* include (i) general-purpose monitoring and measurements of the network topology and flow [38], (ii) simulation of cascading failures in power grids [35], (iii) coordinated smart transformers for self-repairable smart grids [36] and (iv) self-healing interdependent power networks that is a contribution of this paper.

SFINA is built on top of Protopeer [11], a distributed prototyping toolkit for the simulation and live deployment of distributed applications. Protopeer introduces two main entities, the Peer and the Peerlet. Peers can communicate with each other by exchanging messages and contain one or more peerlets that encapsulate the application logic. SFINA is written within peerlets, i.e. ComputationalAgent and TimingAgent, with every network modeled by one peer. Therefore, SFINA inherits all the functionality of the peers such as timing, logging, communication as well as the capability to make parallel and networked flow computations by running multiple peers in a live experiment. This distributed modeling approach provides the flexibility to extend the SFINA functionality to support interdependent networks as shown in Section 3.2.

Figure 3 illustrates how SFINA manages time. System runtime begins with the *bootstrapping time* during which SFINA performs initialization of the peers. The *simulation time* follows: Every peer executes *simulation steps* during which a set of network flow computations are performed. Simulation steps are aligned with the clock of the peers, therefore, in the *simulation mode* of Protopeer, a global clock progresses the simulation steps, whereas in the *live mode* every peer has its own clock and time progresses asynchronously. Within a simulation step, several algorithmic *iterations* of network flow recomputations are performed as a result of processed *events*. For instance, a power cascading failure can be simulated by an iterative algorithm that recomputes flow redistributions as a result of power lines failing when exceed their capacity. In contrast to the simulation steps that are fixed and managed by the SFINA core, iterations are a result of application logic.

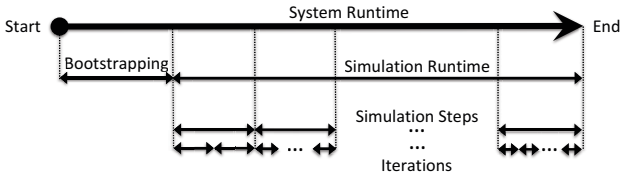


Figure 3: Time management in the SFINA framework.

More information about the SFINA architecture is available in earlier work [35] and in documentation[6] for application and core developers.

[5] Available at http://www.matsim.org (last access: March 2018)
[6] Avalaible at https://github.com/SFINA/Manual (last access: March 2018)

3.2 Software extension for interdependent networks

Figure 4 illustrates the UML system design of the interdependent networks model illustrated in Section 2. The model is designed as a layer extension of the SFINA framework, therefore, all earlier SFINA functionality is fully supported without introducing any changes on how single-network SFINA applications are developed. The topology and flow information of the interlinks is stored in separate files. For the overall higher modularity and reusability of the SFINA framework, the primary SimulationAgent is split into the ComputationalAgent and the TimingAgent, which in synergy perform all the earlier operations of the SimulationAgent [35].

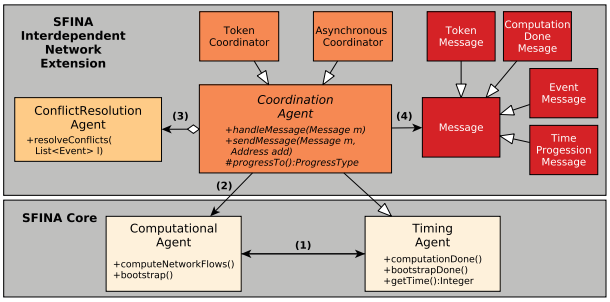


Figure 4: The UML system design of the interdependent networks model implemented in the context of the SFINA framework.

The components of the SFINA extension for interdependent networks are outlined as follows:

- CoordinationAgent: It resides on each peer, which has a network that shares interlinks with networks of other peers. It communicates with these other peers by exchanging messages that orchestrate the time progression and flow computations of the simulation. The CoordinationAgent extends the peerlet TimingAgent and notifies the ComputationalAgent when to progress the next simulation step. The CoordinationAgent is an abstract class implemented according to the template method pattern to support different coordination mechanisms given the application scenarios and protocols to which system operators agree to adhere in real-world scenarios. The software artifact supports the following coordination mechanisms at this stage:
  - TokenCoordinator: The interdependent networks perform their operations in a cyclical fashion by passing a token to the next peer ID.
  - AsynchronousCoordinator: The interdependent networks compute their operations in parallel by either simulating network latencies via the respective Protopeer interfaces [11] or by deploying Protopeer in live mode.
- ConflictResolutionAgent: It resolves conflicting events for the CoordinationAgent, for instance, competing flow exchanges over an interlink. Conflicts resolution is performed in a deterministic way by implementing rules.

- Message: The CoordinationAgent realizes coordination via messages extending the Protopeer Message. New message types can be defined for coordination functionality beyond the one in the TokenCoordinator and the AsynchronousCoordinator. Any basic coordination functionality relies on the following three messages, while the TokenCoordinator additionally uses the TokenMessage:
  - ComputationDoneMessage: A peer with a network notifies another peer with which interlinks are shared that its flow computations are complete, i.e. iteration ended.
  - TimeProgressionMessage: A peer with a network notifies another peer that its ComputationalAgent is progressed to the next simulation step, i.e. all iterations are complete.
  - EventMessage: A peer with a network notifies another peer with which interlinks are shared about events triggered via their interlinks.
  - TokenMessage: This message is passed by a TokenCoordinator to the TokenCoordinator of another peer when the ComputationalAgent completes its computations.

The interactions between the involved components in the SFINA extension are the following (Figure 4):

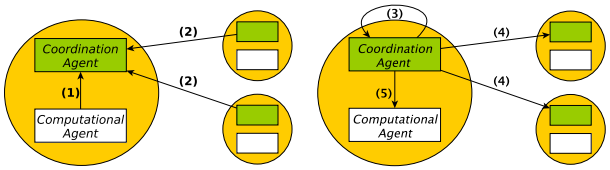
(1) The ComputationalAgent follows the observer design patter and when it completes its computations, it calls the computationDone() method of the CoordinationAgent inherited from the TimingAgent.
(2) The CoordinationAgent adopts the state design pattern to determine via progressTo() the next action (ProgressType) of the ComputationalAgent: (i) initiates another iteration of computeNetworkFlows(), (ii) advances to the next simulation step or (iii) remains idle.
(3) The CoordinationAgent has a ConflictResolutionAgent to resolve event conflicts by calling the method resolveConficts(List<Event> l).
(4) The two extensions of the CoordinationAgent, namely the TokenCoordinator and AsynchronousCoordinator, make use of the extended Message classes to orchestrate the network flow computations between peers.

The process flow lifecycle of the CoordinationAgent is illustrated in Figure 5. During the *sensing* phase, the CoordinationAgent is called by the ComputationalAgent and the CoordinationAgent of other peers. In the *actuating* phase, the CoordinationAgent determines the progress of the iteration or simulation step, triggers the next network flow computations and communicates with the CoordinationAgent of the other peers.

The coordinated time progress of the simulation is performed by the TokenCoordinator or the AsynchronousCoordinator that extend the CoordinationAgent and overwrite the progessTo() method. Both coordinators implement the coordination logic for progressing simulation time in the readyToProgress() method. Algorithm 1 illustrates this logic for the TokenCoordinator.

Progress to the next iteration is performed when the network flow computations do not converge (line 2-3 of Algorithm 1), e.g. a power network undergoes cascading failures or new events from interdependent networks are received. Progress to the next simulation step is performed when the network flow computations are complete, i.e. no more iterations are required, and the TokenCoordinator holding the token resides on the leading peer (line 5-7 of Algorithm 1). Iterations are skipped when computations in other networks are in progress (line 11-12 of Algorithm 1). The token is passed to the next peer after computations are complete (line 8-9 of Algorithm 1).

Algorithm 2 illustrates the coordination logic of the AsynchronousCoordinator. The logic of the simulation progress requires completion of all network flow computations (line 1 of Algorithm 2). The criteria for progression to the next iteration (line 2-3 of Algorithm 2) or simulation step (line 5-6 of Algorithm 2) are the same with the ones of the TokenCoordinator.



(a) Sensing (b) Actuating

Figure 5: The process flow lifecycle of the CoordinationAgent to progress simulation. All peers are equipped with a ComputationalAgent as well as a CoordinationAgent in case they share interlinks with networks in other peers. (1) The ComputationalAgent calls computationDone(). (2) Incoming coordination messages may be received by peers with which interlinks are shared. (3) The CoordinationAgent determines the simulation progress (ProgressType) by calling progressTo(). (4) Outgoing coordination messages are sent to peers with which interlinks are shared. (5) In the next simulation step or iteration, the network flows are computed.

Algorithm 1 The logic of readyToProgress() in the TokenCoordinator.

```
1:  if has token then
2:      if network flow computations do not converge then
3:          progress to next iteration
4:      else
5:          if other networks are converged then
6:              if is first network then
7:                  progress to next simulation step
8:              else
9:                  send token to next network
10:             end if
11:         else
12:             skip next iteration
13:         end if
14:     end if
15: else
16:     do nothing
17: end if
```

Algorithm 2 The logic of readyToProgress() in the AsynchronousCoordinator.

```
1:  if other network flow computations are complete then
2:    if network flow computations do not converge then
3:      progress to next iteration
4:    else
5:      if other networks are converged then
6:        progress to next simulation step
7:      else
8:        skip next iteration
9:      end if
10:   end if
11: else
12:   do nothing
13: end if
```

The key difference between the TokenCoordinator and the AsynchronousCoordinator is the order of the network flow computations among the interdependent networks. The TokenCoordinator introduces a sequential order, whereas the AsyncronousCoordinator does not define an order, which can be a result of different asynchronous clocks in each peer i.e. distributed simulation with the live Protopeer deployment. Visualization of interdependent networks is supported by the SFINA GUI as shown in Figure 6.

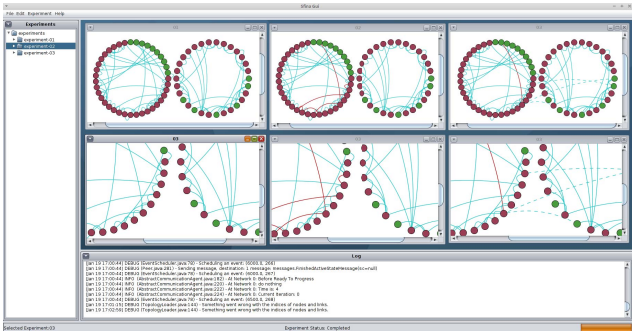


Figure 6: Visualization of interdependent networks via the SFINA GUI.

A demonstration of the software artifact is made available[7] to the community. It contains a user guide, visualizations, an executable as well as a virtual machine. The demonstration concerns the simulation of cascading failures over self-healing interdependent power networks, which is the case study illustrated in Section 4.

4 SELF-HEALING INTERDEPENDENT POWER NETWORKS

This section introduces a case study on interdependent power networks to demonstrate the proof of concept and usability of the SFINA extension for prototyping and analysis of self-management systems. A self-healing mechanism is introduced to mitigate cascading failures in a *damaged* power transmission network, while using a second *healer* network as an operating reserve, from which power flow can be received or sent. In other words, the healer network acts as a 'battery' reserve. Such a coordination scenario is applicable in real-world, e.g. multi-area optimal power flows [41], though it is studied here in the new context of cascading failures. Moreover, exploring the mitigation potential of interdependent power networks against cascading failures does not require additional costly infrastructure in the individual networks, in constrast to using smart transformers [36] or backup generators [25].

A cascading failure is typically a result of power flow redistribution over parallel lines when one of these parallel lines is trimmed by an event such as a cyber-attack or a system failure. The redistribution is a result of the physical laws that govern power systems. The lines that overtake the flow redistribution are likely to become overloaded and trimmed as well causing a further power flow accumulation and spread of the damage. A cascading failure can split the network to several disconnected regions referred to as *islands*.

The self-healing mechanism is positioned within an earlier model[8] [36] of cascading failures that computes the AC power flows of damaged islands and performs various standardized load shedding[9] strategies [32]. The self-healing mechanism operates in each island and performs an automated (i) *load reduction* and (ii) *load recovery* without requiring manual interventions by system operators. Load reduction is performed either by a load transfer to the healer network if there is an available interlink in the island or by load shedding. A combination of load transfer and load shedding can be performed as well. The self-healing mechanism compensates this load reduction in the damaged network by recovering load from the healer network via the interlinks. Computationally, this power flow exchange is modeled as a particle swarm optimization problem and it is applied to every potential island caused by the cascading failure. The optimizer maximizes load-reduction when alternative load can be restored from the healer network under the constraint[10] that no damage is introduced in the healer network.

The proposed self-healing model can be used for a cooperative operational planning between the operators of interdependent power networks, reliability post-analysis as well as online regulation of interdependent power flows. The latter scenario requires a communication infrastructure and computational resources to meet an almost real-time control. Such opportunities are addressed in related work [13, 19, 40]. Earlier performance measurements of the SFINA simulation framework confirm the computational feasibility for an online applicability [35, 36].

4.1 Self-healing measurements

Self-healing over interdependent networks is evaluated with the following customized metrics of a general mathematical framework for measuring cascading failures [38]: (i) *damage spread*, (ii) *power cascade* and (iii) *damage correlation*.

[7] Available at http://sfina-net.org/shared/SFINA.zip (last access: March 2018).

[8] The Algorithm 1 of the earlier work [36] outlines the exact AC power flow computations performed for the simulation of cascading failures. The implementation of the algorithm is part of the SFINA application 'coordinated smart transformers' as shown in Figure 2. The self-healing mechanism is positioned within this algorithm.

[9] Load shedding is a limited disruption of serving electrical power to consumers. It is actually the result of a blackout and it is performed in a controlled way by system operators to prevent infrastructural damages and the further spread of the blackout.

[10] This is feasible by generation curtailment [10, 18] or by compensating low power generation from renewables [28].

The damage spread concerns the probability of decreasing the *links survivability* over the iterations of the cascading failure. Given the number of survived links $\hat{m}_i$ after the removal of link i and the total links m, where $\hat{m}_i \leq m$, the links survivability is defined as $\frac{\hat{m}_i}{m}$. The damage spread d_t at iteration t is defined as follows:

$$d_t = \sum_x F_t(x)\Delta x - \sum_x F_{t-1}(x)\Delta x, \quad (1)$$

where x represents the links survivability and Δx is the integration step of the cumulative distribution function $F_t(x)$ over the iterations of the cascading failure.

The power cascade c_t concerns the probability of progressing the iteration of the cascading failure. It is defined as follows:

$$c_t = \frac{1}{m}\sum_{i=1}^{m} p_{i,t}, \quad (2)$$

where:

$$p_{i,t} = \begin{cases} 1 & \text{if cascade progresses to } t \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

represents the progress or not of the cascade to the next iteration after removal of link i.

The damage correlation $r_t(\mathbf{s}_{i,t}, \mathbf{s}_{j,t})$ measures the Pearson correlation coefficient between the status of the links $\mathbf{s}_{i,t}$ and $\mathbf{s}_{j,t}$ after the removal of link i and j respectively, where $\forall u \in \{1, ..., |\mathbf{s}_{i,t}|\}$, $s_{i,t,u} = 0$ for failure or 1 for survival.

5 EXPERIMENTAL EVALUATION

This section illustrates the implementation choices, the experimental settings as well as the evaluation results of the self-healing mechanism for interdependent power networks.

5.1 Implementation and experimental settings

Four IEEE reference networks are used[11] for the evaluation: (i) *case-30*, (ii) *case-39*, (iii) *case-57* and (iv) *case-2383*, where the numbers denote the network size in number of nodes. Case-39 has capacity information referred to as *link rating*. The rest of the networks use the α parameter with $\alpha = 2.0$. In this case, capacities are approximated by multiplying the α parameter with the initial power flows in each power line when no network disruptions occur. All power networks run AC power flow analysis except case-2383 that runs DC power flow analysis for faster performance and convergence. In all cases, the power flow analysis runs using the SFINA backend of InterPSS [42], as illustrated in Figure 2. The self-healing mechanism is implemented by extending the TokenCoordinator of the SFINA framework. The coordination of load reduction and load recovery between the damaged and the healer network is performed using event messages.

The self-healing capability is evaluated by an $m-1$ contingency analysis that repeats the following process: a link is removed, the power network undergoes a cascading failure, the metrics of Section 4.1 are computed at every cascade iteration, the network is restored back to its initial state and the whole process repeats for all $m-1$ link removals. This analysis characterizes probabilistically the overall network reliability and healing and is regarded a state of the art evaluation method for power networks [20, 22, 39]. The $m-1$ contingency analysis is performed for two network settings: (i) *baseline*, in which the damaged network is not connected to the healer network and (ii) *interdependent*, in which the damaged network self-heals in synergy with the healer network. The goal of the evaluation is to confirm whether self-healing interdependent networks achieve higher performance measured with the metrics of Section 4.1 compared to the baseline setting.

Table 1 summarizes three experimental scenarios used in the evaluation. Each scenario involves a damaged and a healer network that are interdependent. The damaged network undergoes a cascading failure and requires healing. The healer network supports the damaged network by exchanging power via bidirectional interlinks. They are chosen by a hyperparameter optimization using grid search, with the objective to maximize the healing of the damaged network while no damages are introduced in the healer network. This section studies the self-healing performance under a varied number of interlinks between the interdependent networks that is referred to as *interlink connectivity*.

Table 1: Experimental scenarios and the networks involved in the baseline and interdependent setting. Each interlink connects the nodes i, j of the damaged and healer network.

	Baseline Network	Interdependent Networks		
Scenario	Damaged	Interlink	Damaged	Healer
1	case-30	1	case-30, $i = 2$	case-39, $j = 2$
		2	case-30, $i = 4$	case-39, $j = 4$
2	case-39	1	case-39, $i = 2$	case-30, $j = 2$
		2	case-39, $i = 4$	case-30, $j = 4$
3	case-57	1	case-57, $i = 2$	case-2383, $j = 2$
		2	case-57, $i = 4$	case-2383, $j = 4$

Power flow exchanges over the interlinks are fixed to 60% of the power flow of the nodes which each interlink connects. This constant fraction emulates the power transfer distribution factors that characterize such interdependent networks in reality [4]. Other values tested do not change the main findings shown in this paper.

5.2 Experimental results

Figure 7 illustrates a visualization of a single instance (one link removal) from the $m-1$ contingency analysis for Scenario 1 and 2. The nodes of the networks are colored according to the physical entity they represent: (i) *generator*, which feeds the network with power, (ii) *bus*, which transfers power flows or is the load (consumption source) in the network and (iii) *slack bus*, which balances the power flow distribution by emitting or absorbing power flow. The dotted links represent the interlinks, in contrast to the solid links that depict the intralinks. For each Scenario 1 and 2, three states are shown: (i) the *initial state* (iteration 1 of baseline), (ii) the *damage state* (last iteration of baseline) and (iii) the *recovery state* (last iteration of interdependent networks).

[11] Available at http://www.pserc.cornell.edu/matpower/docs/ref/matpower5.0/menu5.0.html (last access: March 2018)

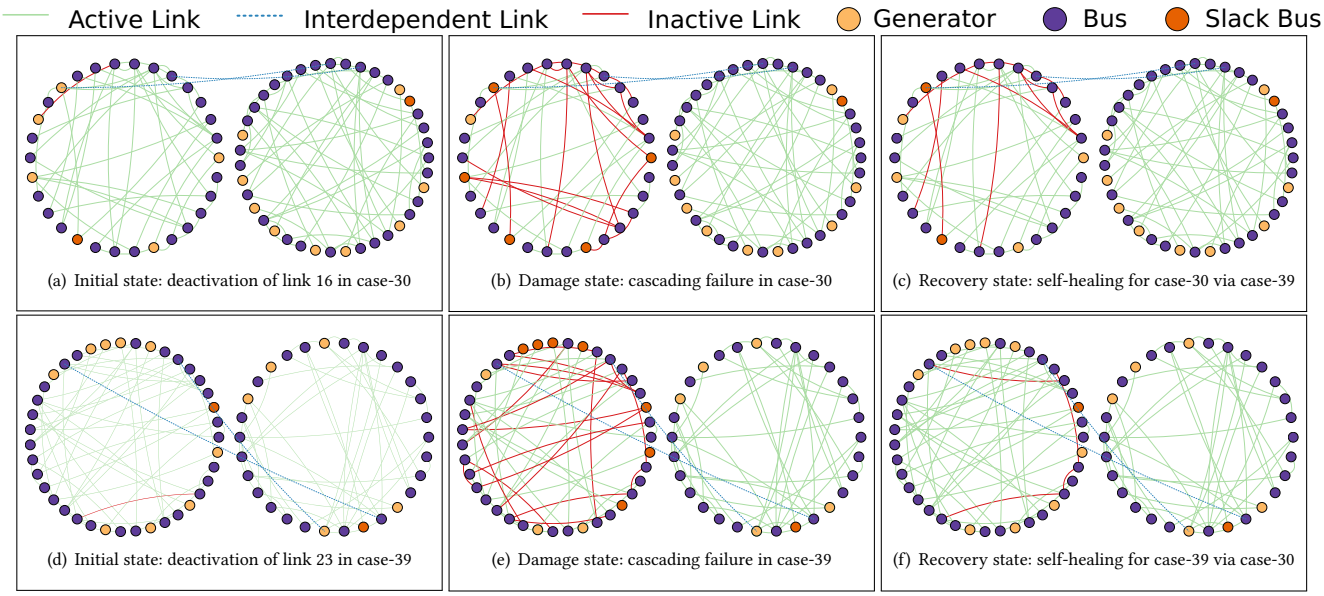


Figure 7: Visualization of the initial, damage and recovery state in Scenario 1 and 2 that involve case-30 and case-39.

The initial state shows the removal of link 16 in Scenario 1 and link 23 in Scenario 2, indicated with red color, which trigger a cascading failure. The link removals result in the damaged state with a large proportion of failing links. In the recovery state, the number of removed links significantly decreases thanks to the exchange of power flows between the interdependent networks.

Figure 8 illustrates the damage spread and power cascade of the baseline and interdependent networks measured over the $m-1$ contingency analysis.

In all scenarios of Figure 8a, 8c and 8e, the damage spread decreases significantly in interdependent networks compared to the baseline. The most significant reduction is observed in iterations 2, 3 and 4. However, the damage may progress for a few more iterations in Scenario 1 and 3 as a result of the power flow exchange between the interdependent networks, which though keeps on average the damage level lower than the baseline. In other words, the spread of the cascade slightly increases, while the damage level decreases. This is not the case in Scenario 2 in which the cascading failure over case-39 terminates one iteration earlier.

In Figure 8b, 8d and 8f, the power cascade of the baseline and the interdependent networks is shown. Two observations are made: (i) The power cascade decreases, though a significant decrease is only observed in Scenario 2. (ii) The self-healing process shows a higher capacity to decrease the damage level rather than the power cascade.

Figure 9 illustrates the damage correlation of the three experimental scenarios. The figure shows how correlated the failure or survival of link pairs are during the $m-1$ contingency analysis. Projecting the heatmap values on the respective network topology can provide insights about the localization of vulnerability in regions of the network, where nodes can be connected via interlinks to nodes of other networks to alleviate regional damages. Exploring this alternative approach to choose interlinks is out of the scope of this paper and it is subject of future research.

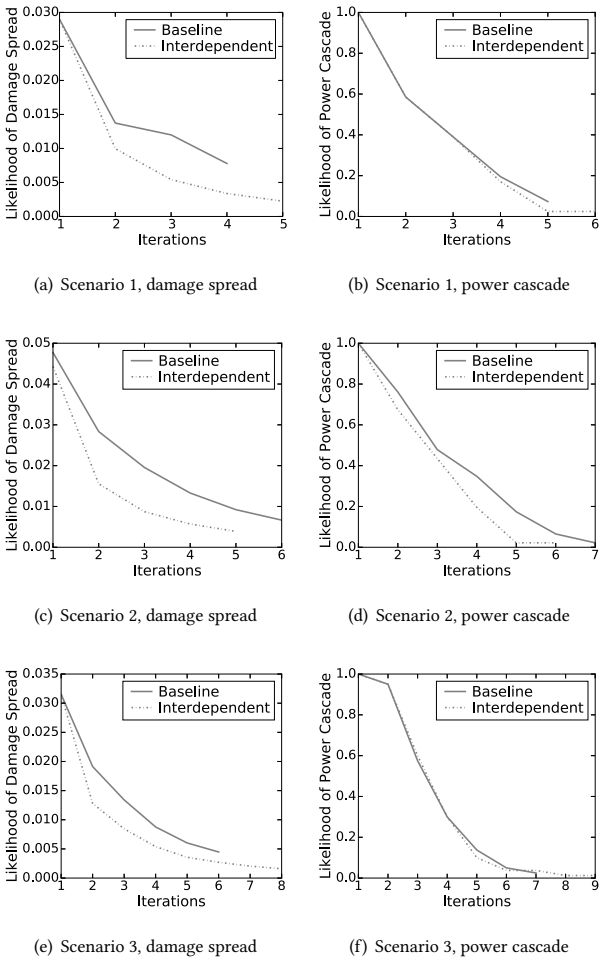
A larger number of links with high damage correlation is observed in Figure 9a, 9c and 9e, where the baseline is illustrated. In contrast, the interdependent networks of Figure 9b, 9d and 9f show a significantly lower number of link pairs with high damage correlation. Pair of links with negative correlation denote that during the $m-1$ contingency analysis, one link survives and the other fails. This is counter-intuitive to happen because the spread of all the triggered cascades overlaps to a high extent.

Figure 10a illustrates the link survivability and Figure 10b the relative load served, whose reduction denotes a blackout that leaves citizens without electricity. In both figures, Scenario 1 is shown and the percentage of interlink connectivity varies from 5% to 25%.

Figure 10 shows a trade-off: Higher interlink connectivity increases the incoming power flow in case-30 from case-39, which causes a low decrease in link survivability. In contrast, the served load increases as the load buses are energized by the interlinks via alternative pathways.

6 COMPARISON WITH RELATED WORK

Based on the earlier comparative taxonomy of seven evaluation keys [9], SFINA is classified as follows: (i) The *modeling focus* is on both systems with interdependencies as well as individual system analysis thanks to the modular and extensible interfaces of the TimingAgent. (ii) The modeling of each physical network as well as the distributed coordination network supports both a bottom-up and top-down *design strategy* on the performed simulations. (iii) Moreover, given that SFINA is by design domain-independent, i.e. separation of system dynamics from network topology, several *interdependency types* can be modeled, i.e. physical, cyber, geographic and logical. (iv) SFINA *events* can represent accidents, attacks as well

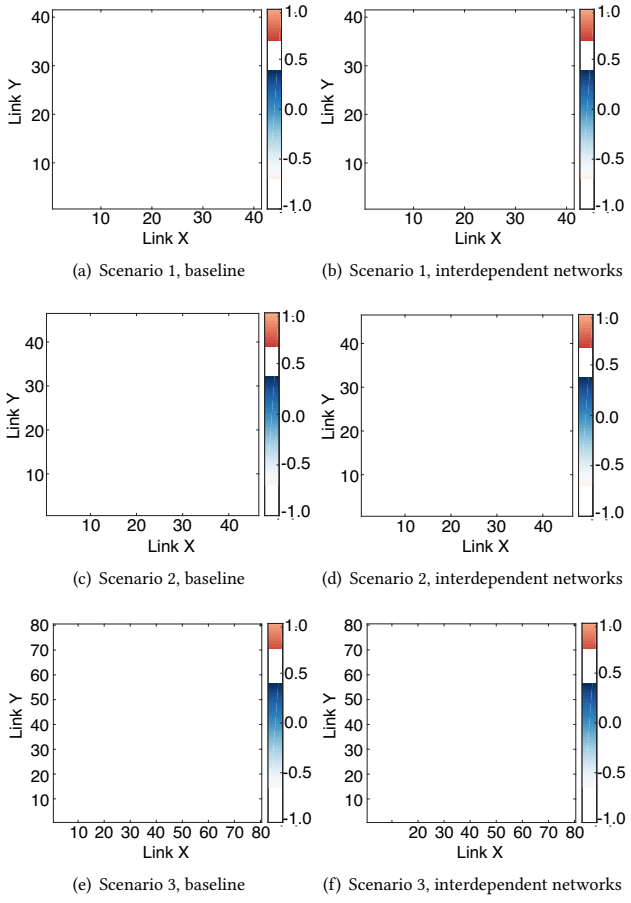


(a) Scenario 1, damage spread (b) Scenario 1, power cascade

(c) Scenario 2, damage spread (d) Scenario 2, power cascade

(e) Scenario 3, damage spread (f) Scenario 3, power cascade

Figure 8: Damage spread and power cascade in the three experimental scenarios.

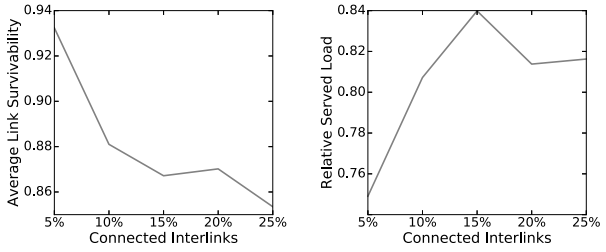


(a) Scenario 1, baseline (b) Scenario 1, interdependent networks

(c) Scenario 2, baseline (d) Scenario 2, interdependent networks

(e) Scenario 3, baseline (f) Scenario 3, interdependent networks

Figure 9: Damage correlation in the three experiment scenarios.



(a) Link survivability (b) Load served

Figure 10: Healing in Scenario 1 under a varied level of interlink connectivity.

as failures, by using probabilistic models or empirical data encoded in Event objects and data entries. (v) Given the flexible top-down and bottom-up design of SFINA, i.e. a coordination network over the interdependent networks, the *course of events* triggered cover cascading, escalating, common cause and confined ones. (vi) The *data requirements* of SFINA vary based on the complexity of the simulated scenarios. For instance, in the case study of Section 4, data about the topology, flows and the physical characteristics of the network are adequate to perform the $m-1$ contingency analysis. Further data can run more complex scenarios, e.g. historical data about failures of network components. (vii) SFINA covers the *scenarios* of information generation, failure analysis, vulnerability assessment as well as mitigation, prevention and self-healing strategies demonstrated in this paper. Compare to the other methodologies illustrated in that comparative evaluation [9], SFINA provides a superior coverage and flexibility over these evaluation keys as it unifies several simulation methods such as agent-based modeling (peer-level), system dynamics (modular domain backends), complex networks (modeling using temporal weighted directed graphs) as well as hierarchical holographic modeling, i.e. a distributed coordination network over a system-of-systems.

An earlier review [31] classifies modeling and simulation methodologies for interdependent critical infrastructures into *empirical*, *agent-based*, *system dynamics-based*, *economic* and *network-based*. The limitation to model and analyze component-level dynamics and the challenge to obtain validation data characterize system dynamics-based approaches. The calibration of utility functions during which elasticity values are required limits the applicability of economic approaches. Although the network topology determines system functionality, topological approaches cannot sufficiently model real-world interdependent networks [15, 30]. Moreover, modeling complex network flow operations can be computationally infeasible unless distributed agent-based approaches are employed. And this is the gap that the proposed model, implemented as extension of the SFINA framework, bridges. By combining system dynamics, topological, flow and agent-based distributed approaches, interdependent networks can be universally modeled and simulated efficiently at different granularity levels, i.e. management level for system operators to the level of the nodes and their domain-specific physical characteristics that govern the flow distributions.

The vast majority of existing modeling and simulation tools for interdependent networks are not open to the broader communities and they either serve commercial purposes or they are closed and customized for the purpose of the conducted research. Such software tools are reviewed in earlier work [31], for instance, N-ABLE, CIP/DSS, Aspen-EE, FAST, and CIMS [8]. The interoperation of NPMT [5] or CIMS [8] with other tools is not straightforward, while CISIA [33] relies on integrated commercial simulators, i.e. PSCAD/ EMTCD, PSLF, and NS2. In contrast, SFINA is released as an open source software and it is designed to play the role of a universal integration framework for research communities of different application domains.

Other relevant approaches [26] and software such as CISIA [7, 33] and I2Sim [24] adopt a macro-level design without modeling individual network components over which cascading failures caused by interdependencies spread. The IRRIIS Generic Information Model defines a semantic model and ontology for interdependencies, which though introduces a significant modeling work for users, who need to create complex domain representations. I2Sim [24] relies on an information database to empirically reason about a healthy operation and the impact of (cascading) failures. Building a comprehensive and complete database of this scale can be impractical or infeasible, in case of systems with a combinatorial complexity and non-linear dynamics. In contrast, SFINA is modular to support plugging of different domain backends, some of which include state of the art toolkits for domain simulation, for instance, MATPOWER for power grids or MATSim for transport systems.

7 CONCLUSION AND FUTURE WORK

This paper concludes that in contrast to related work, a general-purpose and domain-independent prototyping of self-management systems for interdependent and multiplex networks is feasible. On the one hand, the inner system dynamics of individual networks are simulated with state of the art toolkits with which SFINA interoperates. On the other hand, the distributed coordination network introduced in this paper provides a modular and extensible artifact to process and manage complex events originated by network interdependencies. The prototyping of a novel self-healing mechanism for interdependent power networks within this open source framework confirms its usability and also demonstrates its flexibility to support complex computational models, i.e. optimization of load exchanges, $m-1$ contingency analysis and measurements of network resilience against cascading failures.

Future work concerns the extension of the built-in SFINA measurements with metrics on interdependent networks [5], the expansion of the rule-based models for conflicts resolution to more sophisticated models, for instance, game theory models, as well as, the enhancement of the visualization capability with graphical layouts designed for interdependent networks [6]. Further case studies with multiplex interdependent networks of different types are ongoing work, along with the initiation of educational courses, workshops and seminars for community building. The evaluation of the self-healing mechanism for interdependent networks show that improvements of the earlier known reliability tradeoffs [2, 21] are feasible. Applying more complex distributed optimization [37] and learning [34] techniques for self-healing promises turning the risks of network interdependencies into opportunities for a highly adaptive and resilient digital society.

ACKNOWLEDGMENTS

The authors would like to heartily thank Jose Espejo-Uribe and Marius Bild for their contributions and dedication on the SFINA project, as well as Prof. Dr. Dirk Helbing for supporting and encouraging this research.

This work is supported by the European Research Council's Advanced Investigator Grant 'Momentum' with grant agreement #324247 and the European Community's H2020 Program under the scheme 'INFRAIA-1-2014-2015: Research Infrastructures', grant agreement #654024 'SoBigData: Social Mining & Big Data Ecosystem' (http://www.sobigdata.eu).

REFERENCES

[1] Massoud Amin. 2002. Modeling and control of complex interactive networks. *IEEE Control Systems Magazine* 22, 1 (2002), 22–27.
[2] Charles D Brummitt, Raissa M D'Souza, and Elizabeth A Leicht. 2012. Suppressing cascades of load in interdependent networks. *Proceedings of the National Academy of Sciences* 109, 12 (2012), E680–E689.
[3] Lubos Buzna, Karsten Peters, Hendrik Ammoser, Christian Kühnert, and Dirk Helbing. 2007. Efficient response to cascading disaster spreading. *Physical Review E* 75, 5 (2007), 056107.
[4] Hung-po Chao, Stephen Peck, Shmuel Oren, and Robert Wilson. 2000. Flow-based transmission rights and congestion management. *The Electricity Journal* 13, 8 (2000), 38–58.
[5] Arun Das, Arunabha Sen, Chunming Qiao, Nasir Ghani, and Nathalie Mitton. 2016. A Network Planning and Management Tool for mitigating the impact of spatially correlated failures in infrastructure networks. In *Design of Reliable Communication Networks (DRCN), 2016 12th International Conference on the*. IEEE, 71–78.
[6] Manlio De Domenico, Mason A Porter, and Alex Arenas. 2015. MuxViz: a tool for multilayer analysis and visualization of networks. *Journal of Complex Networks* 3, 2 (2015), 159–176.
[7] Stefano De Porcellinis, Roberto Setola, Stefano Panzieri, and Giovanni Ulivi. 2008. Simulation of heterogeneous and interdependent critical infrastructures. *International Journal of Critical Infrastructures* 4, 1-2 (2008), 110–128.
[8] Donald D Dudenhoeffer, May R Permann, and Milos Manic. 2006. CIMS: A framework for infrastructure interdependency modeling and analysis. In *Proceedings of the 38th conference on Winter simulation*. Winter Simulation Conference, 478–485.
[9] Irene Eusgeld, David Henzi, and Wolfgang Kröger. 2008. Comparative evaluation of modeling and simulation techniques for interdependent critical infrastructures. *Scientific Report, Laboratory for Safety Analysis, ETH Zurich* (2008), 6–8.

[10] Farshid Faghihi, Pierre-Etienne Labeau, Jean-Claude Maun, Arnaud Vergnol, and Vanessa De Wilde. 2015. A net balance-based approach in risk assessment of distributed generation curtailment. In *PowerTech, 2015 IEEE Eindhoven*. IEEE, 1–6.

[11] Wojciech Galuba, Karl Aberer, Zoran Despotovic, and Wolfgang Kellerer. 2009. 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), 60.

[12] Sergio Gomez, Albert Diaz-Guilera, Jesus Gomez-Gardenes, Conrad J Perez-Vicente, Yamir Moreno, and Alex Arenas. 2013. Diffusion dynamics on multiplex networks. *Physical review letters* 110, 2 (2013), 028701.

[13] Robert C Green, Lingfeng Wang, and Mansoor Alam. 2013. Applications and trends of high performance computing for electric power systems: Focusing on smart grid. *IEEE Transactions on Smart Grid* 4, 2 (2013), 922–931.

[14] Dirk Helbing. 2013. Globally networked risks and how to respond. *Nature* 497, 7447 (2013), 51.

[15] Paul Hines, Eduardo Cotilla-Sanchez, and Seth Blumsack. 2010. Do topological models provide good information about electricity infrastructure vulnerability? *Chaos: An Interdisciplinary Journal of Nonlinear Science* 20, 3 (2010), 033122.

[16] Petter Holme and Jari Saramäki. 2012. Temporal networks. *Physics reports* 519, 3 (2012), 97–125.

[17] Andreas Horni, Kai Nagel, and Kay W Axhausen. 2016. *The multi-agent transport simulation MATSim*. Ubiquity Press London.

[18] Yalin Huang and Lennart Söder. 2017. An investigation on the impacts of distributed generation curtailment regulations on distribution network investment. *Electric Power Systems Research* 145 (2017), 175–184.

[19] Prashant Kansal and Anjan Bose. 2012. Bandwidth and latency requirements for smart transmission grid applications. *IEEE Transactions on Smart Grid* 3, 3 (2012), 1344–1352.

[20] Mojtaba Khanabadi, Hassan Ghasemi, and Meysam Doostizadeh. 2013. Optimal transmission switching considering voltage security and N-1 contingency analysis. *IEEE Transactions on Power Systems* 28, 1 (2013), 542–550.

[21] Mert Korkali, Jason G Veneman, Brian F Tivnan, James P Bagrow, and Paul DH Hines. 2017. Reducing Cascading Failure Risk by Increasing Infrastructure Network Interdependence. *Scientific Reports* 7 (2017).

[22] Fengzhang Luo, Chengshan Wang, Jun Xiao, and Shaoyun Ge. 2010. Rapid evaluation method for power supply capability of urban distribution system based on N- 1 contingency analysis of main-transformers. *International Journal of Electrical Power & Energy Systems* 32, 10 (2010), 1063–1068.

[23] Joshua D Lyon, Kory W Hedman, and Muhong Zhang. 2014. Reserve requirements to efficiently manage intra-zonal congestion. *IEEE Transactions on Power Systems* 29, 1 (2014), 251–258.

[24] José Martí, Carlos Ventura, Jorge Hollman, K Srivastava, and Hugón Juárez. 2008. I2Sim modelling and simulation framework for scenario development, training, and real-time decision support of multiple interdependent critical infrastructures during large emergencies. In *NATO RTO Modelling and Simulation Group Conference, Vancouver, BC, Canada*.

[25] Manuel Matos, Ricardo Bessa, Audun Botterud, and Zhi Zhou. 2017. Forecasting and setting power system operating reserves. In *Renewable Energy Forecasting*. Elsevier, 279–308.

[26] Hyeung-Sik J Min, Walter Beyeler, Theresa Brown, Young Jun Son, and Albert T Jones. 2007. Toward modeling and simulation of critical national infrastructure interdependencies. *Iie Transactions* 39, 1 (2007), 57–71.

[27] Adilson E Motter and Ying-Cheng Lai. 2002. Cascade-based attacks on complex networks. *Physical Review E* 66, 6 (2002), 065102.

[28] Tu A Nguyen and ML Crow. 2016. Stochastic optimization of renewable-based microgrid operation incorporating battery operating cost. *IEEE Transactions on Power Systems* 31, 3 (2016), 2289–2296.

[29] A Nottrott, J Kleissl, and B Washom. 2012. Storage dispatch optimization for grid-connected combined photovoltaic-battery storage systems. In *Power and Energy Society General Meeting, 2012 IEEE*. IEEE, 1–7.

[30] Min Ouyang. 2013. Comparisons of purely topological model, betweenness based model and direct current power flow model to analyze power grid vulnerability. *Chaos: An Interdisciplinary Journal of Nonlinear Science* 23, 2 (2013), 023114.

[31] Min Ouyang. 2014. Review on modeling and simulation of interdependent critical infrastructure systems. *Reliability engineering & System safety* 121 (2014), 43–60.

[32] Sakshi Pahwa, C Scoglio, Sanjoy Das, and N Schulz. 2013. Load-shedding strategies for preventing cascading failures in power grid. *Electric Power Components and Systems* 41, 9 (2013), 879–895.

[33] S Panzieri, R Setola, and G Ulivi. 2004. An agent based simulator for critical interdependent infrastructures. In *Securing Critical Infrastructures, CRIS2004: Conference on Critical Infrastructures*. Citeseer, 25–27.

[34] Peter Pilgerstorfer and Evangelos Pournaras. 2017. Self-adaptive learning in decentralized combinatorial optimization: a design paradigm for sharing economies. In *Proceedings of the 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. IEEE Press, 54–64.

[35] Evangelos Pournaras, Ben-Elias Brandt, Manish Thapa, Dinesh Acharya, Jose Espejo-Uribe, Mark Ballandies, and Dirk Helbing. 2017. Sfina-simulation framework for intelligent network adaptations. *Simulation Modelling Practice and Theory* 72 (2017), 34–50.

[36] Evangelos Pournaras and Jose Espejo-Uribe. 2017. Self-repairable smart grids via online coordination of smart transformers. *IEEE Transactions on Industrial Informatics* 13, 4 (2017), 1783–1793.

[37] Evangelos Pournaras, Mark Yao, and Dirk Helbing. 2017. Self-regulating supply-demand systems. *Future Generation Computer Systems* (2017).

[38] Manish Thapa, Jose Espejo-Uribe, and Evangelos Pournaras. 2017. Measuring network reliability and repairability against cascading failures. *Journal of Intelligent Information Systems* (13 Jul 2017).

[39] Maria Vrakopoulou, Kostas Margellos, John Lygeros, and Göran Andersson. 2013. Probabilistic guarantees for the N-1 security of systems with wind power generation. In *Reliability and Risk Evaluation of Wind Integrated Power Systems*. Springer, 59–73.

[40] Jia Zhan, Nikolay Stoimenov, Jin Ouyang, Lothar Thiele, Vijaykrishnan Narayanan, and Yuan Xie. 2013. Designing energy-efficient NoC for real-time embedded systems through slack optimization. In *Proceedings of the 50th Annual Design Automation Conference*. ACM, 37.

[41] Feng Zhao, Eugene Litvinov, and Tongxin Zheng. 2014. A marginal equivalent decomposition method and its application to multi-area optimal power flow problems. *IEEE Transactions on Power Systems* 29, 1 (2014), 53–61.

[42] Michael Zhou and Shizhao Zhou. 2007. Internet, open-source and power system simulation. In *Power Engineering Society General Meeting, 2007. IEEE*. IEEE, 1–5.

[43] Ray Daniel Zimmerman, Carlos Edmundo Murillo-Sánchez, and Robert John Thomas. 2011. MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education. *IEEE Transactions on power systems* 26, 1 (2011), 12–19.