A Distributed Computing Platform for Community Resilience Estimation

Ahmed U. Abdelhady
Graduate Student, Dept. of Civil Engineering, University of Michigan, Ann Arbor, USA

Szu-Yun Lin
Graduate Student, Dept. of Civil Engineering, University of Michigan, Ann Arbor, USA

Lichao Xu
Graduate Student, Dept. of Civil Engineering, University of Michigan, Ann Arbor, USA

Omar A. Sediek
Graduate Student, Dept. of Civil Engineering, University of Michigan, Ann Arbor, USA

Andrew W. Hlynka
Senior Research Area Specialist, Office of Research, University of Michigan, Ann Arbor, USA

Sherif El-Tawil
Professor, Dept. of Civil Engineering, University of Michigan, Ann Arbor, USA

Seymour M.J. Spence
Assistant Professor, Dept. of Civil Engineering, University of Michigan, Ann Arbor, USA

Jason McCormick
Associate Professor, Dept. of Civil Engineering, University of Michigan, Ann Arbor, USA

Vineet R. Kamat
Professor, Dept. of Civil Engineering, University of Michigan, Ann Arbor, USA

Carol Menassa
Associate Professor, Dept. of Civil Engineering, University of Michigan, Ann Arbor, USA

ABSTRACT: Mitigating the risks of extreme natural hazards, such as hurricanes and earthquakes, triggers intricate interdependencies between various aspects of a community. These interdependencies can have a significant impact on community resilience quantification. To address this problem, a fully scalable and versatile distributed computing platform for the estimation of community resilience through the integration of discipline-specific models (simulators) is considered. A list of developed simulators is presented to clarify the idea that each simulator is treated as a black box that interacts with the simulation platform by subscribing to its input and publishing its output. Examples of how these developed simulators are connected effectively to calculate losses to the built environment due to hurricanes and losses to the built environment and lifeline infrastructure due to earthquakes are presented to demonstrate the versatility and scalability of the platform in the context of community resilience quantification.

1. INTRODUCTION

The estimation of the resilience of communities against natural hazards involves the integration of social, physical and economic aspects. Additionally, complex interactions will exist between these discipline-specific aspects, leading to a challenging problem that involves simulating within a multi-disciplinary setting and at multiple scales. Motivated by these needs, a versatile computational tool that allows researchers from different backgrounds and fields to link their computational models together to study the effects of natural hazards on community resilience is required.

The ability to communicate between separate pieces of software is not an unusual concept in computer programming, but it is typically hard-coded and managed by the participating simulators. Alternatively, a variety of standards and tools exist to handle this problem. Some common standards include "High-Level Architecture" (HLA) (IEEE Standard for Modeling and Simulation (2010)) and "Data Distribution Service" (DDS) (O.M. Group (2015)), offering a set of rules for developing a compliant system and rules to design a compatible simulator. Both commercial and open-source implementations exist for these standards. These solutions have limitations due to the nature of "standards" without a single maintained implementation: most available solutions admit to not being entirely compliant, are typically not easy to use and put additional burden on a user to understand many rules before beginning a simple example (Hollenbach (2009)).

There exist commercial and open-source tools for data-communication that do not rely on rule-heavy standards. "Lightweight Communications and Marshalling" (LCM) (Huang et al. (2010)) and "MathWorks Simulink" (Mathworks (2018)) are some examples. These implementations vary greatly in their ease-of-use. LCM requires a defined message format to be declared and compiled with the tool before a simulator can "publish" or "subscribe" to that message in programming code. This approach has been used in (Lin et al. (2018a) and Lin et al. (2018b)) but it does not allow for dynamic system redesign without significant work. Simulink has a powerful graphical interface and can use simulators from a variety of languages, but it has some limitations in message and function format for a simulator to be fully compatible.

These challenges inspired the design of a new software solution named the "Simple Real-Time Infrastructure" (SRTI) for the estimation of community resilience. The acronym RTI comes from terminology used in HLA to represent a shared-channel that can be published or subscribed to it. There are multiple core goals to the design of the SRTI:

1. Remain open-source and free to use for the research community.
2. Provide a pre-compiled version that can be utilized on most computers without recompiling source code.
3. Support distributed systems across multiple machines.
4. Consider scalability for more complex modules in future iterations of the SRTI design.

Within this setting, a set of developed simulators have been linked together via the SRTI to evaluate damage caused by hurricanes to the built environment and damage caused by earthquakes to the built environment and lifeline infrastructure. These scenarios illustrate the potential of the proposed platform and the use of distributed computing to effectively estimate general community-level resilience metrics.

2. SIMPLE REAL-TIME INFRASTRUCTURE "SRTI"

The structure of the SRTI (Figure 1) contains three components: the RTI Server, the RTI Lib API (Application Program Interface), and the user's simulators. The RTI Server and RTI Lib API are both precompiled and can be downloaded by the user.

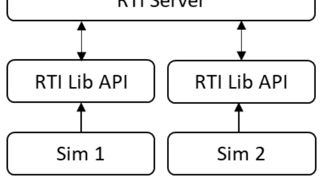

Figure 1: Diagram of the connection between the RTI Server, RTI Lib API and the simulators.

The RTI Server acts as a shared access point for all simulators to connect. The RTI Server must be launched before any simulator can connect to it, therefore it must be open before any simulator begins running.

The connection between the RTI Server and a simulator is done through sockets. Through the use of the RTI Lib API, this connection is abstracted and kept from the user's concern and can be changed in the future without need for the user updating their use of the API. The existence of the RTI Lib API makes connecting to the RTI Server and accepting and sending new messages a simpler process.

The SRTI relies on "publishing" and "subscribing" to messages with a known title. It is possible for multiple simulators to subscribe to the same type of message and for multiple simulators to publish the same type of message. This process is different from one simulator receiving a specific action request from a specific source, although this could be achieved through specific messages with that intended purpose.

From the user's perspective, a message must have two components: a "name" and "content." Internally, the RTI Lib API also includes elements that can have proprietary meaning, such as the name of the "source" of the message and the system clock "timestamp" when the message was first sent. These elements can be accessed by the user's simulator if deemed important.

The message elements are combined as a single JSON (JavaScript Object Notation) object, a readable standard of data representation. This format can easily be sent and received as String data, a common data type in most programming languages. The use of the JSON format is independent of the user allowing a change in format internally without altering the user's use of the SRTI.

JSON is the recommended format for "content," and the RTI Lib API includes functions to assist the user in creating a String value that represents such an object. This "content" would be capable of representing multiple objects within a single message, including integer numbers, floating point numbers, Boolean values, strings, and multi-dimensional arrays. Alternatively, the user can use any other alpha-numeric format they choose, provided the format can be read by other simulators.

The exact format of the "content" of the message does not need to be pre-defined for the SRTI to receive and send it to the connected simulators. However, all simulators that publish or subscribe to this message must be aware of the name and content of the message. This responsibility falls on the user. By giving the user full control of what "content" is passed between simulators, the SRTI tool remains versatile for both simple and complex simulations.

After subscribing to a given message, the RTI Lib API will always listen for that message while it runs, and it will store the message locally until the simulator requests to access it. It stores a buffer queue of all messages received (until the message is requested and used) to prevent data loss. The RTI Server will listen for any incoming messages, and immediately re-publish that message to any simulator subscribed to that message.

The RTI Server was written in Java and can be run on systems with the free Java Runtime Environment installed (provided by Oracle). The RTI Lib API is available in native Java and C++ and can be rewritten in other languages that support JSON parsing and socket communication. Both the RTI Server and RTI Lib API are available as source code and as pre-compiled libraries (.jar or .dll) to be used immediately with simulators written in a compatible language.

Listing 1 includes a simple example to describe the order of actions to utilize the SRTI. First, launch the RTI Server. It will display a simple GUI that confirms the "hostname" and "portnumber" credentials in which to connect. Next, import the RTI Lib API to make it accessible to your simulator code. Finally, add SRTI-specific function calls to connect to the RTI Server (using the "hostname" and "portnumber") and add logic to receive subscribed messages and publish new messages. The example represents a simple Java simulator that receives a numeric value, adds 1, and publishes a new value.

Listing 1: Simple Java simulator utilizing RTI Lib API to receive a number and publish updated number.

```
import mainServer.RTILib;

public class SimpleSim{

  public static void main
    (String [] args){

    RTILib lib = new RTILib();
    lib.setSimName("SimpleSim");
    lib.connect("localhost","4200");

lib.subscribeTo("OtherSimMessage");
    while(true){
      String newMessage =
        lib.waitForNextMessage
        ("OtherSimMessage");

      int value =Integer.parseInt
        (lib.getJsonObject("mValue",
        lib.getMessageContent
        (newMessage)));
      value = value + 1;

      String content =
        lib.setJsonObject
        ("", "newValue", value);
      lib.publish
        ("SimpleSimMessage",
content);

    }

    lib.disconnect();
  }
}
```

3. DEVELOPED SIMULATORS

This section lists the seventeen developed simulators to quantify damage caused by hurricanes to the built environment and damage caused by earthquakes to the built environment and the lifeline systems and infrastructure.

3.1. City simulator

The city simulator provides the topological configuration of various components of the considered community, including location and category of the utility facilities, connectivity between facilities, building coordinates and orientation, structural system, occupancy, soil properties, components capacity, etc.

3.2. Hurricane hazard simulator

For a given storm track and hurricane category, the *hurricane hazard simulator* provides time histories of the wind speed and direction for each building in the considered community. The simulator is based on the parametric models for the radial and tangential components of the hurricane wind field, which can be used without invoking the momentum equations and are provided in Jakobsen and Madsen (2004).

3.3. Direct wind pressure simulator

As a hurricane passes through the considered community, wind speeds and directions change continuously. The *direct wind pressure simulator* calculates the dynamic wind pressure that acts on each building as a function of wind velocity and direction. Code-specified wind pressure represents the envelope of maximum pressures due to all directions. In other words, code-specified wind pressure is not enough for considering the directional effect in pressure calculations. A hybrid approach that is based on ASCE/SEI 7-16 (2017) component and cladding wind pressure with modification based on Gurley et al. (2005) is used to calculate the dynamic wind pressure.

3.4. Wind-borne debris simulator

The trajectory of flying debris in a hurricane wind field is traced using a three-dimensional 6-degree-of-freedom trajectory model presented by Grayson et al. (2012). Debris sources are roof cover, roof sheathing and gable-end sheathing. The final outputs are the location where debris lands and kinetic energy upon landing.

3.5. Building hurricane damage simulator

The *building hurricane damage simulator* evaluates the damage for the components of each building due to two actions: excessive direct wind pressure and impact of flying debris.

3.6. Seismic hazard simulator

The *seismic hazard simulator* provides ground motions considering spatial propagation effects (vertically and horizontally). The vertical propagation is performed through different soil layers using elastic lumped mass type analysis formulated by Idriss and Seed (1968). The horizontal propagation is performed by scaling the ground motion history using the attenuation relationship formulated by Campbell and Bozorgnia (2008).

3.7. Structural analysis simulator

The ground motion acceleration at the location of each building in the community is received by this simulator at each time step during the earthquake from the *seismic hazard simulator*. This acceleration is used to evaluate the structural responses (i.e. story drift, floor velocity and floor acceleration) using nonlinear dynamic analysis.

3.8. Building seismic damage simulator

The *building seismic damage simulator* evaluates the damage state of each building in the community at each time step during the earthquake using the response limit states for the seismic hazard obtained from HAZUS (2003) and the structural response.

3.9. Component seismic damage simulator

This simulator evaluates the damage state of each structural and non-structural component in the building based on the engineering demand parameters (*edp*) and the fragility curves specified in the FEMA P-58 database (FEMA (2012a)).

3.10. Seismic debris simulator

The amount of debris generated at each building in the community is calculated using HAZUS methodology (HAZUS (2003)), which adopts an empirical approach to estimate two types of debris. The first type is the debris that falls in large pieces (e.g. RC or steel members). The second type is the debris that falls in smaller pieces (e.g. brick, glass, wood, etc.) that can be easily removed from the site.

3.11. Seismic casualties simulator

The casualties defined in FEMA (2012b) are either fatalities (loss of life) or injuries that occur inside the building envelope. To calculate casualties, it is necessary to determine a time of day and day of week to consider the population distribution in each building based on its occupancy (i.e. commercial, schools, etc.). In the case of building collapse, the number of injuries and fatalities are calculated based on the casualty rates specified in HAZUS (2003). Alternatively, if collapse did not occur, the number of injuries and fatalities at each time step during the earthquake are calculated using consequence functions specified in FEMA (2012b).

3.12. Lifeline systems direct damage simulator

This simulator obtains the physical damage of components of the lifeline systems which are the power system (EPS), water distribution system (WDS) and natural gas system (NGS) using a fragility function approach. The lognormal fragility functions used to estimate the capacities of different damage states for different types of utility facilities are adopted from the HAZUS (2003).

3.13. Lifeline systems interdependent damage simulator

The damage and functionality of components are updated considering functional and spatial interdependencies. The former includes the dependence of the pumping stations in water and gas systems on electric power for operating pumping machines, and the reliance of electric power plants on the water distribution system for cooling purposes. The later includes the spatial overlap of water and gas systems.

3.14. Lifeline systems performance assessment simulator

This simulator assesses the connectivity (C) of each system as follows.

$$C = \frac{1}{N_{\mathrm{D}}}\sum_{i}^{N_{\mathrm{D}}}\left(\frac{P_i}{P_{0,i}}\right) \tag{1}$$

where N_{D} is the number of demand nodes in the system, $P_{0,i}$ denotes the original number of supply nodes that connect to the ith demand node and P_i is the number of the supply nodes connected to the ith demand node after a perturbation.

3.15. Lifeline systems recovery strategy simulator

The *lifeline systems recovery strategy simulator* allocates the limited recovery resource based on the relative performance of the systems, as shown in Eq. 1.

$$R_{\mathrm{k}}(t) = \frac{1 - C_k(t)}{\sum_i^{N_s}(1 - C_i(t))} \times R_{\mathrm{Total}} \qquad (2)$$

where C_i denotes the connectivity of the ith system and the total amount of available recovery resource, R_{Total}, is fixed as 45 units/day. It is worth noting that the allocation of recovery resources is time-varying during the recovery process.

3.16. Recovery simulator

The required restoration time of each damage component is estimated based on the restoration functions in HAZUS (2003) and the recovery priority of the components in each network is specified in this simulator. If a damage component has been allocated one unit of resources in the recovery step (day), then its reconstruction progress will advance forward for one day, otherwise the progress will pause.

3.17. Interdependent recovery simulator

The same types of the interdependences as with the *lifeline systems interdependent damage simulator*, i.e., functional and spatial, are considered for recovery. The slave components are inoperative if the master components they depend on have not been fully recovered.

4. APPLICATIONS

This section shows how the above-mentioned simulators communicate, using the SRTI, to evaluate damage caused by hurricanes to the built environment and damage caused by earthquakes to the built environment and lifeline infrastructure in the context of quantifying community resilience.

4.1. Hurricane scenario

Simulators used in the hurricane scenario are shown in Figure 2. The *city simulator* publishes a message with all inputs describing the considered community. At each time step, the *hurricane hazard simulator* generates wind speed and direction for each building using coordinates from the *city simulator*. The *direct wind pressure simulator* calculates dynamic wind pressures to be used by the *building hurricane damage simulator* to determine if damage occurs. In the case of damage occurrence, internal pressure is recalculated as a function of level of damage which may lead to further damage. The *direct wind pressure simulator* and *building hurricane damage simulator* continue to iterate until balance occurs between the internal pressure and the level of damage.

Damaged components that are considered as debris sources are released in the wind field and traced using the *wind-borne debris simulator* to check if it hits a debris vulnerable component (e.g. glass window, glass door, etc.). In the case of impact, the *building hurricane damage simulator* determines if the vulnerable component got damaged. New damage induces variations in internal pressure that may lead to more damage. Iterations between the three simulators continue until balance occurs between damage and internal pressure.

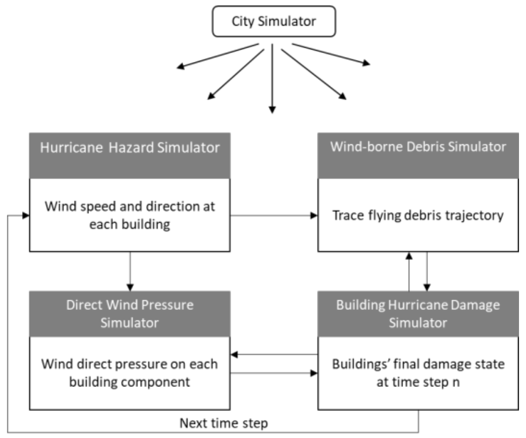
City Simulator
Hurricane Hazard Simulator
Wind speed and direction at each building
Wind-borne Debris Simulator
Trace flying debris trajectory
Direct Wind Pressure Simulator
Wind direct pressure on each building component
Building Hurricane Damage Simulator
Buildings' final damage state at time step n
Next time step

Figure 2: Hurricane scenario schematic diagram.

4.2. Seismic scenario

In Figure 3, the *city simulator* provides seismic building simulators with the required information about the community being studied. At each time step during the earthquake the *seismic hazard* *simulator* evaluates the ground motion parameters for each building. The *structural analysis simulator* uses the ground motion parameters to obtain the structural responses of each building using nonlinear dynamic analysis. The structural response is used to evaluate the structural and non-structural damage with the *building seismic damage simulator* and *component seismic damage simulator*, respectively. Subsequently, the casualties and debris generated at each building are evaluated in the *seismic casualties* and *seismic debris simulators*, respectively, based on the level of damage of all the components in the building. Table 1 lists the description of all the messages that passes between different simulators in the seismic scenario.

Table 1: Description of the messages being passed between different simulators

Message	*Description*
GM(t)	*Ground motion parameters at t*
EDPs(t)	*Engineering demand parameters at t*
BDS (t)	*Building damage state at t*
CDS (t)	*Component damage state at t*

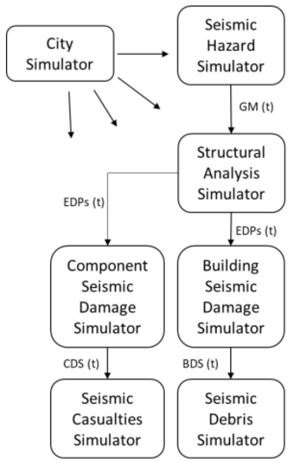
City Simulator
Seismic Hazard Simulator
GM (t)
Structural Analysis Simulator
EDPs (t)
EDPs (t)
Component Seismic Damage Simulator
Building Seismic Damage Simulator
CDS (t)
BDS (t)
Seismic Casualties Simulator
Seismic Debris Simulator

Figure 3: Seismic scenario schematic diagram.

4.3. Lifeline systems scenario

Figure 4 shows the simulators used for the lifeline systems scenario. First, the *city simulator* publishes the topographic configuration to the *seismic hazard simulator* which generates ground motions at each component of the considered networks. Second, damage is calculated on two stages: the direct physical damage caused by the earthquake using the *lifeline systems direct damage simulator* and the interdependent damage using the *lifeline systems interdependent damage simulator*, as described earlier.

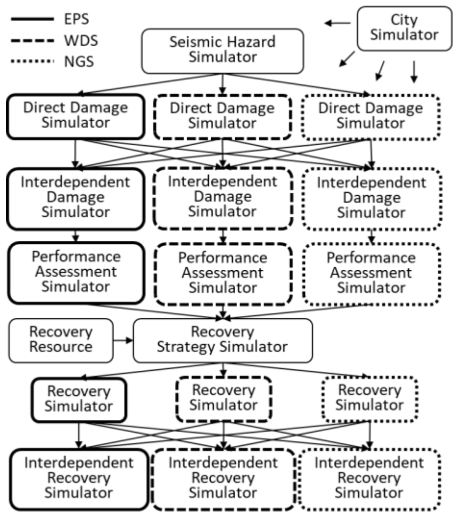
EPS
WDS
NGS
Seismic Hazard Simulator
City Simulator
Direct Damage Simulator
Direct Damage Simulator
Direct Damage Simulator
Interdependent Damage Simulator
Interdependent Damage Simulator
Interdependent Damage Simulator
Performance Assessment Simulator
Performance Assessment Simulator
Performance Assessment Simulator
Recovery Resource
Recovery Strategy Simulator
Recovery Simulator
Recovery Simulator
Recovery Simulator
Interdependent Recovery Simulator
Interdependent Recovery Simulator
Interdependent Recovery Simulator

Figure 4: Schematic for lifeline systems scenario.

Third, performance assessment of the whole system is calculated using the *lifeline systems performance assessment simulator*, which is the basis for what recovery strategy is followed. Finally, the recovery and interdependent recovery are evaluated.

5. CONCLUSIONS

This paper introduces the SRTI as a platform for data communication between computer programs or simulators. Its versatility makes it well suited for the implementation of distributed simulations that are essential for community resilience estimation.

The development of the SRTI is an ongoing project with a variety of planned improvements. The use of both network sockets and JSON objects that require parsing makes the SRTI system less efficient than other similar solutions. The purpose of SRTI is ease-of-use over efficiency, but because these methods have been abstracted from the user's perspective, they can be updated and improved in future versions.

The presented scenarios show that SRTI is a powerful tool in its current state with a lower barrier of entry than other platforms, making it suitable for prototyping complex simulation systems which are necessary for quantifying community-level resilience metrics.

6. REFERENCES

ASCE/SEI 7-16 (2017). "Minimum design loads and associated criteria for buildings and other structures." American Society of Civil Engineers, Reston, VA.

Campbell, K. W. and Y. Bozorgnia (2008). "NGA ground motion model for the geometric mean horizontal component of PGA, PGV, PGD and 5% damped linear elastic response spectra for periods ranging from 0.01 to 10s." *Earthquake Spectra*, 24(1), 139-171.

FEMA. (2012a). "Performance Assessment Calculation Tool (PACT)." *FEMA P58-3*, Applied Technology Council, Redwood City, CA.

FEMA. (2012b). "Seismic performance assessment of buildings." *FEMA P58-1*, Applied Technology Council, Redwood City, CA.

Grayson, J. M., Pang, W., and Schiff, S. (2013). "Building envelope failure assessment framework for residential communities subjected to hurricanes." *Engineering Structures*, 51, 245–258.

Gurley, K., Pinelli, J.-P., Subramanian, C., Cope, A., Z. Zhang, L., Murphree, J., Artiles, A., Misra, P., Gulati, S., and Simiu, E. (2005). "Florida public hurricane loss projection model, engineering team final report." International Hurricane Research Center Florida International University, Miami, FL.

HAZUS (2003). "Earthquake loss estimation methodology: Technical manual." National Institute of Building for the Federal Emergency Management Agency, Washington, DC,

Hollenbach, J. W. (2009). "Inconsistency, Neglect, and Confusion; A Historical Review of DoD Distributed Simulation Architecture Policies." *SISO*.

Huang, A. S., Olson, E., Moore, D.C. (2010). "LCM: Lightweight communications and marshalling." *2010 IEEE/RSJ international conference on Intelligent robots and systems (IROS)*, 4057-4062.

Idriss, I.M. and Seed, H.B. (1968). "Seismic Response of Horizontal Soil Layers." *Journal of the soil and Foundations Division*, ASCE, Vol. 94 No. SM4, July, pp. 1001:1031.

IEEE Standard for Modeling and Simulation (2010). "High Level Architecture (HLA) – Framework and Rules." *IEEE Std 1516-2010 (Revision of IEEE Std 1516-2000)*, 1-38.

Jakobsen, F. and Madsen, H. (2004). "Comparison and further development of parametric tropical cyclone models for storm surge modelling." *Journal of Wind Engineering and Industrial Aerodynamics*, 92, 375-391.

Lin, S. Y., Xu, L., Chuang, W. C., El-Tawil, S., Spence, S. M. J., Kamat, V. R., Menassa, C. C., McCormick, J. (2018a). "Modeling Interactions in Community Resilience." *Proceeding of the Structures Congress 2018*, Fort Worth, Texas, USA, April 19-21, 2018.

Lin, S. Y., Chuang, W. C., Xu, L., El-Tawil, S., Spence, S. M. J., Kamat, V. R., Menassa, C. C., McCormick, J. (2018b). "A Framework for Modeling Interdependent Effects in Natural Disasters: Application to Wind Engineering." *Journal of Structural Engineering*, accepted on October 19, 2018.

McKenna F, Fenves GL, Scott MH. (2000). "Open system for earthquake engineering simulation." *Berkeley (CA): University of California*.

Mathworks (2018). "Simulink - Simulation and Model-Based Design – MATLAB & Simulink." https://www.mathworks.com/products/simulink.html , (November 12, 2018).

O.M. Group (2015). "Data Distribution Service (DDS), Version 1.4."