Proceedings of the 1999 Winter Simulation Conference
P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans, eds.

DISTRIBUTED REAL-TIME SIMULATION FOR INTRUDER DETECTION SYSTEM ANALYSIS

Jeffrey S. Smith

Auburn University
Industrial & Systems Eng. Dept.
207 Dunstan Hall
Auburn University, AL 36830-5346,
U.S.A.

Brett A. Peters

Texas A&M University
Industrial Engineering Department
238 Zachry Engineering Center
College Station, TX 77843-3131,
U.S.A.

Sabina E. Jordan
Mark K. Snell

Sandia National Laboratories
Department 5838
Mail Stop 0780
1515 Eubank Blvd. Southeast
Albuquerque, NM 87123, U.S.A.

ABSTRACT

This paper describes a distributed simulation system developed for evaluation of physical security systems. The work extends previous work by extracting several system components that were previously integrated into the simulation model and creating distributed modules that interact directly with the system database. This allows the modules to incorporate arbitrarily complex logic and to be developed without detailed knowledge of the structure or language to implement the principal simulation component.

1 INTRODUCTION

This paper describes a distributed simulation system developed for evaluation of physical security systems. Jordan *et al.* (1998) describe the general problem being addressed by this simulation technology. The work described in this paper extends the previous work by extracting the line-of-sight, entity movement policy, and combat components from the simulation and implementing them as stand-alone components in a distributed environment. This decomposition simplifies the simulation logic and allows arbitrarily complex line-of-sight, entity movement, and combat logic to be developed and used without detailed knowledge of the simulation logic and the specific implementation language.

2 GENERAL SYSTEM STRUCTURE

The system being modeled involves entities moving within a facility. The facility generally represents a building or set of buildings and the entities generally represent people moving within and between the buildings. The entities and the facility structure are described in the following sections.

2.1 Intruder and Guard Entities

Simulation entities represent two classes of participants: intruders and guards. Intruders attack the facility and are trying to either simply reach a target within the facility or reach a target, obtain something of value, and exit the facility. Guards patrol the facility and try to interrupt intruders so as to prevent them from achieving their goals. When guards and intruders come in contact with one another, combat potentially ensues. The specific movement policies of guards and intruders are of particular interest and are discussed in more detail in Section 3.3.

2.2 Facility Description

The physical components of facilities are described as a set of *paths*, *barriers*, *sensors*, and *targets*. Entities (intruders and guards) are only allowed to travel through the facility on paths. Paths are represented by a graph structure where the nodes represent specific physical locations and the arcs represent the path between the end nodes. Targets are nodes that are of strategic importance and are generally on the paths of intruders. Barriers represent physical structures that potentially impede entity movement or vision. Walls, ceilings, floors, doors, and fences are common examples of barriers. Figure 1 illustrates a simple facility with four barriers (B1-B4) and six nodes (N1-N6). In the facility represented by this figure, entities can travel (and see) unimpeded between nodes N4 and N5 and nodes N4 and N6. As such, the travel time between these nodes will depend only on the entity speed. However, barriers B1, B4, and B3 impede travel between nodes N1 and N2, N3 and N4, and N2 and N5, respectively. The nature of the impedance caused by the barrier depends on several factors including barrier type, barrier state, entity type, barrier tactic, etc. These issues are discussed in Section 3.1. The travel time between any one of these pairs of

nodes will be determined by the speed of the entity and the time required to defeat the barrier.

Facilities also have a set of *sensors* that potentially detect the presence of entities. Two general classes of sensors are used: active sensors and passive sensors. Active sensors are considered to be continuously monitoring a specific zone within a facility. Examples of active sensors include cameras and motion detectors. Whenever an entity enters the zone covered by a sensor, there is a possibility of detection. The specific probability of detection depends on a number of factors including the sensor type, the entity type, and the sensor state. Conversely, passive sensors require some positive action by an entity in order to sense the entity's presence. Examples of passive sensors include card swipes, numeric key pads, or tampering with a control panel. Since passive sensors require positive action, they are associated with barriers and/or nodes. When an entity attempts to cross a barrier with an associated passive sensor, or arrives at a node with an associated passive sensor, there is a possibility of detection. As with active sensors, the probability that a detection will occur at a passive sensor depends on a number of factors including the sensor type, entity type, and sensor state. The detection model is described in more detail in Section 3.4.

2.3 System Dynamics

As described above, the system models entities moving through the facility. Intruder entities are trying to reach or acquire a specified target and guards are trying to prevent intruders from achieving their goals. As such, entities' movements through the facility are specified as paths through the facility graph. When there are no barriers between a pair of nodes, the time required to traverse the arc is determined by the speed of the entity. If there is a barrier between the nodes, the entity must first *defeat* the barrier and then traverse the arc. Barriers are defeated using one of two tactics: force and deceit. Force implies that the barrier is physically damaged or destroyed, allowing the entity to pass. Once an entity defeats a barrier using force, the barrier is generally no longer in tact, and subsequent entities can pass through the barrier without delay. Deceit implies that the entity is attempting to defeat the barrier through "dishonest" means (e.g., lying to a guard, using a stolen ID card or keypad code, etc.). This type of defeat generally leaves the barrier in tact and subsequent entities must also defeat the barrier.

As guards move through the facility, they are constantly looking for intruders. In addition, intruders are watching for guards as they move. When guard and intruder entities come into contact with one another, combat potentially ensues. Results of the combat would be one or the other or both entities are neutralized or injured, or that both remain unharmed. The result of combat is a probabilistic event whose characteristics depend on the current location of each entity. Moreover, entity movement policies potentially change as a result of sensor detections and sightings (e.g., the guards might chase intruders and the intruders might abandon their mission and exit the facility).

A "run" of the simulation is terminated when one of the following conditions is met.

1. All intruders have been neutralized,
2. All intruders have completed their goals (generally reaching a target or retrieving a target and exiting the facility), or
3. The simulation time expires.

The following section describes a distributed environment that supports the described system.

3 DISTRIBUTED SIMULATION SYSTEM

Jordan *et al.* (1998) and Smith *et al.* (1998) describe a stand-alone simulation tool for analyzing intruder protection systems of the type described in Section 2. However, while the stand-alone tools described in these papers provide significant performance/realism improvement over previous analytical and simulation methods, the tools are somewhat inflexible. More specifically, integrating complex entity movement policies and line-of-sight models is quite difficult and requires significant knowledge of the

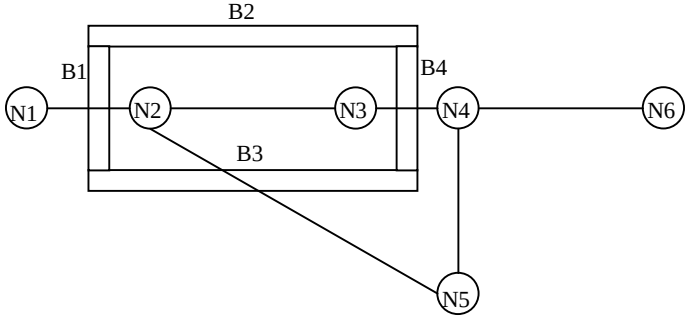

Figure 1: Example Facility

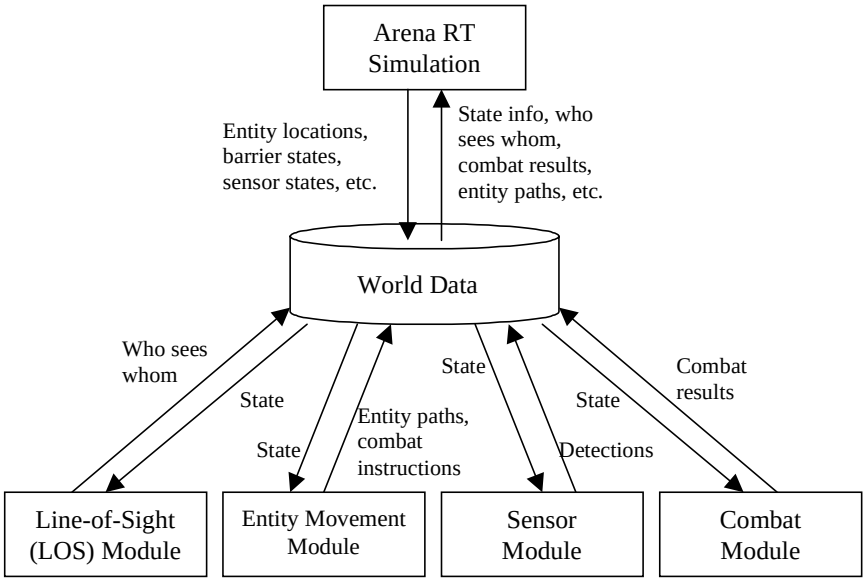

Figure 2: Distributed Simulation System Data Access Configuration

underlying simulation structure. The system described in this paper overcomes these limitations.

The distributed system partitions the system functions into 5 modules:

- Arena RT simulation module – responsible for maintaining the system state, updating entity locations, barrier and sensor states, tracking performance statistics, and animating the run;
- Line-of-sight (LOS) module – responsible for determining "who sees whom" (i.e., which entities see which other entities);
- Entity Movement module – responsible for determining entity paths through the facility;
- Detection module – responsible for determining when entities are detected by the security system as they move through the facility; and
- Combat module – responsible for determining the outcome of combat between entities.

In the distributed environment, each of these modules will require access to system state information. This information access is facilitated using a World Data database (as shown in Figure 2). Each of the modules and their corresponding data access requirements will be described in the following sections.

3.1 Simulation Model

Aside from the database integration and module synchronization functions, the simulation component is fairly straightforward. The facility graph nodes are modeled as stations and the facility graph arcs are modeled as routes between stations. Delays associated with defeating barriers are modeled as delays within the station corresponding to the node. The time to traverse the arc between two nodes is modeled as a route delay between stations. The basic pseudo code for the simulation model is as follows.

```
General Entity Logic
Arrive at node i
Set current node = next node
Sample for detection at passive sensors associated with node
call System Synchronization (see Section 3.6)
If next node != current node
   Check for barriers between current node and next node
   If barriers exist
      Sample time to defeat barrier
      Delay for sampled time
   Endif
   Sample time to traverse the arc
   Route to next node

Else
   Wait for next node to be updated by the entity movement module
Endif
```

Within the simulation, the *next node* is read from the World Data database. This value will be written to the database by the Entity Movement module (as described in Section 3.3). All of the sampled values (detection by the passive sensor, time to defeat barrier, and time to traverse the arc) depend on several components of the current

system state. These parameters are read from the World Data database.

Consider for example, determining the time required for an entity to defeat a barrier. In the current model, the time required to defeat a barrier depends on the entity type, the barrier state, and the tactic that the entity uses. So, for example when an intruder reaches a locked door and the entity movement module indicates that the intruder should defeat the barrier using force, the simulation queries the barrier information table to determine the delay based on an intruder using force on the door in its current state. Similarly, if a guard reaches a door and the entity movement module indicates that the guard should use his key, the simulation queries the database for the associated delay. Once the barrier has been defeated, the simulation updates the state of the barrier in the database. The specific state after defeat using a particular tactic is also specified by barrier type in the database.

The Arena simulation also includes an animation component that displays a 2-D view of the facility and the entities moving through the facility. The primary purpose of the animation is for validation of the database and individual module implementations.

3.2 Line-of-Sight Module (LOS)

The LOS is responsible for determining which entities see which other entities during the system execution. The LOS extracts entity locations and directions of view and barrier configurations and current states to make this determination. Currently, barriers are limited to vertical rectangles (e.g., walls, doors, fences) and horizontal trapezoids (e.g., ceilings). The LOS uses a geometric model to find all barriers between two entities based on the entities' locations and the barrier shapes. The barrier state information is then used to determine whether barriers are intact and present an obstruction to sight. The conditions for one entity to see another are as follows (the source entity is the entity that is looking and the target entity is the entity that the source entity is trying to see):

1. The target entity must be in visual range of the source entity (visual range is a parameter from the database);
2. The source entity must have "clear view" of the target entity – where clear view is defined as either no barriers between the entities, only transparent barriers between the entities, only barriers that are not intact, or combinations of transparent and forced barriers ; and
3. The target entity must be in the source entity's field of view – where the field of view is a cone defined by a direction of view and a field of view width angle.

3.3 Entity Movement Module

The entity movement module specifies entities' paths through the facility and determines when an entity will initiate combat with another entity. The module reads the system state from the database and uses internally coded entity movement and combat initiation policies/strategies to determine the paths and combat initiation. As a simple example, a fixed path strategy could be implemented by simply monitoring an entity's current node and updating its next node accordingly. As a more complex example, if the entity movement module is granted access to the barrier state information, this information could be used in a shortest path calculation to identify the quickest expected path to the target (or out of the facility). Similarly, combat could be initiated whenever a guard sees an intruder or more complex logic could be included (e.g., only initiate combat when the number of guards in a pre-specified area exceeds the number of intruders seen).

The simulation models presented by Jordan *et al.* (1998) and Smith *et al.* (1998) used relatively simplistic entity movement policies that were coded directly in the simulation logic. While the current example entity movement modules exhibit independent entity movement policies, the separation of the entity movement module from the simulation will allow arbitrarily complex entity movement policy models to be developed without detailed knowledge of the simulation implementation.

In addition, the entity movement module itself may be broken into several independent modules. For example, there may be one movement module for the intruders and one for the guards, since these two types or entities have different objectives and are therefore likely to have different operational logic. As another example, there may be a movement module for each individual entity with a particular entity serving as a commander and directing the activities of the others. Again, the flexibility of the distributed design allows these different scenarios to be seamlessly integrated as needed.

3.4 Detection Module

The detection module uses information about the active sensors to detect presence of entities within the facility. Each active sensor has a field of view based on its direction, range, and angle of view. As entities move through the system, the detection module checks to see if any entities are within a sensor's field of view. This check is similar to the LOS calculations as it also has to consider any barriers between the sensor and the entity. If the entity is within the sensor's field of view, the sensor will *detect* the entity with a certain probability.

In addition to the monitoring of each sensor, the detection module emulates the information propagation process through the sensor system. That is, once a sensor

is tripped, this information must be relayed through the system and eventually interpreted as a detection. The propagation and interruption take some time and require some additional information processing. This logic can be contained in a single module or can be represented by a hierarchy of modules. The distributed design allows flexibility for implementation of the detection module.

3.5 Combat Module

Once the entity movement module specifies that combat should be initiated, the combat module determines the outcome of the combat. Through its connection to the World Data database, the combat module can determine entity locations, barrier configurations and states, and other facility information. In addition, the World Data database also includes information about specific entities' weapons, ammunition levels, and equipment. As with the entity movement module, the simulation models presented by Jordan *et al.* (1998) and Smith *et al.* (1998) used relatively simplistic combat logic – each combat was basically a 4-outcome probabilistic event based only on the entities' positions. In contrast, the separation of the combat module from the simulation, will allow virtually any desired combat logic to be used.

3.6 System Synchronization

Since the logical modules were incorporated into the simulation model in the original implementation described by Jordan *et al.* (1998) and Smith *et al.* (1998), the synchronization between logical modules was handled implicitly by the simulation engine. However, once these modules are removed from the simulation and implemented as stand-alone components, explicit synchronization between modules is required.

The simulation model manages the synchronization function. The basic idea is that at discrete points in time, the simulation updates the system state, signals all modules, waits for the modules to respond, and advances simulation time to the next significant event. The following pseudo code illustrates this concept.

```
System Synchronization
Update system state (entity locations,
barrier states, etc.)
Send tick to all modules (LOS, entity
movement, combat)
Wait for all modules to respond with a tock
```

The idea is that no simulation time advances between sending the ticks and receiving the tocks. As a result, the LOS, entity movement, and combat modules can be considered to be state machines without explicitly considering time. Note that this does not preclude the modules from maintaining a memory of previous states and using this memory in deciding a course of action. It is this timing mechanism that allows arbitrarily complex module logic since the simulation time is not dependent on the running time of the modules.

In the current simulation model, the system synchronization is called on a user-assigned fixed cycle and when an entity arrives at a node (see Section 3.1). The following pseudo code illustrates the fixed synchronization cycle concept.

```
Update System
    Create a single entity
L1  Call System Synchronization
    Delay FixedUpdateCycle
    Goto L1
```

CONCLUSIONS

Evaluation of physical security systems is an important task. Jordan *et al.* (1998) motivate and describe this general problem and present an evaluation method based on a single simulation model. This paper describes a distributed simulation system developed for evaluation of physical security systems. It extends the previous work by extracting the line-of-sight, entity movement, detection, and combat activities from the simulation and implementing them as stand-alone components in a distributed environment. The design of this distributed system is described and overviews of each module are provided. The decomposition provides many benefits for modeling these types of systems. In particular, it vastly simplifies the simulation logic, and it allows arbitrarily complex operating logic to be developed and used within the modules without requiring detailed knowledge of the simulation logic or the specific implementation language.

ACKNOWLEDGEMENTS

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

REFERENCES

Jordan, S.E., Snell, M.K., Madsen, M.M., Smith, J.S., and Peters, B.A., "Discrete-event Simulation for the Design and Evaluation of Physical Protection Systems," *1998 Winter Simulation Conference Proceedings*, Washington, D.C., December, 1998, 899-907.

Peters, B.A. and Smith, J.S., "Real-time Simulation-Based Shop Floor Control," *Proceedings of ArenaSphere '98*, Pittsburgh, PA, June 1998, 188-194.

AUTHOR BIOGRAPHIES

JEFFREY S. SMITH is an Associate Professor in the Industrial and Systems Engineering Department at Auburn University. His research interests are in computer-aided manufacturing, shop floor control, and discrete event simulation. He holds a B.S.I.E. degree from Auburn University and M.S.I.E. and Ph.D. degrees from Penn State University.

BRETT A. PETERS is an Associate Professor in the Department of Industrial Engineering at Texas A&M University. His research interests are in facility location and design, computer-aided manufacturing, and logistics. He holds a B.S.I.E. degree from the University of Arkansas and M.S.I.E. and Ph.D. degrees from the Georgia Institute of Technology.

SABINA E. JORDAN is a Principal Member of the Technical Staff at Sandia National Laboratories. She has been involved in safeguards and security R&D for the past ten years, including security vulnerability assessment tool development. She holds a B.S. degree in Computer Engineering from the University of New Mexico and an M.S. degree in Electrical Engineering from the University of Southern California.

MARK K. SNELL is a Principal Member of the Technical Staff at Sandia National Laboratories. He has been involved in developing and using security vulnerability assessment tools since 1982. He holds a B.S. degree in Economics from Syracuse University and a Ph.D. in Operations Research from Cornell University.