Modeling Autonomous Security Systems

Colin Cochran (New Mexico Institute of Mining and Technology)

Summer Undergraduate Program in Engineering Research at Berkeley

Faculty Mentor: Professor Edward Lee (UC Berkeley)
Graduate Mentor: Stephen Neuendorffer (UC Berkely)

ABSTRACT

Autonomous security systems have the advantage that they require no outside intervention to function. Problems arise, however, when one wishes to modify or extend the system, or interface it with other systems. These problems may stem from the communication medium, such as a hard-wired implementation that requires intrusive measures to modify the system. Interfacing multiple systems may also pose problems when they operate under different computational models. In either case, the resulting system may not function as expected. This paper proposes a method to create models of these systems that can be easily modified or interfaced with other system models that may be running under different computational domains, thereby making alterations to the original system practical. Our approach to the problem consisted of extending the functionality of an existing modeling application, namely Ptolemy II. The existing application offered a way to model heterogeneous systems, which solved the computational compatibility issue. An interface library was developed that allowed Ptolemy II models to interact with a commercially available home automation product called X-10. These X-10 devices allow remote control of appliances, lights, motion sensors and other security fixtures, thereby rendering a highly extendable physical infrastructure. Finally, we construct working models of security systems that interface with various systems running under different computational domains. Thus, by leveraging the system modeling power of Ptolemy II and the physical extendibility of X-10 we show that constructing and modifying complex security systems is indeed practical.

1. INTRODUCTION

This paper outlines a method to model complex security systems while accomplishing two design goals:

1. Provide a means to interface security models with any other model.
2. Provide an intuitive facility to modify models.

Two established technologies provided solutions for both of our goals. One was a modeling environment capable of constructing heterogeneous systems while the other was a home automation infrastructure that transformed a home electrical system into a communications network. This paper systematically describes both technologies and their respective advantages. Thereafter, it describes how these technologies were interfaced to exploit these advantages. Finally, we show how our goals were met while outlining ways to improve upon our design for future research.

2. MODELING ENVIRONMENT: PTOLEMY II

Ptolemy II provides a component based facility to build heterogeneous systems. The component structure suited our security system design since physical implementations could be neatly symbolized in Ptolemy II models. Secondly, Ptolemy II provided a means to interface systems that differed in computational domain. These two traits meant extending an established system would not require extensive rework, which is our goal.

2.1 Component Based Design

Models in Ptolemy II are constructed using components (actors) that perform various actions on given input, hence their name. Actors communicate with each other over connections between ports on the actors. When a model is run, actors signal each other with input to act upon while adhering to communication protocols. These protocols are determined by the computational domain and are implemented by directors.

The component based modeling structure of Ptolemy II lent an intuitive means for constructing security models. A complex system could be aggregated as a compound component and then interfaced or embedded with or within any other system. This modular structure hastened model construction while rendering intelligible, complex systems. [2]

2.2 Computation of Models

Ptolemy II models can make use of various computation models, which governs the interaction among actors. Models can also interact with other models that differ in computational domain. This grants the user flexibility when creating complex models. It also allows any embedded security system, modeled in Ptolemy, to interface with any other system, modeled in Ptolemy. For, example, a home security system, following a discrete event model of computation, can interface with a home environment control system, following a continuous time model of computation. The interface between these models of computation is abstracted to the developer. [1]

3. PHYSICAL SECURITY INFRASTRUCTURE: X-10

Our approach to building a reconfigurable security system called for some infrastructure to add, remove or modify a system's physical setup easily. Due to its ease of use and extendibility, the X-10 infrastructure was chosen to implement the physical security fixtures. This commercially available home-automation product offered a modular approach to an autonomous system by implementing a communication network via the electrical system of a given structure. This meant than an existing alternating current (AC) power line network—found in most modern structures—could be exploited for communication purposes between security fixtures. [3]

3.1 X-10 Communication

Communication amongst security hardware inside a structure is accomplished by exploiting the structure's electrical system. The X-10 network uses electrical power lines as communication lines and wall outlets as communication ports. Communication between devices follows a transmission protocol for sending and receiving commands to and from X-10 compliant devices, which are plugged into standard wall outlets. Any appliance that is connected to these modules may be controlled by a central X-10 command transceiver. In this way, an X-10 network can be established in any structure that has an electrical system.

Devices	Supported Commands
Appliance Module	*on* *off* *all units off*
Lamp Module	*on* *off* *bright** *dim** *all lights on* *all lights off* *all units off*
Occupancy Sensor	*on* *off*

Figure 1

Several X-10 modules were used to construct the security networks. These include appliance modules, lamp modules and occupancy sensors. Each device supports a static set of X-10 commands; an exhaustive list is given in figure 1. Furthermore, each module has an alpha-numeric address composed of a *house code* and a *unit code*, which are set by their respective dials located on the module. The modules will only respond to commands with this address.

In our implementation, the physical signal transmission is handled by an X-10 serial interface controller. The implementation currently supports the following interfaces:

1. *CM11A* Serial Interface (wired)
2. *CM17A* Serial Interface (wireless)

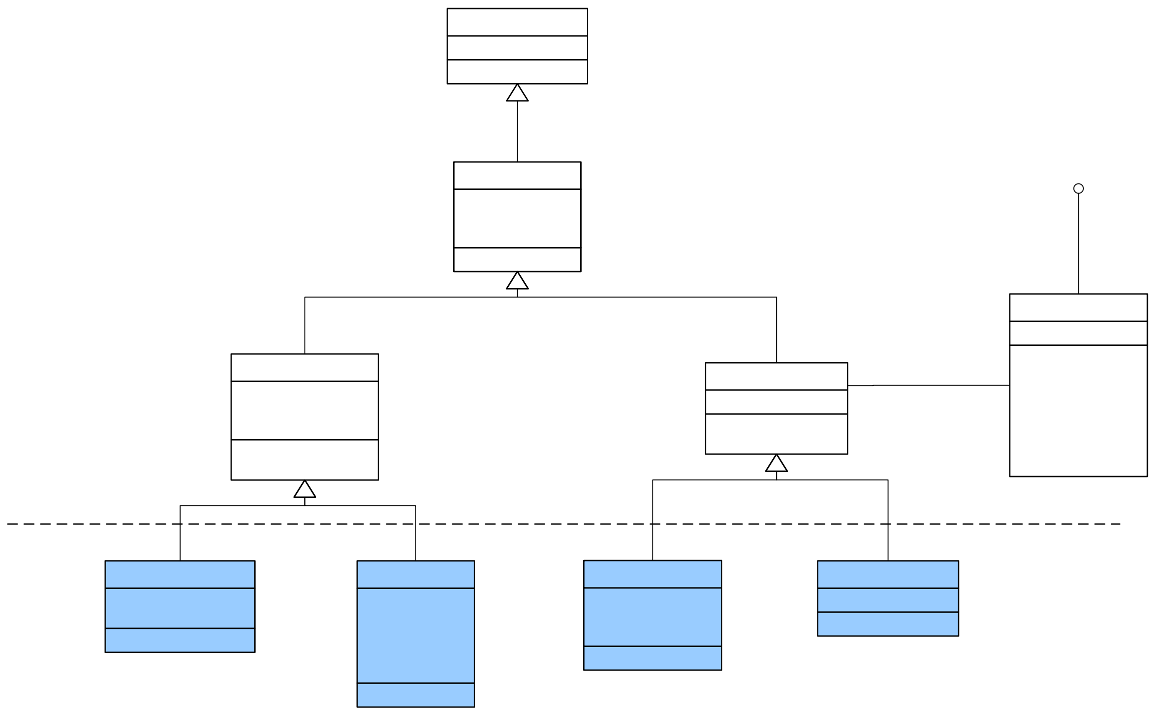
Each transmitted command is synchronized with the zero crossing point of the AC power line. Signals are sent within 200 microseconds of this point and one command packet requires eleven cycles of the power line to transmit. The serial interface controller handles this functionality and is abstracted to the user. Although a two-way X-10 infrastructure exists, the one-way variant was chosen due to time constraints.

4. THE PTOLEMY II / X-10 INTERFACE

An interface was required to link Ptolemy II models with X-10 networks. The majority of the research time was spent here for creating an interface library that allowed Ptolemy II models to control X-10 modules. Once the interface library was established, the advantages of both technologies could be exploited.

4.1 Library Design

UML Representation of the X-10 Interface Library *Figure 2*



The library design places Ptolemy X-10 controller components into two categories: senders and receivers. Senders transmit commands to specific X-10 devices, such as appliance controllers, while receivers listen to X-10 network activity and filter out incoming messages with specific addresses. This implementation offers the following advantages.

1. Library extendibility
2. Intuitive device abstraction

Extending the Ptolemy X-10 library is facilitated through code reuse. New actors are developed by extending the functionality of a sender, receiver or both. Since the core functionality for each type of controller is already implemented, new actors only add functionality that is specific to their unique purpose.

Typically, each Ptolemy component represents one and only one X-10 module. Each component has a specific set of commands that it may send or receive. Since the interface functionality to X-10 modules is abstracted, the user does not need to understand how it functions to build Ptolemy II models. Obviously, this approach expedites model construction and reduces model complexity. The house and unit code for each X-10 component in Ptolemy is set to the same address as the X-10 module it symbolizes. This implementation avoids confusion in large models and provides the user with building blocks to construct systems. Figure 2 provides a unified modeling language (UML) representation of the X-10 interface library. Public members and fields are denoted by a "+" sign while protected members and fields are denoted by a "#" sign.

In some cases, however, a one-to-one actor-module relationship is not advantageous. The X-10 network listener actor is a good example. This actor was designed to monitor X-10 network activity. It enables a user to parse all X-10 commands propagating through the network and subsequently act upon them as necessary. This exploits the fact that all commands in X-10 are necessarily broadcasted as datagrams. This leads to some security pitfalls, however, which are discussed later.

4.2 Actor Design

Sender

#_destination

+houseCode

A set of X-10-actors was developed and provided building blocks for our security model implementations. Two senders and two receivers were created and provided enough functionality for our system models. Each actor works with a specific set of commands. Generally, each command maps to one and only one port on the actor. The following is an exhaustive list of actors implemented in the library.

1. *Appliance Controller*
2. *Lamp Controller*
3. *Occupancy Sensor*
4. *Listener*

Although each actor was designed to function in various computational domains, the discrete event (DE) is generally the most intuitive director to apply to models based upon these actors. This is because we would like components to react to certain events such as when an occupancy sensor detects movement in a hallway or when a model changes states, and to be able to send commands according to a time schedule.

4.2.1 Appliance and Lamp Controller

The appliance and lamp controllers are X-10 senders that share some transmittable commands. Figure 3 depicts a Venn diagram describing the set of instructions that either controller may send.

Appliance and Lamp Controller Command Sets

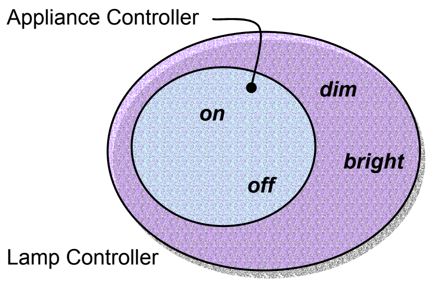


Figure 3

Exactly one input port exists for each command on either controller. Thus, the appliance controller has two input ports while the lamp controller has four. When any of these ports receive a true input, the controller sends the respective command to the X-10 module located somewhere in the structure. The lamp controller has a special integer parameter that governs the amount to dim or brighten the light. These actors are depicted in figure 4 with the

Lamp and Appliance Controller Actors

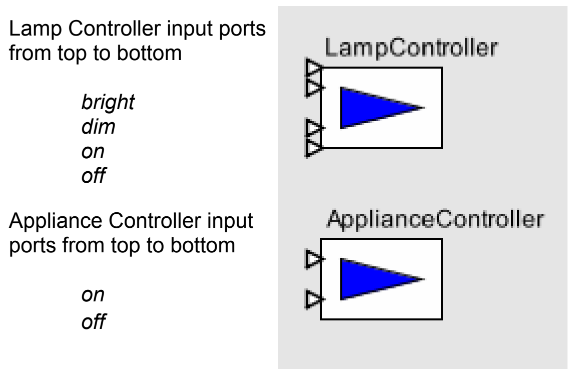

Lamp Controller input ports from top to bottom
bright
dim
on
off
LampController
Appliance Controller input ports from top to bottom
on
off
ApplianceController

Figure 4

aforementioned ports enumerated on the left. As one may notice, the appliance controller functionality is a subset of the lamp controller functionality. Although this is true, the controller actor provides a clear representation the X-10 appliance module and only provides the user with commands that an appliance module understands. Again, this abstracts the inner workings of the X-10 infrastructure to avoid confusion in models.

4.2.2 Occupancy Sensor and Listener

The occupancy senor actor is a receiver that only listens for messages sent by specific senor modules. Thus, each occupancy actor represents exactly one occupancy sensor module. This is accomplished by matching the address on the module with the one on the actor. The occupancy sensor actor has one output port that sends a Boolean true if and only if the respective module detected motion.

The listener actor behaves differently in that it does not follow our one-to-one paradigm. The X-10 listener was designed to provide a means to monitor X-10 network activity. It will respond to any command sent on the X-10 network and output a string representation of each command on its output port. The idea here is that subsequent actors may filter out commands.

5. MODELS

A series of models were constructed in Ptolemy II and interfaced with X-10 modules. The first models simply controlled lamps and appliances through command shells. Later, models were designed to respond to input from various occupancy sensors and subsequently turn on lamps or appliances to signal that movement had been detected in certain zones of a structure.

Finally, a complex system was developed that would switch states depending on which occupancy sensors had detected movement. Depending on the current state, the control system would dim, turn on, or turn off series of lights. Furthermore, the model was able to communicate system activity to other models running on various computers via a secure connection over Ethernet.

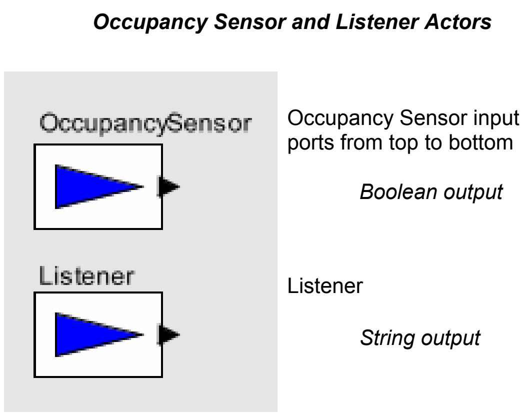

Occupancy Sensor and Listener Actors
OccupancySensor
Occupancy Sensor input ports from top to bottom
Boolean output
Listener
Listener
String output

Figure 5

5.1 Design Shortcomings

Although we were able to construct and modify models with great success, we discovered serious pitfalls with the X-10 network. In its current implementation, X-10 may not be a suitable means of communication within in security system.

5.1.1 Inherently Insecure Communication

The X-10 network does not provide a secure means of communication since its communication lines are easily compromised. Anyone who has access to an AC power outlet may gain access to the security system. Thus, many exploits are available to an adversary; some are outlined in this section.

Currently, a module cannot determine the origin of a given command. Therefore, a module cannot verify that the command had a valid source. The module can potentially execute a false command sent by an adversary who wishes to compromise the security system.

Since commands are broadcasted and unencrypted, commands may be monitored by anyone without authorization. Thus, an adversary can eavesdrop on a security system's network traffic.

Potentially the most hazardous attacks X-10 networks are susceptible to are denial of service (DoS) attacks. An adversary could easily flood an X-10 network with false commands. The system would not be able to distinguish between valid or invalid commands, thereby rendering a useless security system.

5.1.2 Communication Issues

The X-10 interface controller we choose detected commands far too slow for a practical system to be implemented. The CM11A interface controller would take nearly two seconds to register a command from the X-10 network. Perhaps other X-10 interface devices are able to register commands quicker; in any case there was not enough research time available to test other interface devices.

The CM11A interface device posed another design issue. There was no way to poll X-10 modules for their current states. This made it very difficult to implement a fault-tolerant security system since it was impossible to determine a security system's current state.

6. CONCLUSION

We have presented a method for constructing and modifying complex autonomous security models while accomplishing the following goals:

1. Provide a means to interface security models with any other model.
2. Provide an intuitive facility to modify models.

We met our design goals by interfacing a system modeling application with a publicly available home automation tool. Ptolemy II offered a modeling system that could create heterogeneous systems and a means to reconfigure them. The X-10 home automation product offered a highly extendable infrastructure to physically implement the security models. The advantages of both technologies were exploited by creating an interface library that allowed the modeling tool to communicate with the remote security fixtures that could be setup anywhere in a structure with AC power lines and standard wall outlets.

However, we found room for improvement in various aspects of the modeling methods. A faster, more secure means of communication between security fixtures is required. The X-10 network offers too many security exploits since the network is easily accessed. Furthermore, the interface device requires too much time to register a command.

To mitigate these issues, we propose that a different communication medium is implemented in our method for modeling security systems. Perhaps an encrypted wireless system would retain the physical extendibility of the system while providing fast, secure lines of communication.

7. Acknowledgments

I would like to thank the faculty and staff of the 2003 SUPERB-IT program for giving me this opportunity to conduct graduate level research—this has been an amazing experience. I would also like to acknowledge my faculty mentor, Professor Edward Lee, who has been a dedicated mentor. He showed great interest in me and my work and kept me motivated through positive support. I'd also like to thank my helpful graduate mentor, Stephen Neuendorffer who provided me with direction and knowledge. Special thanks go to Xaiojun Liu who was always around the lab and who was steadfastly determined to help me and my fellow CHESS-mates work through bugs during our irregular work hours.

8. REFERENCES

[1] Homelinux. *X10 10 CORE API.* http://x10.homelinux.org/, 2003.
[2] Edward A. Lee. Overview of the Ptolemy Project, *Technical Memorandum UCB/ERL M03/25*, University of California, Berkeley, CA, 94720, USA, July 2, 2003.
[3] Edward A. Lee and Yuhong Xiong, "Behavioral Types for Component-Based Design," *Memorandum UCB/ERL M02/29*, University of California, Berkeley, CA 94720, USA, September 27, 2002.