INTELLIGENT NETWORK CONFIGURATION OPTIMIZATION TOOLKIT (INCOT)

Robert Richards, Coskun Tasoluk
Stottler Henke Associates, Inc.
San Mateo, CA

ABSTRACT

Stottler Henke has developed and continues to enhance the Intelligent Network Configuration Optimization Toolkit (INCOT) for the US Air Force. INCOT's goal is to provide an intelligent tool to allow for the rapid design and optimization of communication networks driven by user requirements, while eliminating the need to have knowledge of a complex network simulation product, such as OPNET Modeler. There is currently a great need for a solution that dramatically reduces the complexity of network configurations for rapid field deployment. By incorporating expert knowledge of network engineering, policies, available components and their characteristics, INCOT is providing an interactive goal-driven rapid design tool. The user is led through the network design process via a step-by-step requirements-driven hierarchical workflow that automates much of the low-level details of network design. INCOT interfaces with OPNET products (e.g., Modeler) as well as NETWARS and TNAPS+. For example, INCOT can automatically lay out all the network components in OPNET Modeler or NETWARS. INCOT also includes a stand-alone TNAPS+ translator so TNAPS+ networks can be brought into OPNET tools or NETWARS for further analysis.

INCOT is an open system and includes a powerful visual authoring tool that allows network experts to enhance INCOT's logic, automations, policies and constraints. The authoring tool supports complex logic, algorithms and procedures to analyze data in order to drive appropriate actions. For example, with INCOT's visual authoring tool an expert can author classified logic, policies and constraints. INCOT is extending the power of network simulation tools, such as OPNET Modeler, NETWARS and TNAPS+, while simplifying its operation for the Air Force and other DoD network engineers, thus allowing less experienced network engineers to perform efficient and effective network engineering. INCOT is available now and is free to anyone in the DoD.

INTRODUCTION

Modern conflict is becoming ever more dependent on larger quantities of information and communications. Intelligence and other critical information in the field are often transmitted over networks that may need to be configured under severe time constraints. Given these time constraints, it is imperative that networks in tactical military situations be designed to maximally support the goals of the mission. Thus, a solution is needed to meet the goal of engineering networks to meet mission critical data requirements, while allowing for the design, testing and optimization of such networks under severe time constraints.

The problem is compounded because network engineering under any circumstance requires expert knowledge that is dependent on the context and environment in which the network will need to operate. In addition, there are various other constraints, including available equipment, security considerations and other policy-based constraints. Present commercial simulation engines are powerful tools for developing and testing potential networks, but their power has, to date, been accompanied with a high level of complexity, so the utilization of commercial simulation engines has associated with it the need for understanding of the complex software, sometimes including programming skills, al-though many aspects of the software may never be used. A related problem is that many network engineers utilize legacy systems, so thus have no experience with modern commercial simulation engines.

Fortunately, the power afforded by the state-of-the-art commercial network simulation tools is sufficient to simulate most of the networks that need to be constructed. The major obstacle is that the complexity of the tools does not al-low their power to be exploited under the time constraints afforded by modern tactical military situations, except via expert users who may not be available. Another way to view the problem is that users with a working knowledge of communications networks (though not necessarily an expert in any network simulation tool) need to be able to construct optimal networks quickly. These less-than-expert users must still build the network as if they had expert knowledge of device characteristics, and must adhere to all policies. An associated problem for network engineers with experience only with legacy systems is to allow the legacy networks to be utilized in state-of-the-art commercial network simulation tools.

SYSTEM DESCRIPTION AND DESIGN

The following sub-sections provide the objectives, a functional description and an overview of the architecture for the automated intelligent system for designing communication networks, referred to as the Intelligent Network Configuration Optimization Toolkit (INCOT).

Objectives

The objective of INCOT is to develop an innovative intelligent toolkit to automate the design of communication networks. This toolkit allows rapid design and optimization of communications networks without requiring the user to have programming skills or knowledge of a particular networking tool. By incorporating expert knowledge of network engineering, policies, available components and their characteristics, INCOT provides an interactive goal-driven rapid development tool to the network engineer that quickly generates optimal network configurations.

Functional Description

INCOT greatly extends the power of other networking tools, while simplifying its operation for Air Force and other DoD network engineers. One of the greatest examples of this extension is the goal or requirements-driven design option; instead of building the physical network and then using a network simulator, e.g., OPNET Modeler, to determine if it meets the requirements, INCOT starts with the requirements and then tries to build the physical network. INCOT does this while only using available devices, allowable services, satisfying security requirements, etc. This process is iterative, with INCOT asking further questions if necessary. INCOT continues refining the network or asking additional questions until the network can no longer be improved.

INCOT can be used independently, but has also been integrated with OPNET tools and tools derived from OPNET, i.e., NETWARS. When INCOT is utilized with OPNET or OPNET derived tools, commands are available that enables the operator to quickly modify network components via a database of known devices, add or delete connections, test for proof of adherence to policies, or perform optimization. An iterative session with INCOT helps the network engineer determine the necessary equipment, cost, end points and performance while taking into consideration geographical constraints, locations of ground facilities, forces and related space-based assets (such as data-relay communication satellites).

Graphical representations (as well as reports) are available to illustrate the quality of the network coverage provided. Finally, mouse-gestures enable the operator to zoom in on a specific geographic area, to obtain detailed information about a specific sub-network or component, and to quickly assemble alternate displays designed to support specific types of decisions.

System Design Overview

INCOT's goals include being a complement, not a competition to COTS networking tools. Therefore, to maximize the foundation provided by the commercial simulation engine, Stottler Henke teamed with OPNET Technologies: their Modeler product as well as other OPNET products provides the industry's leading network technology development tools. Modeler itself allows for design and analysis of networks, devices, protocols and applications. Modeler already provides graphical editors that mirror the structure of actual networks and network components, so INCOT leverages these as well as many other powerful facilities available in Modeler.

In addition, INCOT is not tied to OPNET products. That is, a subset of INCOT functionality can be performed without any OPNET or other tools. Since INCOT performs requirements-based automated network design that other tools do not, many aspects of this automation can be performed without other tools. For example, the inventory of equipment and the input of requirements can be performed independent of other tools. This also allows INCOT to be available to users that do not have access to OPNET products.

Figure 1 presents an overview of the major components and data flows of INCOT. The following sub-sections describe the major components of INCOT and the user interaction.

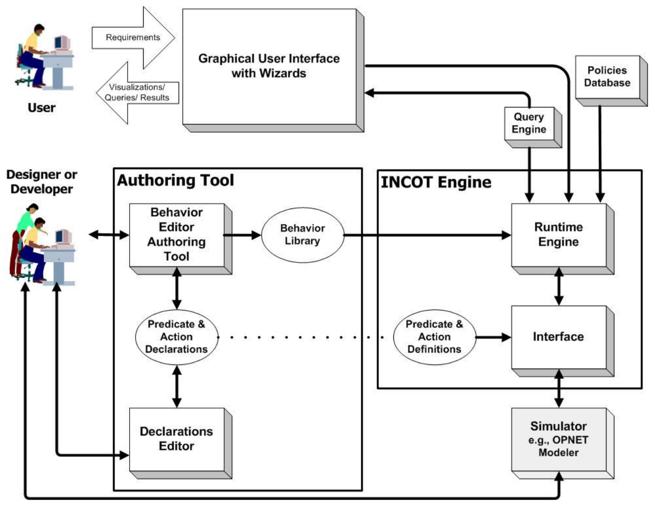

Requirements
Graphical User Interface with Wizards
Policies Database
Visualizations/ Queries/ Results
User
Query Engine
Designer or Developer
Authoring Tool
INCOT Engine
Behavior Editor Authoring Tool
Behavior Library
Runtime Engine
Predicate & Action Declarations
Predicate & Action Definitions
Interface
Declarations Editor
Simulator e.g., OPNET Modeler

Figure 1. System Design for INCOT

Graphical User Interface with Wizards —The user interacts with INCOT and indirectly with other tools, e.g., NETWARS, via the Graphical User Interface with Wizards [1]. Via the Graphical User Interface, the user provides INCOT with the network goals, with their relative priority, the environmental / geographical situation, policies, and constraint data.

Much of the initial design setup interaction provided is via a Wizard interface. That is, the user is led through the network design process via step-by-step requirements-driven hierarchical workflow. As expanded upon later, the user is first asked high-level questions, e.g., the number of locations that will be involved in the network, and then later is asked about the particulars e.g., how many secure computers are needed in a particular subnet.

INCOT Engine —From the goals, available devices, policy constraints and present environmental/geographical situation, the INCOT Engine can hypothesize on network configurations. If more information is needed from the network engineer, the INCOT Engine can formulate questions via the Query Engine that are presented to the network engineer. When the INCOT Engine has enough information to start designing the network configuration, it builds a network configuration and optionally submits it to OPNET for testing.

Policies Database — The Policies Database provides a repository for all the various policies that may need to be in effect. A default set of policies are initially active. The user has the option of relaxing certain policies and activating other policies. This can be done interactively to see if the "bending" of a policy will allow for more of the goals to be reached.

Query Engine — The Query Engine is simply an interface between the INCOT Engine and the Graphical User Interface with Wizards. Again, if more information is needed from the network engineer, the INCOT Engine can formulate questions via the Query Engine. That is, the Query Engine will take the information request from the INCOT Engine and formulate a query that is easy to understand by a network engineer.

Simulator — INCOT can interface with different OPNET and OPNET-derived simulators. For example, INCOT can work with NETWARS, a DISA product built with OPNET products. The Simulator products handle the network simulations to provide the results for any particular configuration generated by the INCOT Engine. OPNET and OPNET-derived products provide a devices database that can be customized so that INCOT will only show the inventory of devices that are available. In addition, OPNET and OPNET-derived products include an Environment / Geography Database; that is, topological information for the entire world is supported already as part of OPNET. Therefore, if INCOT proper requires any device or topological information it will query the appropriate OPNET or OPNET-derived products.

INCOT EVOLUTION

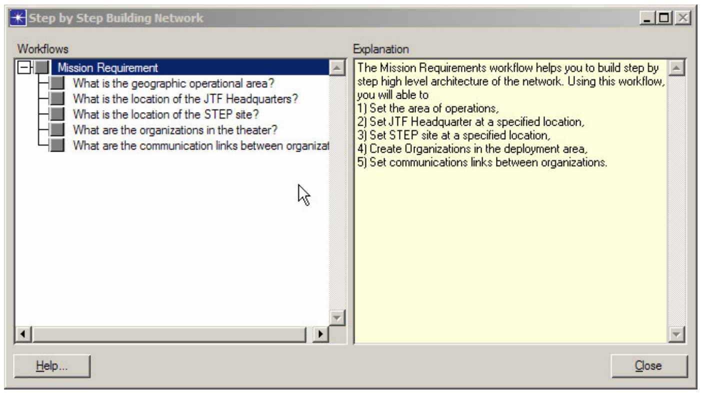
The Beta and earlier versions of INCOT were implemented using

- OPNET Modeler, with the following add-on modules,
    - Radio,
    - Terrain Modeling, which are customized via the
    - OPNET Development Kit (ODK).

The Beta software implemented a Wizard interface that was derived from the Task Manager in NETWARS. The Network Warfare Simulation (NETWARS) software is developed by the Defense Information Systems Agency (DISA). NETWARS is a communications modeling tool, built on top of OPNET tools, that enables the warfighter to credibly model tactical and operational communications demands with all the stresses that combat places on communications systems. The NETWARS Web page is located at www.disa.mil/tis/netwars. NETWARS is a component of the next-generation Joint Network Management System (JNMS). The Joint Network Management System (JNMS) is an automated software system that will provide communications planners with a common set of tools to conduct high-level planning, detailed planning and engineering, monitoring, control and reconfiguration, spectrum planning and management, and security of systems.

The Task Manager in NETWARS and the Wizard in INCOT *look* essentially the same; Figure 2 shows the Wizard in INCOT. The difference is that the Task Manager is only a Help system, it informs the user on what to do for each stage textually. The Wizard provides the same information but *also* walks the user through the whole process.

When a user starts a new project in INCOT, the Mission Requirements Workflow portion of the Wizard (displayed in Figure 2) is present. Using The Mission Requirements Workflow, the user defines the equipments available in the mission, the areas of operations, JTF and STEP sites, sites (organizations) in the operation area, the transmission system backbone, etc.


Step by Step Building Network

Workflows

- Mission Requirement
    - What is the geographic operational area?
    - What is the location of the JTF Headquarters?
    - What is the location of the STEP site?
    - What are the organizations in the theater?
    - What are the communication links between organizat

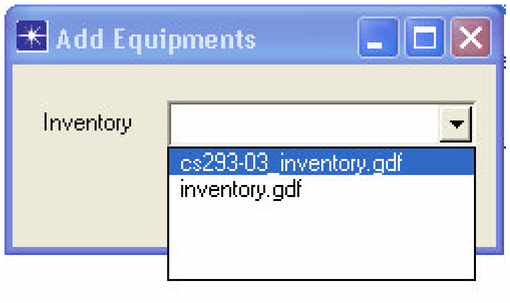
Explanation

The Mission Requirements workflow helps you to build step by step high level architecture of the network. Using this workflow, you will able to
1) Set the area of operations,
2) Set JTF Headquarter at a specified location,
3) Set STEP site at a specified location,
4) Create Organizations in the deployment area,
5) Set communications links between organizations.

Figure 2. INCOT Wizard Interface

If the user selects the first option, "What is the available equipment for the operation", the user will be taken to the

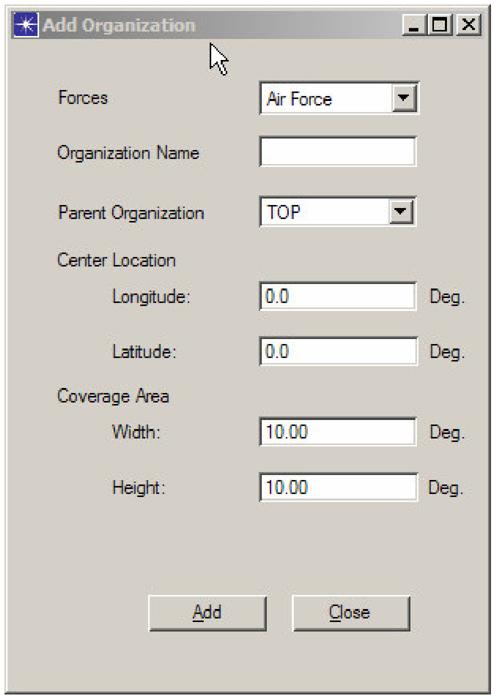
Add Equipment dialog, shown in Figure 3. The user can set which inventory to use in this mission via a pull down menu. Each inventory has a list of equipment with quantities.



cs293-03_inventory.gdf
inventory.gdf

Figure 3. Add Equipment Dialog

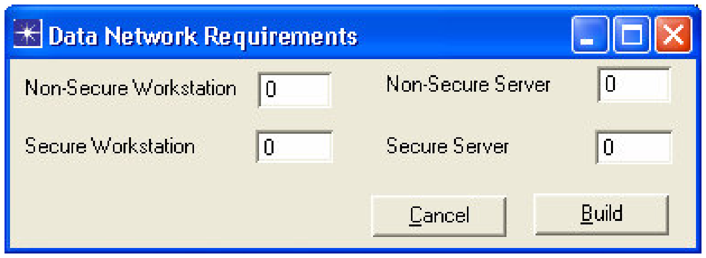
The rest of the Wizard works similarly, that is, the user selects a step in the workflow and the Wizard guides the user through the process of providing the information. As another example, when the user selects the "What are the organizations in the theater?" step, the *Add Organization* dialog will be presented as shown in Figure 4.



Forces: Air Force
Organization Name:
Parent Organization: TOP
Center Location
Longitude: 0.0 Deg.
Latitude: 0.0 Deg.
Coverage Area
Width: 10.00 Deg.
Height: 10.00 Deg.

Figure 4. Add Organization Dialog

or example, by selection *Data Requirements* the user defines the number of secure and non-secure workstations or servers in the organizations via the dialog shown in Figure 5.



Non-Secure Workstation: 0
Non-Secure Server: 0
Secure Workstation: 0
Secure Server: 0

Figure 5. Data Network Requirements Dialog

When the user has completed defining all the data and voice requirements, they can select *Auto Build Backbone* to have INCOT build the rest of the organization's network based on the data and voice requirements and the inventory of equipment that is available.

The Beta release of INCOT demonstrated how INCOT can further enhance NETWARS by evolving the Task Manager to a Wizard and adding automated network construction. More information on the Beta release may be obtained on the INCOT Web page at www.StottlerHenke.com/INCOT.

INCOT RELEASE

The released version has extended the functionality beyond the Beta functionality in three significant ways

1. TNAPS+ Integration: INCOT can now read TNAPS+ legacy data, since the Air Force and Marines still use TNAPS+.
2. Abstraction of INCOT core: INCOT can be used independent of any other product.
3. Visual Authoring Environment: INCOT's network automation algorithms are now accessible and modifiable via a visual authoring tool.

The following sections describe in further detail these INCOT capabilities

TNAPS+ INTEGRATION

The Tactical Network Analysis and Planning System Plus (TNAPS+) is a legacy PC-based tool utilized by the Air Force and other military branches for tactical communications planning and control (https://esc-digd.hanscom.af.mil/Tnaps). TNAPS+ assists the planner in building an exercise or operation database that consists of state-of-the-art commercial equipment and TRI-TAC equipment. TNAPS+ produces a series of output records describing the resulting networks and equipment configurations. The system controller can use this database, with program support, to monitor, manage, and reconfigure in-place communications. It is a detailed planning and engineering tool built around a database management system (DBMS). TNAPS+ supports tactical communications planning and control at two levels: the network level and the nodal/equipment level. Most of TNAPS+ is a Windows product with a small portion running under DOS but operating on the same database. An example of TNAPS+ Windows user interface is shown in Figure 6.

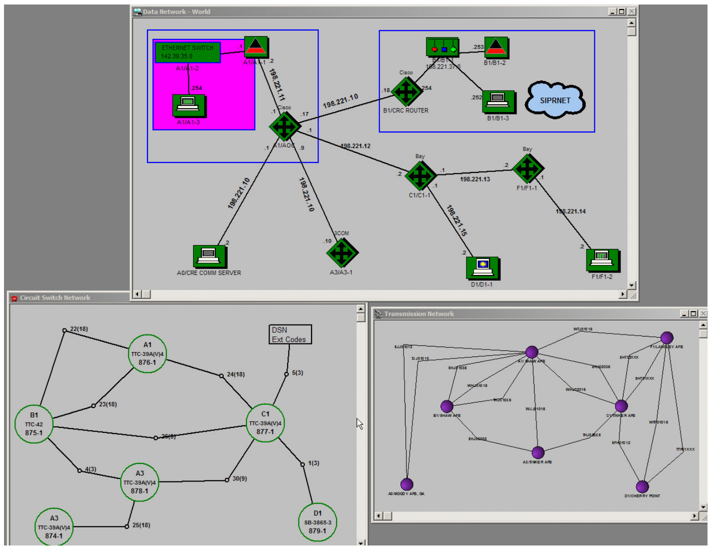


Figure 6. TNAPS+

Building the network in TNAPS+ is a requirement for many Air Force deployments; therefore users do not wish to duplicate anything in TNAPS+ and INCOT. Because of this, the INCOT has the capability to read data from TNAPS+. This allows the TNAPS+ user to benefit from much of the additional functionality provided by INCOT, OPNET and OPNET-derived products without having to rebuild the network.

This also makes training much simpler, because the Air Force presently trains users to use TNAPS+ but not OPNET tools. By reading information from TNAPS+, much less knowledge needs to be learned about OPNET tools or INCOT to be productive.

INCOT/OPNET tools have many capabilities beyond TNAPS+. An example is analysis based on location. TNAPS+ does not store geographical location information with entities, one can record textually where something is but this has no meaning to TNAPS+. So once information is imported, location information could be added so the wireless transport analysis can be performed, etc. Most other types of analysis are not available in TNAPS+ either.

INCOT CORE

As noted above, the Beta and earlier versions were integrated with OPNET or OPNET derived product. However, as INCOT became more powerful it became obvious that a standalone INCOT Core was a powerful tool in and of itself. Since INCOT performs requirements-based automated network design that other tools do not, many aspects of this automation can be performed without other tools. For example, the inventory of equipment and the input of requirements can be performed independent of other tools. This also allows INCOT to be available to users that do not have access to OPNET products or NETWARS.

INTELLIGENT AUTOMATION VISUAL AUTHORING TOOL

INCOT provides a user accessible authoring tool to manipulate the algorithms for the network layout automations. The tool is called SimBionic. First SimBionic is described in its generic form, after which its use as an intelligent automation visual authoring tool for network design is described.

SimBionic

SimBionic is a visual framework that simplifies the authoring of simulated behaviors or algorithms. SimBionic's framework consists of a canvas depicting algorithms as a finite state machine (FSM) graph, a palette of geometric objects and glyphs, and a dictionary of actions and predicates. The user defines a basic vocabulary of actions and predicates which appear as textual and geometric shapes on the canvas. The actions correspond to states in an FSM. Predicates are used to determine valid transitions between states. The basic model is extended in two major ways. First, algorithms are hierarchical in that they may invoke each other. Second, each algorithm may have a number of specializations indexed through a descriptor hierarchy. These two extensions serve to encapsulate functionality, and to selectively specialize algorithms whenever necessary without arduous re-modification of existing algorithms [2].

SimBionic employs four programming constructs:

- actions, which define all the different actions the algorithms can perform;
- algorithms (also referred to as behaviors) that string together actions and conditional logic;
- predicates, which set the conditions under which each action and algorithm will occur; and
- connectors, which control the order in which conditions are evaluated, and actions and algorithms take place.

These four constructs allow one to create algorithms that range from simple sequences to complex conditional logic. Via SimBionic's authoring canvas, see Figure 7, users can visually create algorithms by drawing actions and invoked algorithms (represented as rectangles) and conditions (represented as ovals) to interact in both simple and complex combinations via connectors (represented as arrow-shaped lines with priority numbers). This canvas also allows users to assign arbitrary expressions and comments to these elements.

Traditional development methods require algorithms to be coded within software by programmers, so it is difficult for network engineers to specify these algorithms directly. Instead, experts must describe the desired algorithms to computer programmers who then translate these descriptions into software. This multi-step approach is cumbersome, time-consuming, and error-prone. By contrast, the SimBionic Authoring Tool presents algorithms graphically, so users can create, edit, and review them easily, without programming. Graphical representations can be understood by experts and by software programmers alike, so they can speak the same language, resulting in a more effective collaborative development.

SimBionic extends the usual notion of finite state machines by making it possible for states to refer to other finite state machines hierarchically, to define modular algorithms that can be combined powerfully. SimBionic software also provides four extensions that increase the power and expressiveness of the basic engine: global and local variables, interrupt transitions, "blackboards" for sharing knowledge among finite state machines, and polymorphic indexing for run-time selection of algorithms.

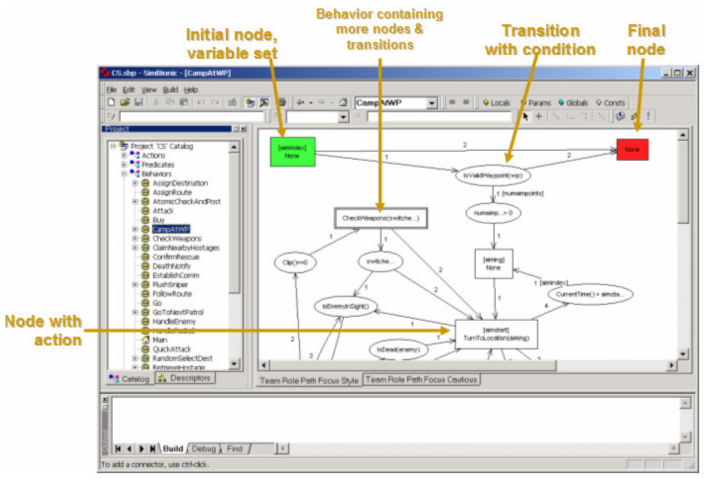

Initial node, variable set
Behavior containing more nodes & transitions
Transition with condition
Final node
Node with action

Figure 7. SimBionic Authoring Environment

Intelligent Automation Visual Authoring Tool for Network Design

SimBionic has been incorporated into INCOT and retains its two basic components: a visual editor for authoring network algorithms, and a run-time *INCOT Engine* for executing those algorithms. The *Authoring Tool* enables one to define or modify INCOT algorithms; while the run-time engine interprets all the active algorithms in parallel and invokes actions and predicates implemented within the application as specified by each algorithm. As described above a SimBionic algorithm is composed of actions, conditions, connectors and other algorithms. The following provides an example of the authoring tool's use per network design for actions and conditions.

- Actions are functions invoked from within SimBionic's algorithms to carry out activities, such as installing devices.
- Conditions check whether an entity is in a specified state. Conditions may invoke predicates to determine the state information. Predicates are functions that return a value representing the state information of an entity, such as *IsInInventory(device_model)*, which checks if the specific device model is currently in the inventory.

The INCOT algorithms can be triggered by user actions, other events in INCOT or by other algorithms.

Figure 8 shows an example of a hierarchal algorithm for setting up connections between two organizations. From the initial state, the first condition checks if the distance between two organizations is too far to be feasible to have a wired link between the two organizations. If the condition is true, it transitions to the next algorithm *Setup-MicrowaveComm(…)*, otherwise it transitions to the alternative algorithm, *SetupWiredComm(…)*. The numbers on the connections inform the FSM the priority to execute connections, if the 1st priority connection fails, then the 2nd priority connection is tried.

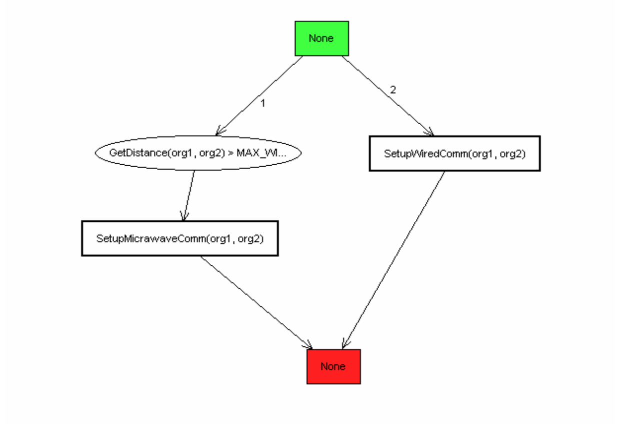

None
1
2
GetDistance(org1, org2) > MAX_WI...
SetupWiredComm(org1, org2)
SetupMicrawaveComm(org1, org2)
None

Figure 8 Connecting Two Organizations

The algorithm for the *SetupMicrowaveComm(org1,org2)* is shown in Figure 11, first it checks if there are already microwave antennas in the organizations. If there are no microwave antennas, the algorithm installs a microwave antenna based on availability in the inventory. Once microwave antennas are set up, it checks if they are in line-of-sight of each other. If they are not, it calls *UpdateAntennaPos()* function to update the antenna position incrementally and checks the line-of-sight again. This process goes on until they are in line-of-sight. It would probably be a better implementation to have the *UpdateAntennaPos()* function perform the entire process of finding the line-of-sight positions, but this example is used to demonstrate the looping capability of SimBionic.

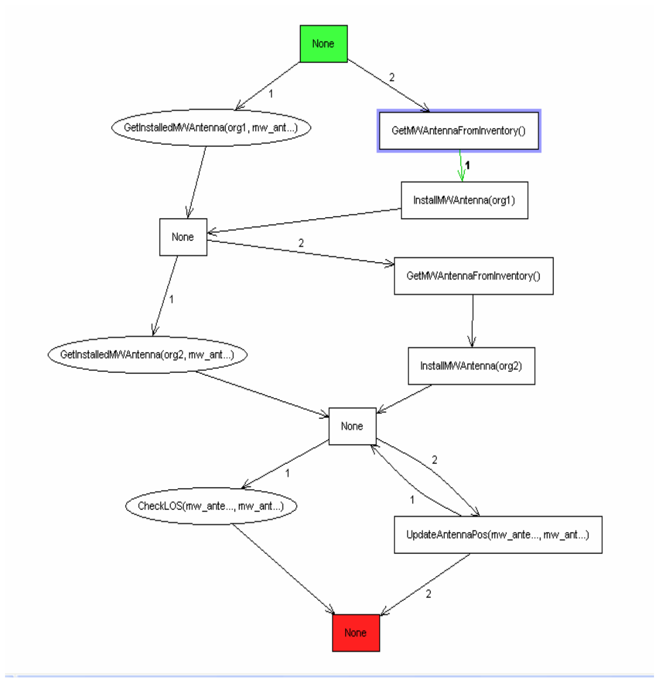

None
1
2
GetInstalledMWAntenna(org1, mw_ant...)
GetMWAntennaFromInventory()
1
InstallMWAntenna(org1)
None
2
GetMWAntennaFromInventory()
1
GetInstalledMWAntenna(org2, mw_ant...)
InstallMWAntenna(org2)
None
1
2
1
CheckLOS(mw_ante..., mw_ant...)
UpdateAntennaPos(mw_ante..., mw_ant...)
2
None

Figure 9 Setting Up Microwave Communication

Figure 10 shows another example of an algorithm that demonstrates selecting the link between device models based on the user-defined policy. The algorithm returns the best possible matching link model between source device and end device. The policy may be OPTIMUM BANDWIDTH (OPTIM…), in which case the link will satisfy the required bandwidth, or ECONOMICAL (ECONO…), in which case the user cares most about cost or it can be HIGHEST BANDWIDTH (HIGHE…) because the user may be considering future expansion. Based on the policy, the algorithm selects the best possible matching link model to use between the end devices.

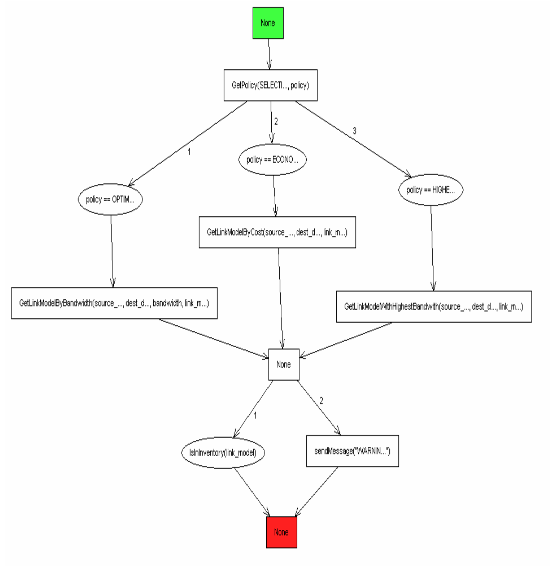

None
GetPolicy(SELECTI..., policy)
1
2
3
policy == OPTIM...
policy == ECONO...
policy == HIGHE...
GetLinkModelByCost(source_..., dest_d..., link_m...)
GetLinkModelByBandwidth(source_..., dest_d..., bandwidth, link_m...)
GetLinkModelWithHighestBandwith(source_..., dest_d..., link_m...)
None
1
2
IsInInventory(link_model)
sendMessage("WARNIN...")
None

Figure 10 Selecting Link Between End Devices

CONCLUSION

By incorporating expert knowledge of network engineering, policies, available components and their characteristics, INCOT is able to provide an interactive rapid development tool to the network engineer, while simplifying its operation for Air Force and other DoD network engineers. INCOT is greatly extending the power of OPNET offerings and OPNET derived tools. One of the greatest examples of this extension is the goal or requirements-driven design option; instead of building the physical network and then using OPNET/NETWARS to determine if it meets the requirements, INCOT starts with the requirements and then builds the physical network. INCOT does this while only using available inventory. By enhancing NETWARS concepts, INCOT will be benefiting the next generation of military networking tools, (e.g., Joint Network Management System (JNMS)), and by interfacing to TNAPS+, INCOT is allowing the Air Force, Marines and other DoD users to re-use legacy network modules and reach the next generation sooner. INCOT is available gratis now to the entire DoD community providing an automated intelligent system for designing communications networks.

REFERENCES

[1] Barfield, W., and Furness, T.A.,. Virtual Environments and Advanced Interface Design, Oxford University Press, 1995.

[2] Fu, D., R. Houlette, O. Bascara, "An Authoring Toolkit for Simulation Entities", Proceedings of the Industry/Interservice, Training, Simulation & Education Conference (I/ITSEC 2001).