



Modeling the Emergent Dynamics of a Tactical Situational Awareness Network Within an Information Operations (IO) Environment

by Brian G. Ruth and J. Dana Eckart

ARL-TR-2641

December 2001

Approved for public release; distribution is unlimited.

20020114 148

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory

Aberdeen Proving Ground, MD 21005-5068

ARL-TR-2641

December 2001

Modeling the Emergent Dynamics of a Tactical Situational Awareness Network Within an Information Operations Environment

Brian G. Ruth

Survivability/Lethality Analysis Directorate, ARL

J. Dana Eckart

Convergent Computing, Inc.

Abstract

A model developed to analyze the global emergent dynamics of situational awareness (SA) dissemination within a digitized brigade undergoing hostile information operations (IO) stress events is presented. Networked battlefield platforms are modeled as interconnected tactical internet nodes through the use of cellular automata. The cellular automata form a discrete spatially extended dynamical system consisting of a parallel networked lattice of computational cells in two dimensions. The global impact of localized IO stress events on SA dissemination can then be analyzed as a function of time by mapping the SA dissemination architecture onto the cellular automata lattice. The lattice cells can represent tactical internet client or server platforms. This model demonstrates the fluctuating patterns of situational "self-awareness" (i.e., a friendly platform's situational awareness of all other friendly platforms within its area of operations) which emerge across the digitized brigade on a global scale and also illustrates the emergent dynamics of SA dissemination within a "movement-to-contact" brigade structure.

Executive Summary

In this report, a model for analyzing the emergent global impact of hostile information operations (IO) stress directed at the networked platforms making up a digitized brigade-level situational awareness (SA) architecture is presented. Since the model specifically addresses a platform- to network-level mapping of structure and behavior, the complexity of message traffic patterns (which arises from intra-platform processes embedded within the platform-to-network-level mapping) cannot be properly quantified and analyzed. Thus, the current model focuses on the dynamic robustness of the brigade SA network topology (i.e., the *potential* ability of the network structure to perpetuate SA dissemination under disruptive stress) within a hostile IO environment.

A brigade-level tactical communications network, as implemented through the U.S. Army's Force XXI Battle Command Brigade-and-Below (FBCB2), consists of two distinct subarchitectures for the parallel dissemination of both situational awareness (SA) and command and control (C2) message traffic through the use of enhanced position location reporting system (EPLRS) data radios. The SA subarchitecture within a digitized brigade uses host platforms functioning as position servers to collect and distribute the data vertically and horizontally across a predetermined hierarchy of radio nets, which include local single channel ground airborne radio system (SINCGARS) based nets as well as the EPLRS-based battalion-level carrier sense multiple access (CSMA) nets and brigade-wide multisource group (MSG) net. This SA dissemination architecture provides a framework within which to construct a cellular-automata-based SA network topology model.

In general, a *cellular automata* (CA) consists of a possibly infinite, n -dimensional regular lattice of cells. Each cell can be in a measurable state chosen from a finite alphabet (i.e., a set of possible cell states). The states of all cells in the lattice are updated simultaneously in discrete time steps through the use of a transition function, which takes as its input the current state of the center cell and some finite collection of nearby cells that lie within a finite distance, collectively known as a *neighborhood*. Cell neighborhoods may be either fixed or variable over time (but fixed within a time step); these neighborhoods are usually defined to be local to a center reference cell, but can extend beyond this conventional limit if necessary. This last condition is necessary to model the SA dissemination network topology within a brigade, where neighborhoods must span across all lattice scales (i.e., local to global).

The first step in building the SA network is to spatially configure a brigade structure relative to a CA lattice; such an example network is a generic "movement to contact" brigade structure. Within this generic structure, each of the 612 blue squares represents a networked platform node which occupies a lattice cell (in the current version of the model, nodes are ground vehicles which remain fixed relative to one another), where a node is designated as either a SINCGARS net client, a self server, a CSMA server, or an MSG server.

The next step in model development is to define the spatial scope of situational awareness operationally required of a platform in order to formulate one or more CA neighborhoods relative to the lattice cell which the platform occupies. Applying an echelon-dependent "area of operations" (AO) rectangle to each platform within the CA brigade model defines an AO *neighborhood* relative to the overall CA lattice, where the AO neighborhood is a sublattice surrounding a center node cell which defines the region of friendly neighboring node cells which the center node must be situationally aware of.

Once the spatial scope of situational awareness has been delineated and implemented for each node cell within the brigade (thus creating an AO-based network topology), a quantitative measure of the degree to which a node is situationally aware of other friendly neighbor nodes can now be defined. Within the context of the CA model, the situational awareness of a TI node cell is a collective measure of the state of the communication channels which exist between the reference node and all neighboring friendly nodes within the reference node's AO neighborhood. The state of the local network topology relative to the i^{th} node can be represented by a metric called the *SA connectivity* $C_i(t_n)$ that indicates the fraction of neighbor nodes within the i^{th} node's AO neighborhood that can transmit SA messages to the i^{th} node at time step t_n .

There are five data processing filters that have been coded into the CA model for analyzing time series data.

- (1) The first data filter calculates $\langle h^{\text{stress}}(t_n) \rangle$, the normalized perturbative stress that is introduced into the entire brigade system via localized IO stress.
- (2) The second data filter calculates globally averaged SA connectivity $\langle C(t_n) \rangle$ as a function of scenario time, where an average value is calculated at each point in discrete simulation time across all node platforms within the CA brigade.
- (3) The third data filter calculates the five fitness metrics $P_{\text{very high}}(t_n)$, $P_{\text{high}}(t_n)$, $P_{\text{medium}}(t_n)$, $P_{\text{low}}(t_n)$, and $P_{\text{very low}}(t_n)$, where $P_{\text{fitness}}(t_n)$ is the fraction of network nodes occupying a particular "fitness" state at time step t_n . The five fitness states are defined by the intervals $C_i(t_n) \leq 0.1$ (*very low*), $0.1 < C_i(t_n) \leq 0.4$ (*low*), $0.4 < C_i(t_n) \leq 0.6$ (*medium*), $0.6 < C_i(t_n) \leq 0.9$ (*high*), and $C_i(t_n) > 0.9$ (*very high*).

- (4) The fourth data filter calculates the *state entropy* $H(t_n)$, a quantitative measure of the disorder/volatility within the network at time step t_n , and the *temporal entropy* $H(t_n, t_{n-1})$, a quantitative measure of the temporal disorder between sequential network states measured at t_{n-1} and t_n .
- (5) The last data filter presents fitness metric data in the form of a *return map*, which is a 2-D scatter plot of $P_{fitness}(t_{n+1})$ vs. $P_{fitness}(t_n)$ for each fitness state.

Taken together, these filters can be used to analyze the emergent dynamics of an SA network exposed to IO stress.

The final step in developing the CA model is the addition of some form of IO stress which directly interacts with a node cell in order to perturb its communication capabilities. This stress can arise either from a stochastic parameter embedded within a nodal state vector or an external neighboring cell (representing a localized IO threat source) that generates a local "stress field." Examples of both types of IO stress were added to the CA model, and subsequent simulation results were analyzed.

The first type of IO stress was modeled as a stochastic process which acts to perturb the functional communication states of each node within the SA network. The dynamics of this stochastic process are described by a stationary Markov chain which is applied uniformly and independently to each of the nodes within the digitized brigade. The stochastic process then acts to transition each network node between functional, transient, or permanent dysfunction, and functional but "spoofed" states. This type of stress is similar to the mean field approximation from statistical physics, where a particular element of a multi-element system is subjected to the average influence of a global field interacting with the rest of the system. In this sense, the "mean field" IO stress is at best a first-order approximation to a real multi-threat IO environment, serving to demonstrate the emergent behavior of the brigade's SA network topology under a coherent hypothetical rather than realistic stress.

Simulation results using the mean field IO stress model show that message flooding (resultant from Global Positioning System [GPS] spoofing) is the dominant cause of SA connectivity degradation, since extreme levels of communication channel latency (resulting from server message queue saturation) are also assumed to cut internodal communication channels until latency levels drop below a threshold level. However, the mean field model has shortcomings in two different areas. First, since the C2 network structure within the brigade has deliberately been decoupled from the SA network structure, the model can significantly underpredict the impact of GPS spoofing on nodal SA connectivity. Second, the model fails to realistically consider correlations between spatially dispersed stress sources, as well as the resultant correlations between SA connectivity levels associated with impacted nodes.

A more realistic form of IO stress arises from perturbations to nodal communication due to localized external fields that are generated by electromagnetic (EM) sources. In this second stress model, the CA dynamics are added by introducing spatially dispersed source cells representing generic radio frequency (RF) "jammer bombs" that act to locally jam platform-mounted radios passing by. Thus, a node is jammed if it passes within a predefined jamming neighborhood of a jammer bomb; alternately, a node is jammed if and only if there is at least one jammer bomb within an identical jamming neighborhood relative to the target node (where the target node is at the center of the translated neighborhood). Since the localized jammer bomb/node interactions act to perturb nodal reception capability states, this in turn perturbs nodal SA connectivity levels throughout the CA brigade.

A jammer bomb scenario is defined with two phases: (1) the digitized brigade moves through a field of jammer bombs previously planted by hostile forces, and (2) hostile unmanned aerial vehicles flying over the advancing brigade deliver a barrage of additional jammer bombs. Twenty simulation trials of the two-phase jammer bomb scenario are run using the CA model, where each trial is seeded with a different random number (needed to stochastically vary the distribution of jammer bombs upon initial insertion into a simulation). In addition, four variants to the original jammer bomb scenario are also simulated, which include (1) reduced AO neighborhoods, (2) reduced jammer bomb lifetimes, (3) a combination of reduced jammer bomb neighborhoods and increased average bomb density, and (4) jammer bomb susceptibility gradients.

Simulation results from all of the jammer bomb scenarios indicate the following set of shared characteristics.

- The dynamic fraction of jammed node within the brigade increases to a maximum value, temporarily remains in a state of quasi-equilibrium, and then steadily decreases back down to a level of 0.
- The dynamic SA connectivity averaged across all nodes within the brigade decreases down to a minimum value, again temporarily remains in a state of quasi-equilibrium, and then steadily increases back up to a level of 1.
- The SA fitness $P_{very\ low}(t_n)$ time series roughly track the scenario-respective $\langle h^{stress}(t_n) \rangle$ time series in terms of metric amplitude.
- The SA fitness $P_{very\ high}(t_n)$ time series roughly track the scenario-respective $\langle C(t_n) \rangle$ time series in terms of relative time series profile (with differences in metric amplitude).

- The SA fitness $P_{high}(t_n)$ time series profile is a mirror-image reflection of the respective SA fitness $P_{very\ high}(t_n)$ time series profile.
- The dynamic state entropy $H(t_n)$ and temporal entropy $H(t_n, t_{n-1})$ demonstrate separation between global brigade, CSMA net, and local net node populations.

However, the jammer bomb scenario variants are distinguishable by their distinctive emergent behaviors as displayed within their respective $P_{very\ high}(t_n)$ time series. Although variations in AO neighborhoods and jammer bomb lifetimes lead to differences in collective structure as displayed within the return maps, the most noticeable factor observed within jammer bomb simulations appears to be the size of the jammer bomb neighborhood and its impact on $P_{very\ high}(t_n)$. These latter results suggest that the greater the jamming range of a stress source and the more ubiquitous this type of source, the more coordinated the overall network response (i.e., the less resistant is the network to associated perturbations to SA connectivity).

The spatio-temporal dynamics of other types of dispersed IO stress sources that modify nodal transmit/receive functionality are likely to create similar modes of emergent global behavior in situational awareness levels which cannot be anticipated by studying isolated platforms or even local area networks (LANs). Given that the threat/target interaction dynamics of an IO stress source can be modeled, it is a fairly straightforward process to program the stress source into the CA code within the context of a scenario. There is also no limitation on the number of different types of IO stress sources which can be programmed into a single scenario given that the discrete spatio-temporal interaction dynamics are known.

INTENTIONALLY LEFT BLANK.

Contents

Executive Summary	iii
List of Figures	xi
List of Tables	xv
1. Introduction	1
1.1 Purpose	1
1.2 Background	1
1.2.1 General	1
1.2.2 Threat	2
1.3 Scope	3
2. The Situational Awareness Network Model	3
2.1 Cellular Automata	3
2.2 The <i>Cellular</i> Simulation System	4
2.3 Building the Cellular Automata Network	5
2.3.1 The Network Cell	5
2.3.2 The Cellular Automata Brigade	5
2.4 Measuring Situational Awareness	6
2.4.1 Area of Operations Neighborhoods	6
2.4.2 Situational Awareness Connectivity	7
2.5 Introducing Information Operations Stress	13
2.6 Analyzing the Emergent Network Dynamics	13
2.6.1 Normalized Network Stress $\langle h^{stress}(t_n) \rangle$	14
2.6.2 Globally Averaged Situational Awareness Connectivity $\langle C(t_n) \rangle$	14
2.6.3 Situational Awareness Fitness	14
2.6.4 Course-Grained Entropy	16
2.6.5 Return Map	17

3. Information Operations Simulations	17
3.1 Mean Field Stress.....	17
3.2 Localized External Stress Fields	25
3.2.1 Jammer Bomb Scenario.....	25
3.2.2 Simulation Results.....	26
3.2.3 Echelon-Specific Nodal Sampling.....	37
3.2.4 Sensitivity Investigation of Model Parameters	38
3.2.4.1 Reduced AO Neighborhood Scenario.....	39
3.2.4.2 Reduced Jammer Bomb Lifetime Scenario	43
3.2.4.3 Combined Moore Neighborhood/Increased Bomb Density Scenario.....	48
3.2.5 Jammer Bomb Susceptibility Gradient.....	54
3.2.6 Disussion of Results	60
4. Conclusions	63
5. References	65
Appendix A. Dynamics of Situational Awareness Message Servers	67
Appendix B. Cellang Source Code for the Situational Awareness Network Model With Jammer Bomb Stress Sources	73
Appendix C. Data Filter Source Code	91
Distribution List	107
Report Documentation Page	111

List of Figures

Figure 1. Situational awareness dissemination architecture within a digitized brigade.....	3
Figure 2. CA cell neighborhoods in two dimensions: (a) von Neumann neighborhood and (b) Moore neighborhood.....	4
Figure 3. Cellular automata configuration of a movement to contact brigade structure.....	6
Figure 4. Area of operations neighborhood scaling within the CA brigade.....	7
Figure 5. Local network topology of a center receiving node's AO neighborhood.....	8
Figure 6. Connectivity distribution $P(N)$ for the CA brigade.....	12
Figure 7. Application of Guzie's five-state partition to the SA connectivity of the i^{th} node.....	15
Figure 8. Dynamics of the SA connectivity state vector for the CA brigade.....	15
Figure 9. Dynamics of the chaotic 1-D logistic map $x_{n+1} = 4x_n(1-x_n)$ presented in (a) standard time series and (b) return map formats.....	18
Figure 10. Representation of an inter-nodal communication channel as a funnel collecting and dispersing balls through a trap door.....	21
Figure 11. Average SA connectivity for the brigade network exposed to a mean field stress which induces nodal transient/permanent dysfunction.....	23
Figure 12. Average SA connectivity for the brigade network exposed to a mean field stress which induces nodal GPS spoofing.....	23
Figure 13. Jamming neighborhood for a generic RF jammer bomb.....	26
Figure 14. Potential jammer bomb site/target cells within the CA brigade.....	27
Figure 15. CA jammer bomb simulation snapshots: (a) time = 3 min; (b) time = 14 min; (c) time = 43 min; (d) time = 68 min.....	29
Figure 16. Fraction of jammed nodes as a function of scenario time t_n , for the CA brigade in the jammer bomb scenario.....	31
Figure 17. Dynamic globally averaged SA connectivity as a function of scenario time t_n , for the CA brigade in the jammer bomb scenario.....	32

Figure 18. $P_{very\ high}(t_n)$, $P_{high}(t_n)$, $P_{medium}(t_n)$, $P_{low}(t_n)$, and $P_{very\ low}(t_n)$ as a function of scenario time t_n , for the CA brigade in the jammer bomb scenario.....	33
Figure 19. Superposition of $P_{very\ high}(t_n)$ (green points) and $P_{very\ low}(t_n)$ (red points) time series data from all 20 simulation trials.....	33
Figure 20. State entropy as a function of scenario time t_n , for the CA brigade in the jammer bomb scenario.....	35
Figure 21. Temporal entropy as a function of scenario time t_n , for the CA brigade in the jammer bomb scenario.....	35
Figure 22. Return map for the $P_{very\ high}(t_n)$ (green points) and $P_{very\ low}(t_n)$ (red points) time series.....	37
Figure 23. Fraction of jammed nodes as a function of scenario time t_n , for brigade (Bde), battalion (Bn), and combined company/platoon (Co-Plt) echelon nodes.....	38
Figure 24. Globally averaged SA connectivity as a function of scenario time t_n , for brigade (Bde), battalion (Bn), and combined company/platoon (Co-Plt) echelon nodes.....	39
Figure 25. Reduced AO neighborhood scaling within the CA brigade.....	40
Figure 26. Connectivity distribution $P(N)$ for the CA brigade with reduced AO neighborhoods.....	40
Figure 27. Globally averaged SA connectivity as a function of scenario time t_n , for both the original jammer bomb and reduced AO neighborhood scenarios.....	41
Figure 28. $P_{very\ high}(t_n)$, $P_{high}(t_n)$, $P_{medium}(t_n)$, $P_{low}(t_n)$, and $P_{very\ low}(t_n)$ as a function of scenario time t_n , for the reduced AO neighborhood scenario.....	42
Figure 29. State entropy as a function of scenario time t_n , for the reduced AO neighborhood scenario.	42
Figure 30. Temporal entropy as a function of scenario time t_n , for the reduced AO neighborhood scenario.....	43
Figure 31. Superposition of $P_{very\ high}(t_n)$ (green points) and $P_{very\ low}(t_n)$ (red points) time series data from all 20 simulation trials from the reduced AO neighborhood scenario.	44
Figure 32. Return map for the $P_{very\ high}(t_n)$ (green points) and $P_{very\ low}(t_n)$ (red points) time series from the reduced AO neighborhood scenario.	44
Figure 33. Fraction of jammed nodes as a function of scenario time t_n , for both the original and reduced jammer bomb lifetime scenarios.	45
Figure 34. Globally averaged SA connectivity as a function of scenario time t_n , for the reduced jammer bomb lifetime scenario.....	46

Figure 35. $P_{very\ high}(t_n)$, $P_{high}(t_n)$, $P_{medium}(t_n)$, $P_{low}(t_n)$, and $P_{very\ low}(t_n)$ as a function of scenario time t_n , for the reduced jammer bomb lifetime scenario.....	46
Figure 36. State entropy as a function of scenario time t_n , for the reduced jammer bomb lifetime scenario.....	47
Figure 37. Temporal entropy as a function of scenario time t_n , for the reduced jammer bomb lifetime scenario.	47
Figure 38. Superposition of $P_{very\ high}(t_n)$ (green points) and $P_{very\ low}(t_n)$ (red points) time series data from all 20 simulation trials from the reduced jammer bomb lifetime scenario.....	49
Figure 39. Return map for the $P_{very\ high}(t_n)$ (green points) and $P_{very\ low}(t_n)$ (red points) time series from the reduced jammer bomb lifetime scenario.....	49
Figure 40. Reduced jamming neighborhood consisting of eight neighbor cells.....	50
Figure 41. Comparison of fraction of jammed nodes as a function of time t_n for both the original jammer bomb and combined Moore neighborhood/increased bomb density scenarios.....	51
Figure 42. Dynamic globally averaged SA connectivity for the combined Moore neighborhood/increased bomb density scenario.....	51
Figure 43. $P_{very\ high}(t_n)$, $P_{high}(t_n)$, $P_{medium}(t_n)$, $P_{low}(t_n)$, and $P_{very\ low}(t_n)$ as a function of time t_n , for the combined Moore neighborhood/increased bomb density scenario.....	52
Figure 44. State entropy as a function of time t_n , for the combined Moore neighborhood/increased bomb density scenario.	53
Figure 45. Temporal entropy as a function of time t_n , for the combined Moore neighborhood/increased bomb density scenario.....	53
Figure 46. Superposition of $P_{very\ high}(t_n)$ (green points) and $P_{very\ low}(t_n)$ (red points) time series data from the combined Moore neighborhood/increased bomb density scenario.	55
Figure 47. Return map for the $P_{very\ high}(t_n)$ (green points) and $P_{very\ low}(t_n)$ (red points) time series from the combined Moore neighborhood/increased bomb density scenario.	55
Figure 48. The seven concentric jamming neighborhoods used to define the discrete susceptibility gradient.....	56
Figure 49. Comparison of fraction of jammed nodes as a function of time within the susceptibility gradient and original jammer bomb scenarios.....	57
Figure 50. Dynamic globally averaged SA connectivity for the susceptibility gradient scenario.	57

Figure 51. $P_{very\ high}(t_n)$, $P_{high}(t_n)$, $P_{medium}(t_n)$, $P_{low}(t_n)$, and $P_{very\ low}(t_n)$ as a function of time t_n for the susceptibility gradient scenario.....	58
Figure 52. State entropy as a function of time t_n for the susceptibility gradient scenario.....	59
Figure 53. Temporal entropy as a function of time t_n for the susceptibility gradient scenario.....	59
Figure 54. Superposition of $P_{very\ high}(t_n)$ (green points) and $P_{very\ low}(t_n)$ (red points) time series data from the susceptibility gradient scenario.	61
Figure 55. Return map for the $P_{very\ high}(t_n)$ (green points) and $P_{very\ low}(t_n)$ (red points) time series from the susceptibility gradient scenario.....	61
Figure A-1. Dynamics of outbound SA message processing within a CSMA server.....	68
Figure A-2. Dynamics of outbound SA message processing within an MSG server.	69
Figure A-3. Dynamics of incoming SA message processing within a CSMA server.....	70

List of Tables

Table 1. Definitions of $J_{ij}(t_n)$ and $J_{ji}(t_n)$ for all possible SA communication channels.....	9
---	---

INTENTIONALLY LEFT BLANK.

1. Introduction

1.1 Purpose

The cellular-automata-based network model presented in this report was developed to analyze the emergent global dynamics of situational awareness (SA) dissemination within a digitized brigade exposed to several different types of hostile information operations (IO) stress (U.S. Department of the Army 1996b).

1.2 Background

1.2.1 General

A *complex system* can be defined as a group of many interacting parts functioning as a unified whole. It is distinguishable from its environment by recognizable boundaries, where the global behavior of the whole is generally not equal to the sum of the parts. Even if the individual behavior of an isolated part is linear, the collective behavior that emerges globally through networked interactions and interdependencies among the parts is often nonlinear. This collective behavior arises from the detailed structure, behavior, and interpart relationships as they are defined on a finer scale, all of which dynamically evolve within the context of an irreducible whole.

To analyze the U.S. Army's evolving tactical internet (TI) as a complex system operating within an equally complex IO environment, the following three distinct scales define the structure, behavior, and relationships: (1) the *microscale*, which addresses interactions and interdependencies between information system (IS) components (e.g., 1553b data bus, combat net radios, personal computers) within a platform, (2) the *mesoscale*, which addresses local interactions/interdependencies between networked battlefield platforms, and (3) the *macroscale*, which addresses the holistic global behavior of collective "system-of-systems" structures such as the TI (zum Brunnen et al. 2000). Usually, IO threat/target system interactions occur at the micro- and/or mesoscale levels, affecting information flows between IS components within a platform, and/or flows between locally networked platforms, respectively. In the current model, the emergent global or macroscale impact of IO stress, which is directed at the mesoscale networked platforms making up a digitized brigade-level situational awareness (SA) architecture is analyzed. Since the model specifically addresses a meso- to macroscale mapping of structure and behavior, the complexity of message traffic patterns (which arises from micro- to mesoscale processes embedded within the mesoscale/macroscale mapping) cannot be properly quantified and analyzed. Thus, this model focuses on the dynamic robustness of

the brigade SA network topology (i.e., the *potential* ability of the network structure to perpetuate SA dissemination under disruptive stress) within a hostile IO environment.

A brigade-level tactical communications network, as implemented through the U.S. Army's Force XXI Battle Command Brigade-and-Below (FBCB2), consists of two distinct subarchitectures for the parallel dissemination of both situational awareness (SA) and command and control (C2) message traffic using enhanced position location reporting system (EPLRS) data radios. The EPLRS-based radio nets provide contention-free routing for both SA and C2 traffic down to local single channel ground airborne radio system (SINCGARS) nets which are also known as stub nets (since they function as junction stubs for the SA and C2 subarchitectures) (U.S. Department of the Army 1999).

The SA subarchitecture within a digitized brigade uses host platforms functioning as position servers to collect and distribute the data vertically and horizontally across a predetermined hierarchy of radio nets. These radio nets include local SINCGARS-based nets, as well as the EPLRS-based battalion-level carrier sense multiple access (CSMA) nets and the brigade-wide multisource group (MSG) net (see Figure 1) (U.S. Department of the Army 1999; U.S. Army Communication Electronics Command 1999). Any digitized platform equipped with an EPLRS radio unit can function as a server. These servers include (a) CSMA servers that route SA traffic upward from their local SINCGARS net to their CSMA net and downward from both CSMA and brigade-wide MSG nets to local nets, (b) MSG servers that route traffic from a CSMA net upward to the MSG net and (c) self servers that broadcast self-generated SA messages directly onto their CSMA community net and are thus not members of a local SINCGARS net. Whereas CSMA server selection is dynamically updated amongst all capable platforms within a local SINCGARS community (although each community must have at least a primary and alternate server), primary and alternate MSG servers are predetermined. In addition, all platforms equipped with EPLRS radio units receive all MSG net broadcasts within the brigade, as well as all broadcasts within the platform's respective CSMA net community (note that, unless they are CSMA or self servers, these platforms still require a local CSMA server to broadcast self-generated SA messages outside of their local SINCGARS net). This SA dissemination architecture provides a framework within which to construct a cellular-automata-based SA network topology model.

1.2.2 Threat

The threats addressed within this report are information operation events which result in either denial of service or message flooding within a hierarchical digitized brigade network.

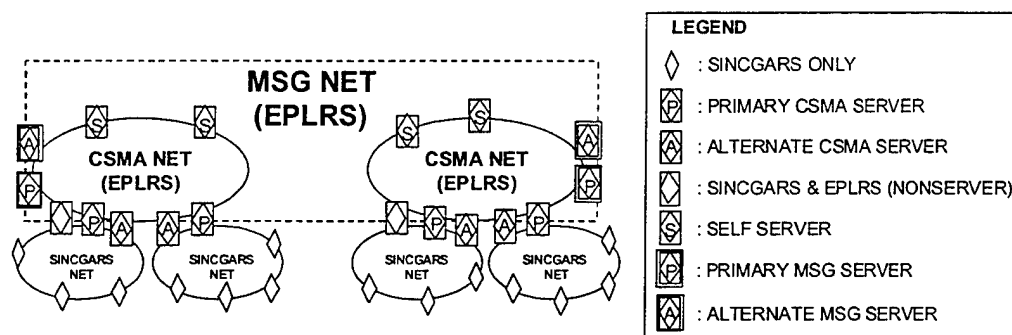


Figure 1. Situational awareness dissemination architecture within a digitized brigade.

1.3 Scope

The applicability of the SA dissemination model presented in this report is constrained by the following assumptions.

- Networked platform nodes within a digitized brigade receive SA update messages from other platform nodes only (i.e., network control stations [NCS] are *not* modeled).
- IO stress will only impact a node's ability to transmit and/or receive SA messages from other neighboring nodes and not affect internal information flows within a platform node.
- The spatial configuration of platform nodes relative to one another remains fixed throughout a simulation (i.e., platforms do not move relative to each other).

2. The Situational Awareness Network Model

2.1 Cellular Automata

In general, a *cellular automata* (CA) (von Neumann 1966) consists of a possibly infinite, n -dimensional regular lattice of cells. Each cell can be in a measurable state chosen from a finite alphabet (i.e., a set of possible cell states). The states of all cells in the lattice are updated simultaneously in discrete time steps using a transition function, which takes as its input the current state of the center cell and some finite collection of nearby cells that lie within a finite distance, collectively known as a *neighborhood*. Figure 2 displays two well-known CA cell neighborhoods in two dimensions, where C = center cell and N = neighbor cell. Cell neighborhoods may be either fixed or variable over time (but fixed within a

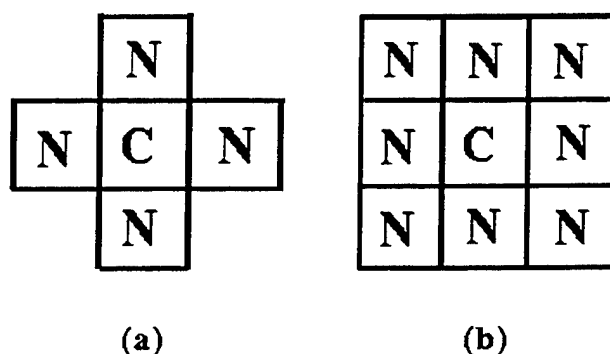


Figure 2. CA cell neighborhoods in two dimensions: (a) von Neumann neighborhood and (b) Moore neighborhood.

time step); these neighborhoods are usually defined to be local to a center reference cell, but can extend beyond this conventional limit if necessary. This last condition is necessary to model the SA dissemination network topology within a brigade, where neighborhoods must span across all lattice scales (i.e., local to global).

2.2 The *Cellular* Simulation System

The SA network topology model is implemented using the *Cellular* simulation system, an easy-to-use toolkit for the modeling of physical systems with cellular automata. The *Cellular* system consists of several modules:

- A programming language, *Cellang*, and associated compiler, *cellc*.
- An abstract virtual cellular automata machine (avcam) for *Cellang* code execution.
- Cellview, a program for viewing simulation results in either two-dimensional (2-D) sections or in three dimensions with variable perspective.

Compiled *Cellang* programs can be run with input provided at any specified time during the execution. The results of an execution can either be viewed graphically, as an output stream of cell locations and values, or passed through a custom filter before being reported. The output stream, or a similar stream produced by a custom filter, can also be directed into *cellview* for viewing, or may be passed through other programs that compile statistics, massage the data, or merely act as a valve to control the flow of data from the cellular automata program to the viewer. For more detailed information on the *Cellular* toolkit, see Eckart (1992a) and Eckart (1992b).

2.3 Building the Cellular Automata Network

2.3.1 The Network Cell

The first step in building the CA-based SA network is to define the attributes of a node platform within the network. Within the context of a CA lattice, a node is defined as a state vector that represents a networked platform occupying a cell within a 2-D CA lattice. A nodal state vector maintains the following information fields:

- node cell location within the lattice (using Cartesian coordinates),
- nodal communication capability state (transmission and reception functionality),
- platform military echelon,
- membership in specific local SINCGARS and/or CSMA radio nets,
- server status (i.e., node is either a SINCGARS net client, a self server, a CSMA server, or an MSG server),
- nodal SA connectivity (see section 2.4.2), and
- nodal SA fitness (see section 2.6.3).

When appropriate, the nodal state vector can also keep running totals of SA messages received from the various radio nets to which a node belongs. In the current version of the network model, all nodes are assumed to be ground vehicles which remain spatially fixed relative to one another.

2.3.2 The Cellular Automata Brigade

The next step in building the network is to spatially configure a brigade structure relative to the CA lattice; an example network is the generic movement to contact brigade structure (U.S. Department of the Army 1996a) depicted in Figure 3. Each of the 612 blue squares in the figure represents a networked platform node that occupies a unique (x, y) position within the CA lattice. Each of these positions is assigned a nodal state vector that is associated with the platform node occupying the corresponding lattice cell. Server nodes are mutually coordinated in that an alternate CSMA or MSG server will assume the server identity if the primary server becomes dysfunctional. This identity will be used only to construct an SA network topology within the context of a CA lattice. Also, the brigade layout depicted in Figure 3 serves only as an illustrative example that approximates a movement to contact structure; a real brigade layout (which can also be programmed for simulation by the CA model) would likely be less spatially ordered than the neat "building block" configuration shown in the figure (Wuerz 2000).

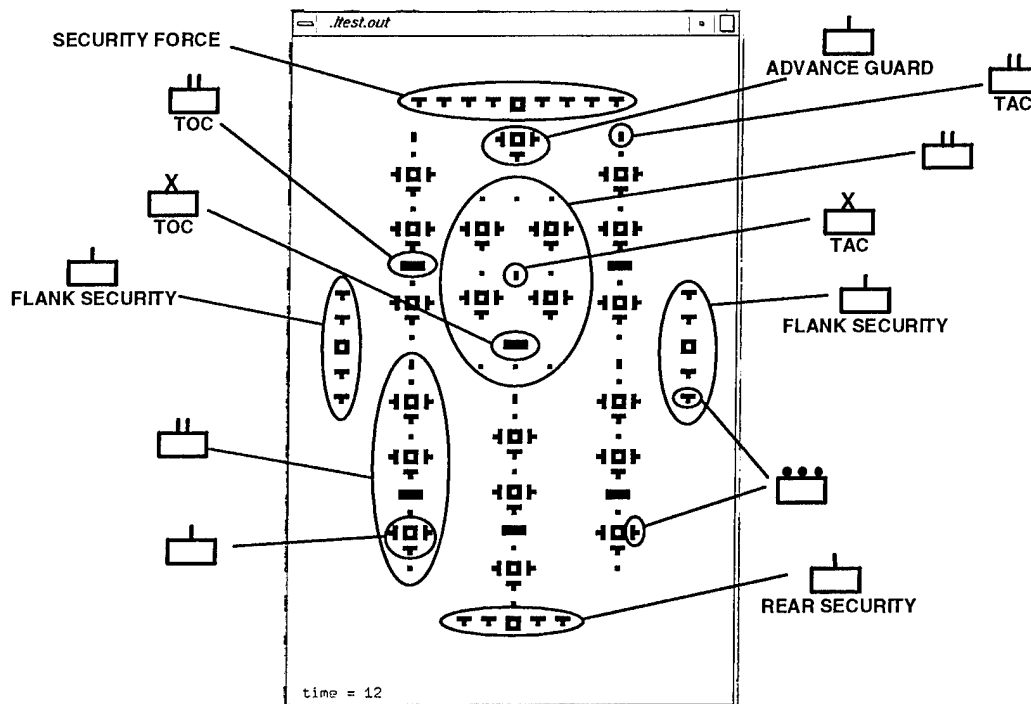


Figure 3. Cellular automata configuration of a movement to contact brigade structure.

2.4 Measuring Situational Awareness

2.4.1 Area of Operations Neighborhoods

The next step in model development is defining the spatial scope of situational awareness that is operationally required of a platform in order to formulate one or more CA neighborhoods relative to the lattice cell the platform occupies. In the FBCB2 Operational Requirements Document (ORD), the key performance parameter (KPP) relevant to SA functionality states that the system shall "...display friendly positions horizontally within a unit, two echelons up and down, and adjacent unit location one echelon down" (U.S. Army Training and Doctrine Command 1998). To allow platforms equipped only with SINCGARS to meet this KPP, CSMA servers filter incoming SA traffic based on an area of operations (AO) rectangle drawn around a client platform (with the client at the rectangle center), where rectangle area increases with higher client platform echelon. In this approach, a CSMA server will only deliver messages to a client which originate from friendly neighboring platforms within the client's AO rectangle; this is done to prevent message saturation within the local SINCGARS net (U.S. Army Communication Electronics Command 1999).

Applying an echelon-scaled AO rectangle (which, in this application, is constrained to an equal-sided square) to each platform within the CA brigade

model defines an *AO neighborhood* relative to the overall CA lattice. The AO neighborhood is a sublattice surrounding a center node cell that defines the region of friendly neighboring node cells which the center node must be situationally aware of. Figure 4 displays the four AO neighborhoods measuring 20 km $\times$ 20 km, 15 km $\times$ 15 km, 10 km $\times$ 10 km, and 5 km $\times$ 5 km, as they are defined for brigade, battalion, company, and platoon echelon node platforms, respectively (U.S. Army Communication Electronics Command 1999); for this application, the CA lattice has been scaled to 1 lattice cell length = 125 m. Note that neighborhood scaling is adjustable within the CA model to reflect variations in mission, enemy, troops, terrain, and time available (METT-T) conditions.

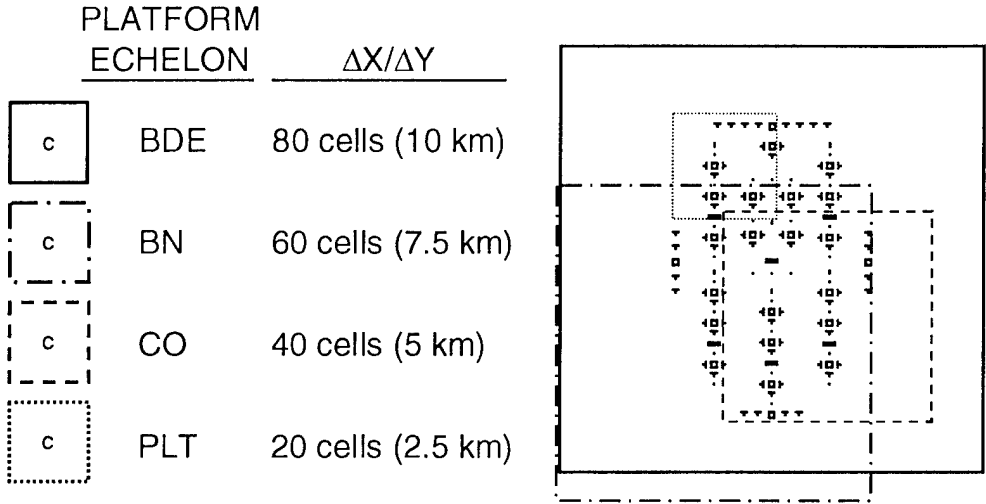


Figure 4. Area of operations neighborhood scaling within the CA brigade.

2.4.2 Situational Awareness Connectivity

Once the spatial scope of situational awareness has been delineated and implemented for each node cell within the brigade (thus creating an AO-based network topology), a quantitative measure of the degree to which a node is situationally aware of other friendly neighbor nodes can now be defined. Within the context of the CA model, the situational awareness of a TI node cell is a collective measure of the state of the communication channels which exist between the reference node and all neighboring friendly nodes within the reference node's AO neighborhood. This is illustrated in Figure 5, which depicts a sublattice structure of a center receiving node (represented by an "R") and the transmitting neighbor nodes (represented by a "T") that lie within the receiving node's AO neighborhood (the "R/T" notation represents server nodes which

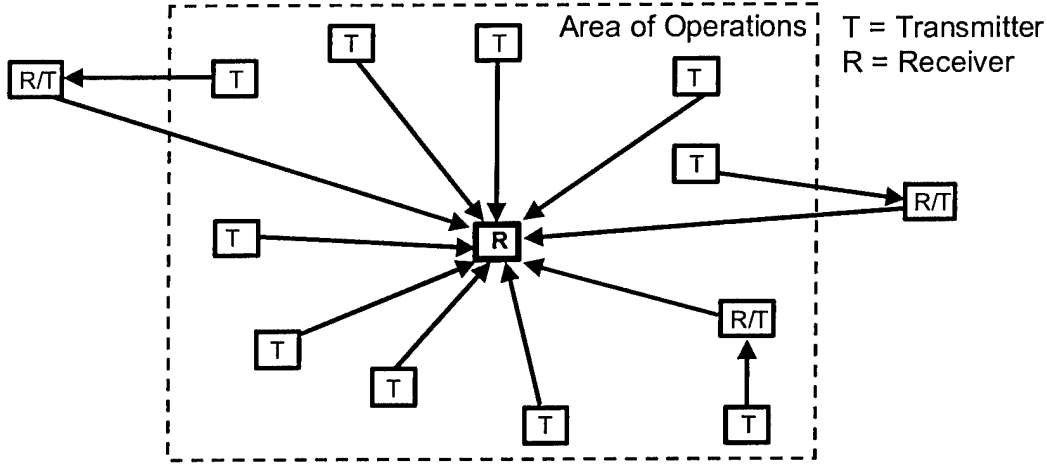


Figure 5. Local network topology of a center receiving node's AO neighborhood.

must be functionally capable of both receiving *and* transmitting). If the binary states $s_i^T(t_n)$ and $s_i^R(t_n)$ are defined as the i^{th} node's ability to transmit (1)/not transmit (0) and receive (1)/not receive (0) SA messages at time t_n (i.e. the n^{th} discrete simulation time step), respectively, then the state of the local network topology relative to the i^{th} node can be represented by a metric called the SA connectivity, as in

$$C_i(t_n) = \frac{s_i^R(t_n)}{N_i} \sum_j J_{ij}(t_n) s_j^T(t_n), \quad (1)$$

where N_i = number of friendly neighbor nodes within the i^{th} node's AO neighborhood and $J_{ij}(t_n)$ = binary communication channel continuity from the j^{th} to i^{th} node. This channel continuity metric is equal to 1 only if (a) the j^{th} neighbor node lies within the i^{th} node's AO neighborhood, and (b) the channel connecting nodes i and j are intact at time step t_n . Thus, $C_i(t_n)$ indicates the fraction of neighbor nodes within the i^{th} node's AO neighborhood that can transmit SA messages at t_n . The representation of SA connectivity expressed in equation (1) is analogous to the local energy E_i within a two-spin, variable-range Ising model from statistical physics, which has also been applied to the modeling and simulation of diverse cooperative/interdependent phenomena such as protein folding (Bar-Yam 1997) and political decision-related structures (Galam 1997).

It should be noted that if communication channel continuity depends on the availability of relay server nodes, then the term $J_{ij}(t_n)$ in equation (1) is also a function of $s^T(t_n)$ and $s^R(t_n)$ for each server making up the channel. This is addressed in Table 1, which defines $J_{ij}(t_n)$ (and the converse continuity from the

Table 1. Definitions of $J_{ij}(t_n)$ and $J_{ji}(t_n)$ for all possible SA communication channels.

Description of Node i	Description of Node j	$J_{ij}(t_n)$	$J_{ji}(t_n)$
Any client in local net X_1 and CSMA net Y_1	Any other client in local net X_1 and CSMA net Y_1	$\delta_{i,j}^{AO}$	$\delta_{j,i}^{AO}$
SINCGARS-only client in local net X_1 and CSMA net Y_1	SINCGARS-only client in local net X_2 and CSMA net Y_1	$s_{CSMA, X_1}(t_n) \wedge s_{CSMA, X_2}(t_n) \wedge \delta_{i,j}^{AO}$	$s_{CSMA, X_1}(t_n) \wedge s_{CSMA, X_2}(t_n) \wedge \delta_{j,i}^{AO}$
SINCGARS-only client in local net X_1 and CSMA net Y_1	SINCGARS-only client in local net X_3 and CSMA net Y_2	$s_{CSMA, X_1}(t_n) \wedge s_{CSMA, X_3}(t_n) \wedge s_{MSG, Y_2}(t_n) \wedge \delta_{i,j}^{AO}$	$s_{CSMA, X_1}(t_n) \wedge s_{CSMA, X_3}(t_n) \wedge s_{MSG, Y_1}(t_n) \wedge \delta_{j,i}^{AO}$
SINCGARS-only client in local net X_1 and CSMA net Y_1	EPLRS-equipped client in local net X_2 and CSMA net Y_1	$s_{CSMA, X_1}(t_n) \wedge s_{CSMA, X_2}(t_n) \wedge \delta_{i,j}^{AO}$	$s_{CSMA, X_1}(t_n) \wedge \delta_{j,i}^{AO}$
SINCGARS-only client in local net X_1 and CSMA net Y_1	EPLRS-equipped client in local net X_3 and CSMA net Y_2	$s_{CSMA, X_1}(t_n) \wedge s_{CSMA, X_3}(t_n) \wedge s_{MSG, Y_2}(t_n) \wedge \delta_{i,j}^{AO}$	$s_{CSMA, X_1}(t_n) \wedge s_{MSG, Y_1}(t_n) \wedge \delta_{j,i}^{AO}$
SINCGARS-only client in local net X_1 and CSMA net Y_1	CSMA server in local net X_2 and CSMA net Y_1	$s_{CSMA, X_1}(t_n) \wedge \delta_{i,j}^{AO}$	$s_{CSMA, X_1}(t_n) \wedge \delta_{j,i}^{AO}$
SINCGARS-only client in local net X_1 and CSMA net Y_1	CSMA server in local net X_3 and CSMA net Y_2	$s_{CSMA, X_1}(t_n) \wedge s_{MSG, Y_2}(t_n) \wedge \delta_{i,j}^{AO}$	$s_{CSMA, X_1}(t_n) \wedge s_{MSG, Y_1}(t_n) \wedge \delta_{j,i}^{AO}$
SINCGARS-only client in local net X_1 and CSMA net Y_1	Self server in CSMA net Y_1	$s_{CSMA, X_1}(t_n) \wedge \delta_{i,j}^{AO}$	$s_{CSMA, X_1}(t_n) \wedge \delta_{j,i}^{AO}$
SINCGARS-only client in local net X_1 and CSMA net Y_1	Self server in CSMA net Y_2	$s_{CSMA, X_1}(t_n) \wedge s_{MSG, Y_2}(t_n) \wedge \delta_{i,j}^{AO}$	$s_{CSMA, X_1}(t_n) \wedge s_{MSG, Y_1}(t_n) \wedge \delta_{j,i}^{AO}$
SINCGARS-only client in local net X_1 and CSMA net Y_1	MSG server in local net X_4 and CSMA net Y_1	$s_{CSMA, X_1}(t_n) \wedge s_{CSMA, X_4}(t_n) \wedge \delta_{i,j}^{AO}$	$s_{CSMA, X_1}(t_n) \wedge \delta_{j,i}^{AO}$
SINCGARS-only client in local net X_1 and CSMA net Y_1	MSG server in local net X_5 and CSMA net Y_2	$s_{CSMA, X_1}(t_n) \wedge s_{CSMA, X_5}(t_n) \wedge s_{MSG, Y_2}(t_n) \wedge \delta_{i,j}^{AO}$	$s_{CSMA, X_1}(t_n) \wedge s_{MSG, Y_1}(t_n) \wedge \delta_{j,i}^{AO}$
EPLRS-equipped client in local net X_1 and CSMA net Y_1	EPLRS-equipped client in local net X_2 and CSMA net Y_1	$s_{CSMA, X_2}(t_n) \wedge \delta_{i,j}^{AO}$	$s_{CSMA, X_1}(t_n) \wedge \delta_{j,i}^{AO}$
EPLRS-equipped client in local net X_1 and CSMA net Y_1	EPLRS-equipped client in local net X_3 and CSMA net Y_2	$s_{CSMA, X_3}(t_n) \wedge s_{MSG, Y_2}(t_n) \wedge \delta_{i,j}^{AO}$	$s_{CSMA, X_1}(t_n) \wedge s_{MSG, Y_1}(t_n) \wedge \delta_{j,i}^{AO}$

Table 1. Definitions of $J_{ij}(t_n)$ and $J_{ji}(t_n)$ for all possible SA communication channels (continued).

Description of Node i	Description of Node j	$J_{ij}(t_n)$	$J_{ji}(t_n)$
EPLRS-equipped client in local net X_1 and CSMA net Y_1	CSMA server in local net X_2 and CSMA net Y_1	$\delta_{i,j}^{AO}$	$s_{CSMA, X_1}(t_n) \wedge \delta_{j,i}^{AO}$
EPLRS-equipped client in local net X_1 and CSMA net Y_1	CSMA server in local net X_3 and CSMA net Y_2	$s_{MSG, Y_2}(t_n) \wedge \delta_{i,j}^{AO}$	$s_{CSMA, X_1}(t_n) \wedge s_{CSMA, Y_1}(t_n) \wedge \delta_{j,i}^{AO}$
EPLRS-equipped client in local net X_1 and CSMA net Y_1	Self server in CSMA net Y_1	$\delta_{i,j}^{AO}$	$s_{CSMA, X_1}(t_n) \wedge \delta_{j,i}^{AO}$
EPLRS-equipped client in local net X_1 and CSMA net Y_1	Self server in CSMA net Y_2	$s_{MSG, Y_2}(t_n) \wedge \delta_{i,j}^{AO}$	$s_{CSMA, X_1}(t_n) \wedge s_{MSG, Y_1}(t_n) \wedge \delta_{j,i}^{AO}$
EPLRS-equipped client in local net X_1 and CSMA net Y_1	MSG server in local net X_4 and CSMA net Y_1	$s_{CSMA, X_4}(t_n) \wedge \delta_{i,j}^{AO}$	$s_{CSMA, X_1}(t_n) \wedge \delta_{j,i}^{AO}$
EPLRS-equipped client in local net X_1 and CSMA net Y_1	MSG server in local net X_5 and CSMA net Y_2	$s_{CSMA, X_5}(t_n) \wedge s_{MSG, Y_2}(t_n) \wedge \delta_{i,j}^{AO}$	$s_{CSMA, X_1}(t_n) \wedge s_{MSG, Y_1}(t_n) \wedge \delta_{j,i}^{AO}$
CSMA server in local net X_1 and CSMA net Y_1	CSMA server in local net X_2 and CSMA net Y_1	$\delta_{i,j}^{AO}$	$\delta_{j,i}^{AO}$
CSMA server in local net X_1 and CSMA net Y_1	CSMA server in local net X_3 and CSMA net Y_2	$s_{MSG, Y_2}(t_n) \wedge \delta_{i,j}^{AO}$	$s_{MSG, Y_1}(t_n) \wedge \delta_{j,i}^{AO}$
CSMA server in local net X_1 and CSMA net Y_1	Self server in CSMA net Y_1	$\delta_{i,j}^{AO}$	$\delta_{j,i}^{AO}$
CSMA server in local net X_1 and CSMA net Y_1	Self server in CSMA net Y_2	$s_{MSG, Y_2}(t_n) \wedge \delta_{i,j}^{AO}$	$s_{MSG, Y_1}(t_n) \wedge \delta_{j,i}^{AO}$
CSMA server in local net X_1 and CSMA net Y_1	MSG server in local net X_4 and CSMA net Y_1	$s_{CSMA, X_4}(t_n) \wedge \delta_{i,j}^{AO}$	$\delta_{j,i}^{AO}$
CSMA server in local net X_1 and CSMA net Y_1	MSG server in local net X_5 and CSMA net Y_2	$s_{CSMA, X_5}(t_n) \wedge s_{MSG, Y_2}(t_n) \wedge \delta_{i,j}^{AO}$	$s_{MSG, Y_1}(t_n) \wedge \delta_{j,i}^{AO}$
Self server in CSMA net Y_1	Self server in CSMA net Y_1	$\delta_{i,j}^{AO}$	$\delta_{j,i}^{AO}$

Table 1. Definitions of $J_{ij}(t_n)$ and $J_{ji}(t_n)$ for all possible SA communication channel (continued).

Description of Node i	Description of Node j	$J_{ij}(t_n)$	$J_{ji}(t_n)$
Self server in CSMA net Y_1	Self server in CSMA net Y_2	$s_{MSG,Y_2}(t_n) \wedge \delta_{i,j}^{AO}$	$s_{MSG,Y_1}(t_n) \wedge \delta_{j,i}^{AO}$
Self server in CSMA net Y_1	MSG server in local net X_4 and CSMA net Y_1	$s_{CSMA,X_4}(t_n) \wedge \delta_{i,j}^{AO}$	$\delta_{j,i}^{AO}$
Self server in CSMA net Y_1	MSG server in local net X_5 and CSMA net Y_2	$s_{CSMA,X_5}(t_n) \wedge s_{MSG,Y_2}(t_n) \wedge \delta_{i,j}^{AO} \delta$	$s_{MSG,Y_1}(t_n) \wedge \delta_{j,i}^{AO}$
MSG server in local net X_4 and CSMA net Y_1	MSG server in local net X_5 and CSMA net Y_2	$s_{CSMA,X_5}(t_n) \wedge s_{MSG,Y_2}(t_n) \wedge \delta_{i,j}^{AO} \delta$	$s_{CSMA,X_4}(t_n) \wedge s_{MSG,Y_1}(t_n) \wedge \delta_{j,i}^{AO}$

Notes:

$s_{CSMA,X_n}(t_n)$: communication capability state of the CSMA server in local SINCGARS net X_n .

$s_{MSG,Y_m}(t_n)$: communication capability state of the MSG server in CSMA net Y_m .

$\delta_{i,j}^{AO}$: delta function representing the inclusion of the j^{th} node within the i^{th} node's AO neighborhood.

$\wedge$: binary AND operator.

i^{th} to j^{th} node $J_{ji}(t_n)$ for all possible SA communication channels. As indicated in this table, an SA communication channel is built using the following generic elements.

- *CSMA servers.* The binary communication capability state of the CSMA server in local SINCGARS net X_n at time step t_n is $s_{CSMA,X_n}(t_n) = s_{CSMA,X_n}^T(t_n) \wedge s_{CSMA,X_n}^R(t_n)$, where $s_{CSMA,X_n}^T(t_n)$ and $s_{CSMA,X_n}^R(t_n)$ are the transmission and reception capability states of the server, respectively.
- *MSG servers.* The communication capability state of the MSG server in CSMA net Y_m at time step t_n is $s_{MSG,Y_m}(t_n) = s_{MSG,Y_m}^T(t_n) \wedge s_{MSG,Y_m}^R(t_n)$, where $s_{MSG,Y_m}^T(t_n)$ and $s_{MSG,Y_m}^R(t_n)$ are the transmission and reception states of the server, respectively.
- *AO neighborhoods.* The binary metric $\delta_{i,j}^{AO}$ is a delta function representing the inclusion of the j^{th} node within the i^{th} node's AO neighborhood, and is defined as

$$\delta_{i,j}^{AO} = \begin{cases} 1 & \text{if the } j^{\text{th}} \text{ cell lies within the } i^{\text{th}} \text{ cell's AO neighborhood} \\ 0 & \text{otherwise} \end{cases}$$

In these expressions, the symbol $\wedge$ is the binary AND operator.

It is interesting to consider the various scales over which the SA connectivity is calculated for the nodes within the CA brigade (Figure 3). Given the total number of SA transmitting neighbor nodes N_i relative to the i^{th} node cell (as defined in equation [1]), one can define the connectivity distribution $P(N)$, which gives the probability that a randomly selected reference node has N neighboring nodes within its AO neighborhood (where $N = 0, 1, 2, \dots, 611$). Figure 6 displays the $P(N)$ distribution for the CA brigade; this conveys the range of scales over which nodes receive SA messages. Note that the higher the number of nodes within a platform's AO neighborhood, the more likely it is to be a higher-echelon platform.

At this point, the neighbor nodes within a center reference cell's AO neighborhood do *not* directly interact with the reference cell as part of its cell transition function. At it is used here, the term "transition" explicitly refers to a change in the reference cell's functional transmission and reception capabilities. In other words, the functional states of node cells evolve independently of one another; what the CA model still lacks at this point in our discussion is a type of neighbor cell that directly interacts with a reference node cell to perturb its communication capabilities through IO stress. This type of threat cell is introduced in the next section.

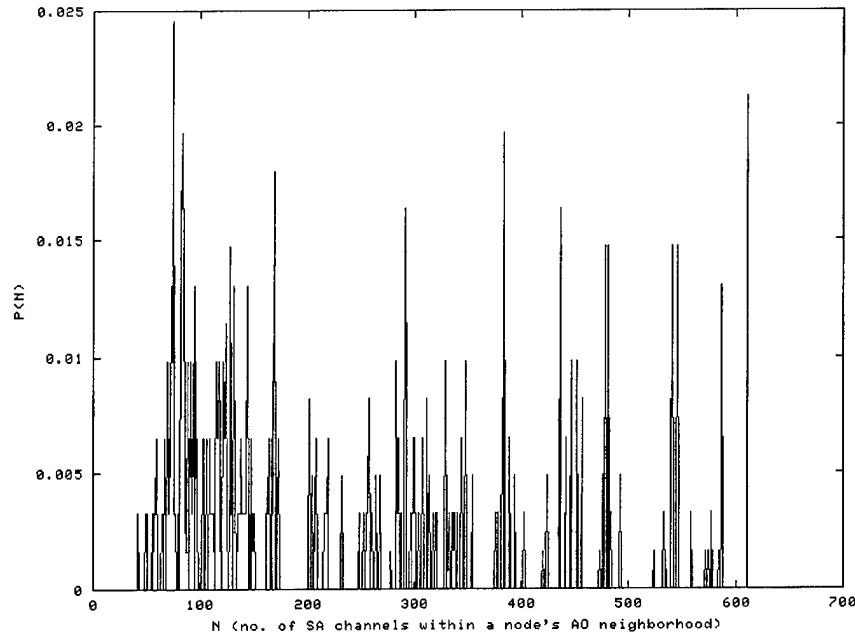


Figure 6. Connectivity distribution $P(N)$ for the CA brigade.

2.5 Introducing Information Operations Stress

The dynamics of the CA model are now added through the introduction of an IO stress, which directly interacts with a node cell in order to perturb its communication capabilities. Interaction of IO stress with nodes can result in either (a) temporary dysfunction of a node (where $s_i^T(t_n) = 0$, $s_i^R(t_n) = 0$, or $s_i^T(t_n) = s_i^R(t_n) = 0$), (b) permanent dysfunction of the node, or (c) “spoofing” of the global positioning system (GPS) receiver operating onboard the node platform. This last type of effect forces the impacted node to launch an unscheduled SA update message (which is then routed throughout the brigade via the network architecture depicted in Figure 1); in this case, both $s_i^T(t_n)$ and $s_i^R(t_n)$ remain equal to 1. IO stress can arise either from a stochastic parameter embedded within a nodal state vector or an external neighboring cell (representing a localized IO threat source) that generates a local “stress field” (these two different types of stress will be explicitly modeled in section 3).

The dynamics which emerge from node cell/IO stress interactions are modeled as perturbations to a node’s communication capability states. Given that equation (1) defines the i^{th} node’s SA connectivity within a stress-free environment, then the stress-perturbed SA connectivity of that node at time step t_n is

$$C_i(t_n) = \frac{(s_i^R(t_{n-1}) + \Delta s_i^R(t_n))}{N_i} \sum_j (J_{ij}(t_{n-1}) + \Delta J_{ij}(t_n)) (s_j^T(t_{n-1}) + \Delta s_j^T(t_n)). \quad (2)$$

In this equation, $s_i^R(t_{n-1})$, $s_j^T(t_{n-1})$, and $J_{ij}(t_{n-1})$ are the i^{th} node’s prior reception and j^{th} node’s prior transmission capability states, and the j^{th} to i^{th} node channel continuity at time step t_{n-1} , respectively, and $\Delta s_i^R(t_n)$, $\Delta s_j^T(t_n)$, and $\Delta J_{ij}(t_n)$ are perturbations (induced at the current time step t_n) to the i^{th} node’s prior reception and j^{th} node’s prior transmission capability states, and the j^{th} to i^{th} node prior channel continuity, respectively. The last metric reflects the introduction (and possible subsequent removal) of stress-related perturbations to those neighboring server nodes connecting the i^{th} and j^{th} nodes (as defined in Table 1). Each of the metrics $\Delta s_k^T(t_n)$, $\Delta s_k^R(t_n)$, and $\Delta J_{ij}(t_n)$ may assume values of either -1 or 1, reflecting the loss or return of communication capabilities, respectively. Thus, equation (2) models a process of *dynamic response*, where stress-induced perturbations can affect nodes either directly ($s_i^R[t_{n-1}] + \Delta s_i^R[t_n] = 0$) or indirectly ($[J_{ij}(t_{n-1}) + \Delta J_{ij}(t_n)] = 0$ or $[s_j^T(t_{n-1}) + \Delta s_j^T(t_n)] = 0$).

2.6 Analyzing the Emergent Network Dynamics

There are several data processing filters that have been coded into the CA model for analyzing time series data. In this section, these filters are described in detail.

2.6.1 Normalized Network Stress $\langle h^{stress}(t_n) \rangle$

The first data filter calculates $\langle h^{stress}(t_n) \rangle$, the normalized perturbative stress that is introduced into the entire brigade system via localized IO stress. This can be expressed as

$$\langle h^{stress}(t_n) \rangle = \frac{1}{M} \sum_{i=1}^M h_i^{stress}(t_n), \quad (3)$$

where M is the total number of node platforms within the digitized brigade and $h_i^{stress}(t_n)$ indicates a stress-induced perturbation of the i^{th} node's communication capability states at time step t_n . Thus, $\langle h^{stress}(t_n) \rangle$ indicates the fraction of stress-impacted nodes as a function of time. For the "movement to contact" brigade structure presented in section 2.3.2, $M = 612$. It should be noted that while $\langle h^{stress}(t_n) \rangle$ reflects interactions between IO stress sources and target node cells, $C_i(t_n)$, on the other hand, reflects inter-cell dependencies which directly contribute to a node's SA connectivity as defined in equation 1. Thus, node cell/IO stress source *interactions* are distinguished apart from reference node/ neighbor node topological *interdependencies*.

2.6.2 Globally Averaged Situational Awareness Connectivity $\langle C(t_n) \rangle$

The second data filter calculates globally averaged SA connectivity as a function of scenario time, where an average value is calculated at each point in discrete simulation time across all node platforms within the CA brigade. The dynamic globally averaged SA connectivity for the CA brigade, $\langle C(t_n) \rangle$, is defined

$$\langle C(t_n) \rangle = \frac{1}{M} * \sum_{i=1}^M C_i(t_n), \quad (4)$$

where again $M = 612$ is the total number of nodes within the brigade.

2.6.3 Situational Awareness Fitness

Given that the total SA connectivity state of the M -node CA brigade can be represented as an M -tuple point within a unit hypercube (which can be thought of as representing the global system "phase space"), the hypercube axes (where the i^{th} axis represents the SA connectivity level $C_i(t_n)$ of the i^{th} node) can be "coarse-grained" (Wolfram 1985) or partitioned into interval-based *fitness states*. Figure 7 defines how a five-state partition (used by Guzie for vulnerability risk assessment [Guzie 2000]) is applied for evaluating the SA connectivity $C_i(t_n)$ of the i^{th} node; this partition divides the unit interval into the interval-based fitness states of *very high* (blue), *high* (green), *medium* (yellow), *low* (red), and *very low* (orange). The assigned colors are used to visually convey a node's fitness state during a simulation run (which will be discussed in the subsequent section of this report). This has the effect of dividing the unit hypercube continuum into 5^M discrete blocks, where each block represents a unique global system state.

VERY HIGH		$C_i(t_n) > 0.9$
HIGH		$0.6 < C_i(t_n) \leq 0.9$
MEDIUM		$0.4 < C_i(t_n) \leq 0.6$
LOW		$0.1 < C_i(t_n) \leq 0.4$
VERY LOW		$C_i(t_n) \leq 0.1$

Figure 7. Application of Guzie's five-state partition to the SA connectivity of the i^{th} node.

For the 612-node brigade structure, there are 5^{612} global system states. Figure 8 illustrates how coarse-graining the SA connectivity levels for each of the 612 brigade nodes modifies the global system's dynamic state vector (where C_1 through C_{612} indicate the SA connectivity for nodes 1–612, respectively) as the system is perturbed via interactions with IO stress sources. Here, continuous levels of nodal SA connectivity are mapped into one of five fitness states at each subsequent time step of a simulation run. Finally, $P_{\text{fitness}}(t_n)$ is defined as the fraction of nodes in a particular fitness state at time step t_n .

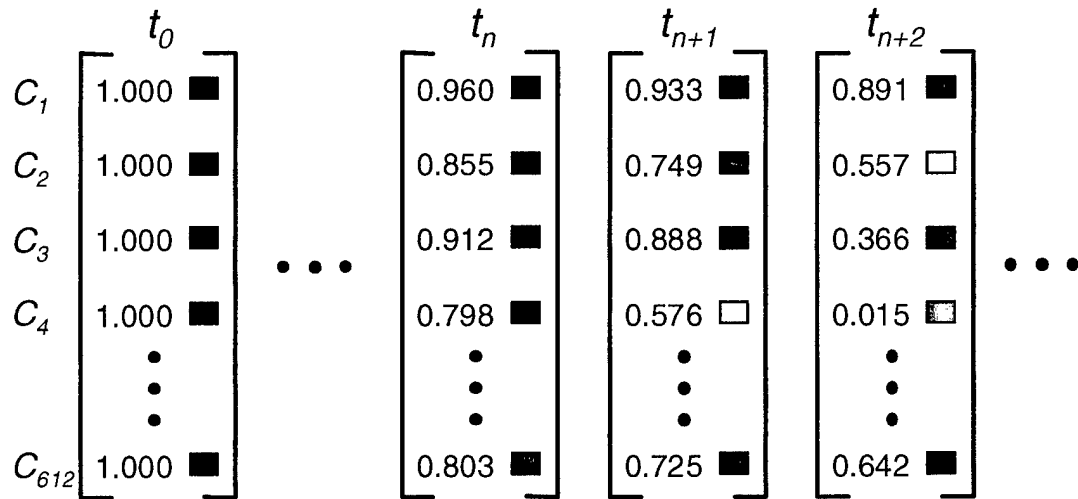


Figure 8. Dynamics of the SA connectivity state vector for the CA brigade.

The third data filter coded into the CA model calculates the five fitness metrics— $P_{very\ high}(t_n)$, $P_{high}(t_n)$, $P_{medium}(t_n)$, $P_{low}(t_n)$, and $P_{very\ low}(t_n)$. These metrics provide a different perspective relative to $\langle C(t_n) \rangle$. They illustrate the distribution of various situational awareness levels across individual platform nodes, rather than refer to a global average. Thus, when the five metrics $P_{very\ high}(t_n)$, $P_{high}(t_n)$, $P_{medium}(t_n)$, $P_{low}(t_n)$, and $P_{very\ low}(t_n)$ (where $P_{very\ high}(t_n) + P_{high}(t_n) + P_{medium}(t_n) + P_{low}(t_n) + P_{very\ low}(t_n) = 1$ for all t_n) are taken in combination, they offer a concise way to represent the 612-tuple system state (one of the 5^{612} discrete blocks in the “phase space” hypercube) at t_n . It should be noted, however, that there can be many different global system states represented by the same 5-tuple combination $[P_{very\ high}(t_n), P_{high}(t_n), P_{medium}(t_n), P_{low}(t_n), P_{very\ low}(t_n)]$.

2.6.4 Course-Grained Entropy

The fitness states introduced in the previous section partition the M -tuple global SA connectivity system state at time step t_n into five mutually exclusive sets of states, where the probability of occupying the k^{th} fitness state is $P_k(t_n)$ and

$$\sum_{k=1}^5 P_k(t_n) = 1. \quad (5)$$

These conditions define a *complete finite probability scheme* based on the coarse graining of nodal SA connectivity measures into discrete states. Given such a scheme, the global *state entropy* $H(t_n)$ is defined as a quantitative measure of the disorder/volatility within the global system at time step t_n , where

$$H(t_n) = - \sum_{k=1}^5 P_k(t_n) * \log_5(P_k(t_n)). \quad (6)$$

The state entropy, which is based on the Shannon entropy of an information source, can range from a level of 0 (indicating that all nodes are in the same fitness state) up to a level of 1 (all nodes are equally distributed among all possible fitness states).

The concept of state entropy can be extended in order to quantify the temporal disorder between system states measured at successive time steps t_{n-1} and t_n . Thus, the *temporal entropy* $H(t_n, t_{n-1})$, based on the conditional Shannon entropy as measured between an information source and receiver, is defined as

$$H(t_n, t_{n-1}) = - \sum_{j=1}^5 \sum_{k=1}^5 P_j(t_{n-1}) * P(k, t_n | j, t_{n-1}) * \log_5(P(k, t_n | j, t_{n-1})). \quad (7)$$

In this equation, $P(k, t_n | j, t_{n-1})$ is the conditional probability that a node in the j^{th} fitness state at time step t_{n-1} maps into the k^{th} fitness state at time step t_n . As with state entropy, the temporal entropy can range from a level of 0 (indicating a static condition where all nodes remain in the same fitness state from t_{n-1} to t_n) up to a

level of 1 (indicating a maximally volatile condition where nodes are equally likely to map into any one of the possible fitness states from t_{n-1} to t_n).

2.6.5 Return Map

Once the $P_{fitness}(t_n)$ time series have been calculated for an IO scenario, the data within each time series can be analyzed for certain structural characteristics. This is done through the use of a *return map*, which is a 2-D scatter plot of $P_{fitness}(t_{n+1})$ vs. $P_{fitness}(t_n)$ for each fitness state.* A return map illustrates the dynamics of a 1-D time series (i.e., a time series with just one independent variable) in a recursive rather than successive format. The utility of a return map for portraying time series structure is illustrated in Figure 9, which portrays the dynamics of the chaotic 1-D logistic map $x_{n+1} = 4x_n(1 - x_n)$ in both standard time series (Figure 9[a]) and return map (Figure 9[b]) formats. In this figure, the first plot appears to depict a random time series with values in the interval $[0, 1]$, while the second plot (displaying x_{n+1} vs. x_n) clearly indicates the presence of an *attractor* (a subset of one or more phase-space points of the form $[x_n, x_{n+1}]$ towards which a deterministic dynamical process will evolve or is "attracted").

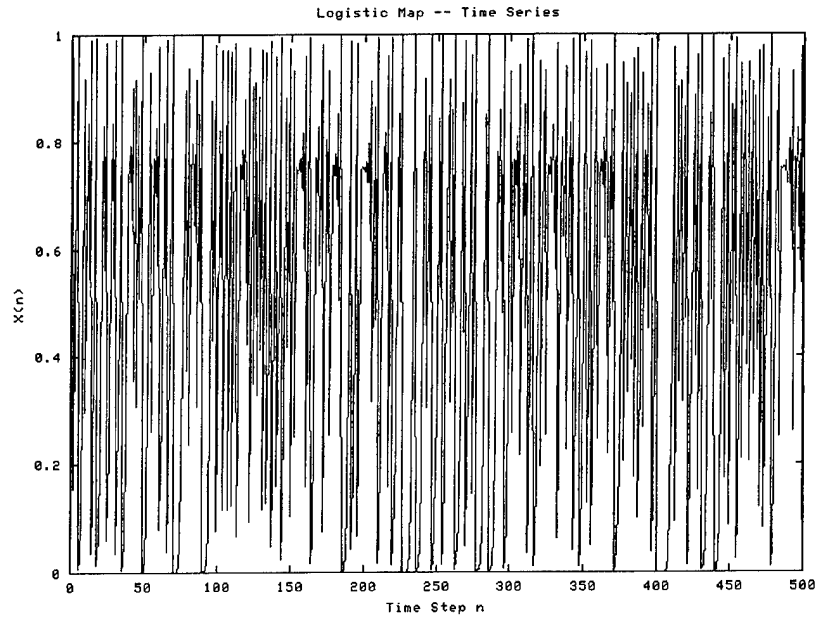
3. Information Operations Simulations

In this section, two different models of IO stress are introduced, and simulation results of CA model runs incorporating these stresses are presented and discussed. Specifically, these stress models address (a) a "mean field" stress embedded within a node cell and (b) localized stress fields generated by source cells external to node cells.

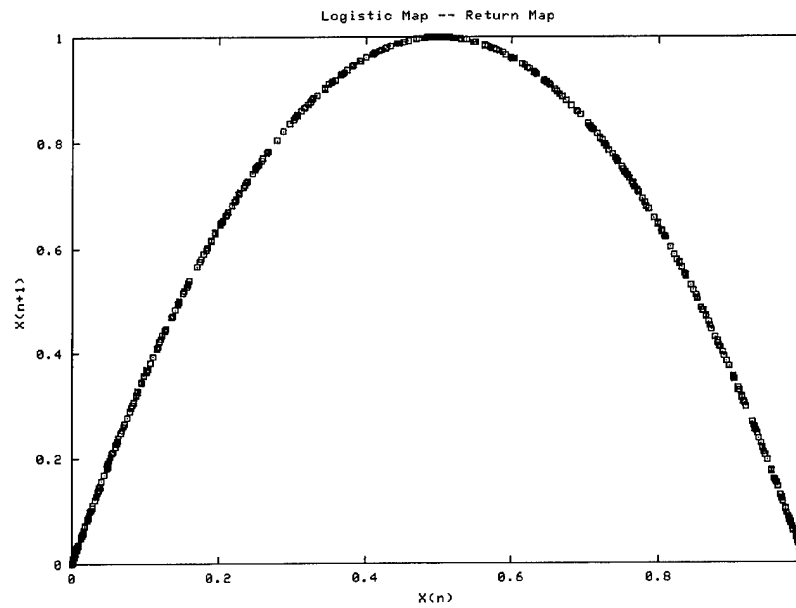
3.1 Mean Field Stress

The first type of IO stress was modeled as a stochastic process which acts to perturb the functional communication states of each node within the SA network. This IO stress arises from a random binary variable representing nodal digital communication capability (i.e., a node either can [1] or cannot [0] communicate with other nodes, where "communicate" implies both transmission and reception subcapabilities). Such a random variable is embedded within each nodal state vector, which are synchronously updated for each node in the network. Since these distributed random variables evolve independently from one another, there is no spatial correlation between the stress at different nodes.

* An equivalent version of the return map would plot $P_{fitness}(t_n)$ vs. $P_{fitness}(t_{n-1})$.



(a)



(b)

Figure 9. Dynamics of the chaotic 1-D logistic map $x_{n+1} = 4x_n(1-x_n)$ presented in (a) standard time series and (b) return map formats.

This type of IO stress is similar to the *mean field approximation* from statistical physics, where every individual element within a multi-element system is subjected to the average influence of a global field interacting with the rest of the system (Bar-Yam 1997). In this sense, the “mean field” IO stress is at best a first-order approximation to a real multi-threat IO environment, serving to demonstrate the emergent behavior of the brigade’s SA network topology under a coherent hypothetical rather than realistic stress.

The dynamics of this mean field stress are described by a stationary Markov chain which is applied uniformly and independently to each of the nodes within the digitized brigade. This Markov chain can be written as the iterative map

$$P_{s_i}(t_n) = P_{s_i}(t_{n-1})\lambda(t_n, t_{n-1}), \quad (8)$$

where

$$P_{s_i}(t_n) = [P(s_i^1(t_n)) \ P(s_i^2(t_n)) \ P(s_i^3(t_n)) \ \dots \ P(s_i^L(t_n))] \quad (9)$$

and

$$P_{s_i}(t_{n-1}) = [P(s_i^1(t_{n-1})) \ P(s_i^2(t_{n-1})) \ P(s_i^3(t_{n-1})) \ \dots \ P(s_i^L(t_{n-1}))] \quad (10)$$

are row vectors denoting state probabilities for the i^{th} network node at time steps t_n and t_{n-1} , respectively, and $\lambda(t_n, t_{n-1})$ is a stationary one-step transition matrix. The notation $P(s_i^k(t_n))$ indicates the probability that the i^{th} node occupies the k^{th} of L possible states at time step t_n , and $s_i^k(t_n) = [s_i^{k,T}(t_n), s_i^{k,R}(t_n)]$ is the i^{th} node’s communication state vector consisting of transmission and reception capability states, respectively.

The stationary transition matrix $\lambda(t_n, t_{n-1})$ introduced in equation (8) can reflect either transient/permanent nodal communication dysfunction or spoofing of the GPS receiver mounted within a node, causing the node to launch an unscheduled SA update message across the network. The former form of the transition matrix is expressed as

$$\lambda^{dysfunc}(t_n, t_{n-1}) = \begin{bmatrix} P(s^F(t_n)|s^F(t_{n-1})) & P(s^{TD}(t_n)|s^F(t_{n-1})) & P(s^{PD}(t_n)|s^F(t_{n-1})) \\ P(s^F(t_n)|s^{TD}(t_{n-1})) & P(s^{TD}(t_n)|s^{TD}(t_{n-1})) & P(s^{PD}(t_n)|s^{TD}(t_{n-1})) \\ P(s^F(t_n)|s^{PD}(t_{n-1})) & P(s^{TD}(t_n)|s^{PD}(t_{n-1})) & P(s^{PD}(t_n)|s^{PD}(t_{n-1})) \end{bmatrix}, \quad (11)$$

while the latter form is expressed as

$$\lambda^{spoof}(t_n, t_{n-1}) = \begin{bmatrix} P(s^F(t_n)|s^F(t_{n-1})) & P(s^S(t_n)|s^F(t_{n-1})) \\ P(s^F(t_n)|s^S(t_{n-1})) & P(s^S(t_n)|s^S(t_{n-1})) \end{bmatrix}, \quad (12)$$

where $s^F(t_n)$, $s^{TD}(t_n)$, $s^{PD}(t_n)$, and $s^S(t_n)$ represent the binary nodal communication states functional, transient dysfunction, permanent dysfunction, and functional but spoofed, respectively. Thus, the stochastic iterative map in equation (8) can describe the transitional probability for each network node between functional ($s_i^T(t_n) = s_i^R(t_n) = 1$) and transient or permanent dysfunction (where $s_i^T(t_n) = s_i^R(t_n) = 0$ in both cases), or between functional and functional but spoofed states.

Since the mean field stress model considers the impact of network flooding on nodal connectivity, the dynamics of SA message dissemination must be incorporated into the model. If it is assumed that (1) a node platform must report its new position every time it moves a distance of 100 m and (2) all nodes within the brigade are moving forward at a constant speed of 3 km/hr, then each node automatically sends out a new SA message once every 120 s. The dynamics of SA message dissemination throughout the CA brigade is based on the process used in the Next Generation Performance Model (NGPM) developed by the U.S. Army Communication Electronics Command (CECOM) (U.S. Army Communication Electronics Command 1999); see Appendix A for details on this process. Finally, in order to prevent flooding due to automatic SA message reporting, initial reporting times between 0 and 120 s are assigned to each node from a uniform random distribution.

When applying a mean field stress as described in equation (8), the SA connectivity described in equation (1) must be modified to the form

$$C_i(t_n) = \frac{s_i(t_n)}{N_i} \sum_j J'_{ij}(t_n) s_j(t_n), \quad (13)$$

where $s_i(t_n)$ and $s_j(t_n)$ are the binary communication capability states (i.e., communicate/can't communicate) of the i^{th} and j^{th} nodes at time step t_n , respectively. In equation (13), the communication channel continuity between the i^{th} and j^{th} nodes is now expressed as

$$J'_{ij}(t_n) = J_{ij}(t_n) \delta_{ij}^{\text{continuity}}(t_n), \quad (14)$$

where the binary channel continuity metric $\delta_{ij}^{\text{continuity}}(t_n) = 1$ only if the channel transmission latency L_{ij} (the total amount of time required for an SA message to travel between the i^{th} and j^{th} nodes) is less than a preset threshold limit $L_{\text{threshold}}$. The situation where $\delta_{ij}^{\text{continuity}}(t_n) = 0$ (i.e., $L_{ij} \geq L_{\text{threshold}}$) arises due to excessive message queue loading of the servers making up the communication channel.

This process is analogous to balls being thrown into a funnel with a trap door (Figure 10). The communication channel connecting the j^{th} to the i^{th} node is represented as a wide-mouthed funnel, which can collect and disseminate SA messages from multiple transmitting nodes (including the j^{th} node) in order to route the messages to the i^{th} node. Each incidence of an incoming ball (i.e., an SA message) contributes an additional member to a stack of balls (i.e., queued SA messages) waiting to be released through a trap door at the bottom of the funnel. This trap door opens (at a fixed frequency) just long enough to allow one ball to exit before closing again, representing a constant outgoing message baud rate for the communication channel. As the funnel fills up, the top of the stack of balls reaches and then surpasses a threshold level (i.e., $L_{\text{threshold}}$). At this point, the channel continuity metric $\delta_{ij}^{\text{continuity}}(t_n) = 0$, meaning that any new SA message coming in from the j^{th} node (i.e., the red ball at the top of the figure) will not arrive at the i^{th} node in time to be useful. Finally, $\delta_{ij}^{\text{continuity}}(t_n) = 1$ again only when the stack level in the funnel falls below the threshold latency level.

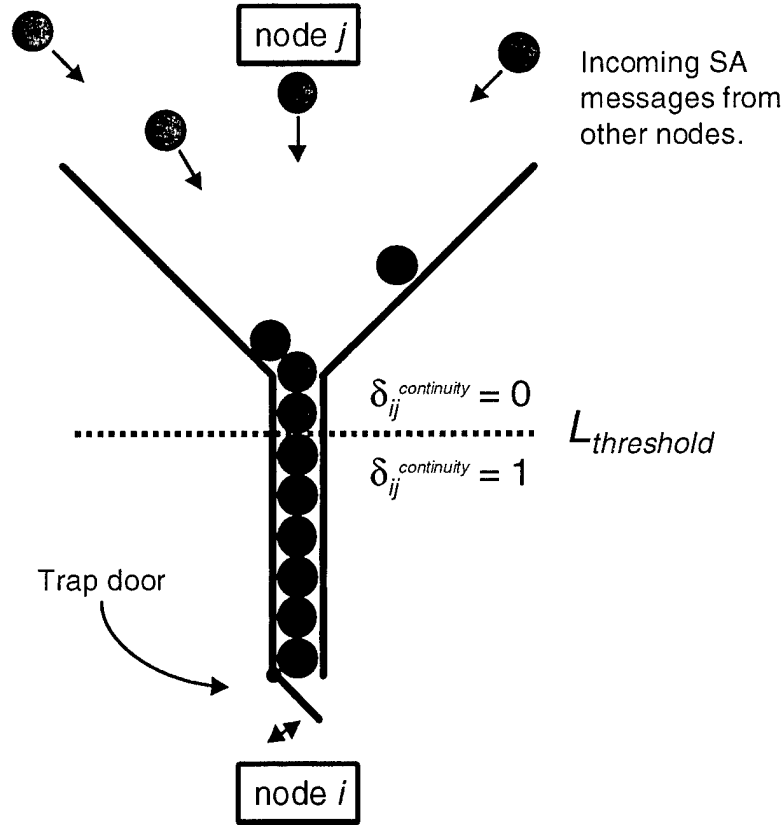


Figure 10. Representation of an inter-nodal communication channel as a funnel collecting and dispersing balls through a trap door.

Once the form of the mean field stress and associated SA connectivity have been established, the CA model simulating this type of IO stress can be run. Figure 11 depicts globally averaged SA connectivity (averaged over 10 simulation runs using different random number seeds) emerging across the brigade network under exposure to the form of mean field stress that induces transient/permanent nodal communication dysfunction. For these simulations, the transitional probabilities in equation (11) are set to

$$\lambda_{low}^{dysfunc}(t_n, t_{n-1}) = \begin{bmatrix} 0.99899 & 0.00100 & 0.00001 \\ 0.29999 & 0.70000 & 0.00001 \\ 0 & 0 & 1.00000 \end{bmatrix} \quad (15)$$

and

$$\lambda_{high}^{dysfunc}(t_n, t_{n-1}) = \begin{bmatrix} 0.9899 & 0.0100 & 0.0001 \\ 0.2999 & 0.7000 & 0.0001 \\ 0 & 0 & 1.0000 \end{bmatrix}, \quad (16)$$

where $\lambda_{low}^{dysfunc}(t_n, t_{n-1})$ and $\lambda_{high}^{dysfunc}(t_n, t_{n-1})$ are transition matrices associated with low and high levels of nodal dysfunction stress, respectively. Also, the channel latency threshold $L_{threshold}$ is set to 300 s (5 min), so that a communication channel is considered broken if the difference between the transmitted and actual positions of the reporting j^{th} node is ≥ 500 m. The figure readily shows that both low and high levels of this first type of mean field stress results in a roughly linear decrease in average SA connectivity. This is due to the coherent nature of the mean field stress as it is applied throughout the brigade structure, and also to the lack of spatial correlation between the functional states of network nodes.

Next, simulations are run using the second type of mean field stress, which induces GPS spoofing and subsequent network flooding. Figure 12 depicts globally averaged SA connectivity (again averaged over 10 simulation runs using different random number seeds) emerging across the brigade network under exposure to GPS spoofing stress. For these simulations, the transitional probabilities in equation (12) are set to

$$\lambda_{low}^{spooft}(t_n, t_{n-1}) = \begin{bmatrix} 0.99 & 0.01 \\ 0.30 & 0.70 \end{bmatrix} \quad (17)$$

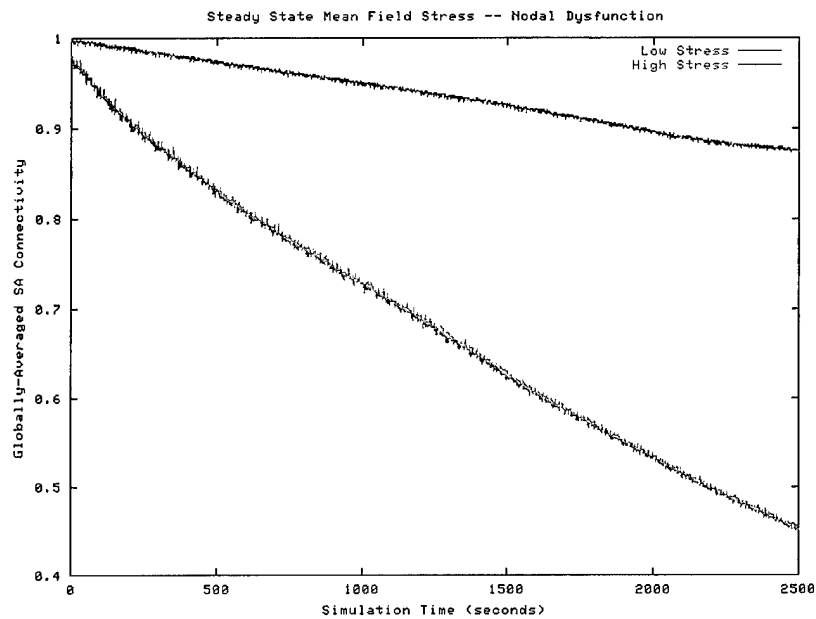


Figure 11. Average SA connectivity for the brigade network exposed to a mean field stress which induces nodal transient/permanent dysfunction.

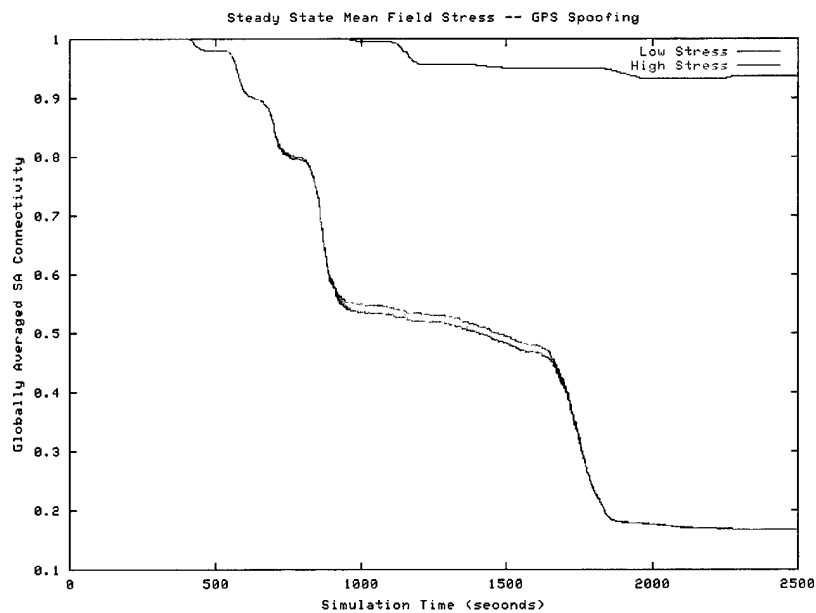


Figure 12. Average SA connectivity for the brigade network exposed to a mean field stress which induces nodal GPS spoofing.

and

$$\lambda_{high}^{spoof}(t_n, t_{n-1}) = \begin{bmatrix} 0.9 & 0.1 \\ 0.3 & 0.7 \end{bmatrix}, \quad (18)$$

where $\lambda_{low}^{spoof}(t_n, t_{n-1})$ and $\lambda_{high}^{spoof}(t_n, t_{n-1})$ are transition matrices associated with low and high levels of GPS spoofing stress, respectively. Again, the channel latency threshold $L_{threshold}$ is set to 300 s. Comparing this figure with Figure 11, it is clear that the message flooding which results from spoofing is the dominant cause of SA connectivity degradation when considering both types of mean field stress. Both the low and high levels of spoofing stress cause the average SA connectivity to decrease in staggered bursts of activity (a phenomenon known within complexity science as “punctuated equilibrium” [Bak and Sneppen 1993; Bak 1996; Newman and Sneppen 1996]) until a quasi-steady-state condition is reached at $t_n = 1850 - 1900$ s. At this point, server message queues are sufficiently loaded so that the transmission latency for all susceptible communication channels achieves or surpasses $L_{threshold}$, and will then likely remain in this state if the simulation timeframe is extended indefinitely.

Although the mean field model serves to comparatively demonstrate different forms of IO stress, it has shortcomings in two different areas. First, the model can significantly underpredict the impact of GPS spoofing on nodal SA connectivity. Since the brigade C2 network structure has deliberately been decoupled from the SA network structure as coded within the CA model, the cascading decrements in average SA connectivity depicted in Figure 12 can only reflect SA-message-induced server queue saturation. It is likely that the addition of a superimposed C2 structure will act to accelerate server queue saturation (where, for the majority of nodes, the channel continuity metric $\delta_{ij}^{continuity}(t_n)$ will jump to a value of zero much sooner) due to C2 message contention. Thus, the network response shown in Figure 12 can at best serve only as a lower bound when trying to predict the negative impact of GPS spoofing on nodal SA connectivity.

A more serious shortcoming of the mean field model is its failure to realistically consider correlations between spatially dispersed stress sources, as well as the resultant correlations between SA connectivity levels associated with impacted nodes. In the mean field approximation of IO stress, stress sources are modeled as random variables embedded within nodal state vectors. These random variables represent stochastic perturbations to a node’s communication capability state which are locally constrained to that node. Within this context, there is no imposed spatial correlation between the perturbations induced in different network nodes. In other words, there is no stress source existing external to and apart from a node. In order to better analyze the emergent network behavior induced by a more realistic IO threat, stress sources must

be introduced which can model an external perturbative field that might impact multiple nodes simultaneously. This enhanced CA model is discussed in the next section.

3.2 Localized External Stress Fields

3.2.1 Jammer Bomb Scenario

A more realistic form of IO stress arises from perturbations to nodal communication due to localized external fields that are generated by electromagnetic (EM) sources. In this second stress model, the CA dynamics are added by introducing spatially dispersed source cells representing generic radio frequency (RF) "jammer bombs" that act to locally jam platform-mounted radios passing by (Bothner 2000). In this case, the stress local to the i^{th} node at time t_n is

$$h_i^{\text{stress}}(t_n) = \sup_k \xi_{ik} B_k(t_n), \quad (19)$$

where "sup" is the *maximum* operator applied to all neighboring lattice cells k relative to the i^{th} node, the binary metric $B_k(t_n) = 1$ only when a jammer bomb occupies the k^{th} cell relative to cell i at time t_n , and ξ_{ik} is a binary coupling coefficient between the i^{th} and k^{th} lattice sites and is equal to 1 only when the k^{th} site is within an isotropic jamming neighborhood relative to the i^{th} site (i.e., the Euclidean distance from the centers of cell i to cell k $d_{ik} \leq 3.2$ cell lengths). Figure 13 illustrates the 36-cell jamming neighborhood (i.e., a center cell plus 36 neighboring cells) as implemented within the CA model. Then, the functional receive state of the i^{th} node is

$$s_i^R(t_n) = 1 - h_i^{\text{stress}}(t_n). \quad (20)$$

Thus, a node is jammed if it passes within the jamming neighborhood of a jammer bomb; alternately, a node is jammed if and only if there is at least one jammer bomb within an identical jamming neighborhood relative to the target node (where the target node is at the center of the translated neighborhood). The latter condition is necessary since the jammer bombs are designed to advance (downward) toward the brigade at a uniform steady rate of one cell per time step; this simulates the same relative movement as if the vehicle nodes were advancing (upward) towards the stationary bombs. Since the localized jammer bomb/node interactions act to perturb nodal reception capability states, this in turn perturbs nodal SA connectivity levels (as defined in equation [2]) throughout the CA brigade.

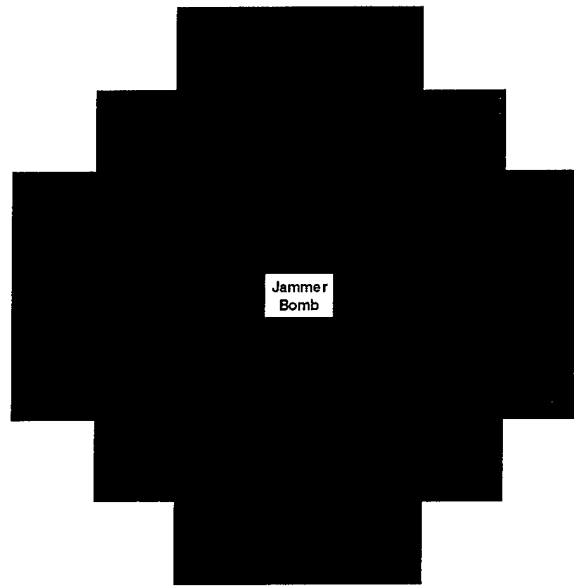


Figure 13. Jamming neighborhood for a generic RF jammer bomb.

3.2.2 Simulation Results

Once the CA model (which includes the cellular brigade configuration, rules defining cellular interdependencies required for nodal SA connectivity calculations, and additional rules defining local jammer bomb/node cell interactions) has been defined and coded,* the parameters that delineate an IO simulation can be specified. First, a simulation timescale must be defined. Given that the CA lattice has already been scaled to 125 m/cell length, only the uniform speed at which the brigade advances (remember that nodes do not move relative to each other, implying a uniform rate of forward movement) must be specified. If it is assumed that the uniform platform speed is 10 km/hr, then 1 simulation time step = $125 \text{ m} / (10,000 \text{ m} / 3,600 \text{ s}) = 45 \text{ s}$.

Next, the process of introducing jammer bombs into the scenario is described. In the current application of the model, we define a two-phase scenario, which entails (1) the digitized brigade encountering and moving through a field of jammer bombs previously planted by hostile forces, and (2) hostile unmanned aerial vehicles flying over the advancing brigade to deliver a barrage of additional jammer bombs. Figure 14 illustrates the pattern of cells within the CA lattice which are designated as either *potential* preplanted bomb sites or aerial barrage targets.

* See Appendix B for a listing of the jammer bomb scenario Cellang source code, and Appendix C for a listing of the data filter C source code that was co-compiled with the Cellang code.

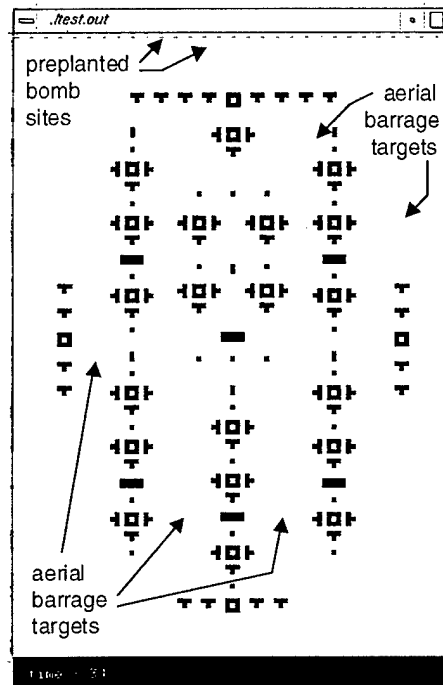


Figure 14. Potential jammer bomb site/target cells within the CA brigade.

The first phase of the IO scenario commences at $t_n = 0$ (i.e., the start of the simulation), and then continues to unfold until $t_n = 60$ min. During this time, jammer bombs are inserted into the row of dark gray cells labeled “preplanted bomb sites” at the top of Figure 14, where the probability of finding a jammer bomb at a cell, given that the cell is a potential bomb plant site, is $P(\text{bomb} | \text{plant site}) = 0.2/\text{time step}$, and the average preplanted bomb population within the CA lattice is 480 bombs/simulation trial. Once a jammer bomb has been inserted into a cell, the bomb moves forward into the brigade at a constant rate of 1 cell length/time step (125 m/45 s), and continues to jam friendly node platforms which penetrate its jamming neighborhood.* This is continued until either the bomb is destroyed by an advancing platform within the brigade (where $P_{\text{destroyed}} = 1$ given that the bomb and platform vehicle meet head-on), or the bomb’s battery is drained of voltage; this last event can occur anywhere from 37.5 to 52.5 min (reflecting a preset bomb lifetime) after the initial bomb is inserted into a plant cell. Thus, the net effect of this first scenario phase essentially results in a preexisting bomb field that evolves deterministically once it moves into the CA simulation window.

* It should be noted, however, that the potential bomb plant sites remain fixed relative to the brigade node cells.

The second phase of the IO scenario commences at $t_n = 48$ min, and then continues to unfold until $t_n = 96$ min (thus overlapping with a portion of the first phase). During this time, jammer bombs are inserted into the light gray cells labeled "aerial barrage targets," which are positioned in the open areas between node cells (see Figure 14). The probability of finding a jammer bomb at a cell, given that the cell is a potential aerial barrage target, is $P(\text{bomb}|\text{target site}) = 0.005/\text{time step}$, and the average air-dropped bomb population within the CA lattice is 163 bombs per simulation trial. As with the jammer bombs in the first phase of the scenario, bombs move forward at a constant rate of 1 cell length/time step,* and they continue to jam friendly nodes until the bomb is either destroyed or runs out of battery-generated power; in this phase, the latter event consistently occurs 75 min (reflecting the maximum bomb lifetime) after the bomb is inserted into a target cell. Thus, the second scenario phase essentially results in an evolving bomb field superimposed over a portion of the preplanted bomb field. Note that even though the bombs are stochastically introduced into the CA simulation window, they behave deterministically once introduced.

Twenty simulation trials of the two-phase jammer bomb scenario are run using the CA model, where each trial is seeded with a different random number (needed to stochastically vary the insertion of jammer bombs into both plant and target cells). Figure 15 displays six different "snapshots" taken at various time steps throughout one of the simulation trials. The color of a platform node cell indicates one of five fitness levels (as defined in Figure 7), and a black cell indicates the presence of a jammer bomb. The time reported in the black text box at the bottom of a snapshot reflects a dimensionless simulation time step, rather than the scaled time in units of minutes. In these snapshots, the following progression of events unfolds:

- Time = 3 min. The moving brigade approaches a preplanted field of jammer bombs.
- Time = 14 min. All of the platforms in the lead security force are jammed as this unit penetrates the bomb field, resulting in *very low* fitness. Even though they are not directly jammed, many of the platforms in the front half of the brigade are reduced from *very high* to *high* fitness as a result of jamming in the lead unit.
- Time = 43 min. The brigade continues to penetrate the bomb field. The lead security force prevents most of the bombs from reaching the center platforms within the brigade; however, fitness levels of these inner platforms are still reduced because of jamming in the front and sides of the brigade.

* As with the potential bomb plant sites, the aerial barrage target sites also remain fixed relative to brigade node cells.

Fitness State Color Code: ■ VERY HIGH ■ HIGH □ MEDIUM ■ LOW ■ VERY LOW

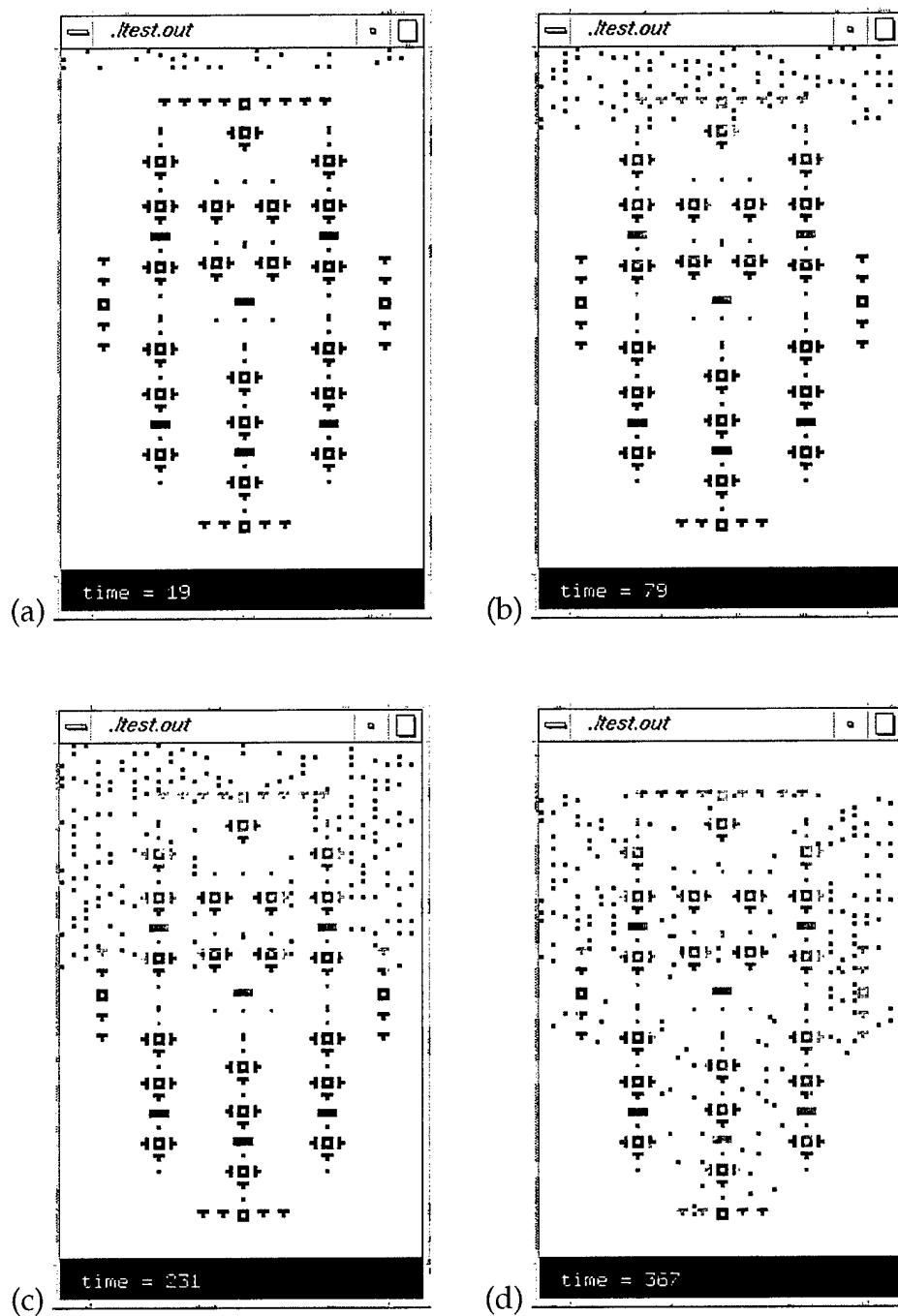


Figure 15. CA jammer bomb simulation snapshots: (a) time = 3 min; (b) time = 14 min; (c) time = 43 min; (d) time = 68 min.

Fitness State Color Code: ■ VERY HIGH ■ HIGH □ MEDIUM ■ LOW ■ VERY LOW

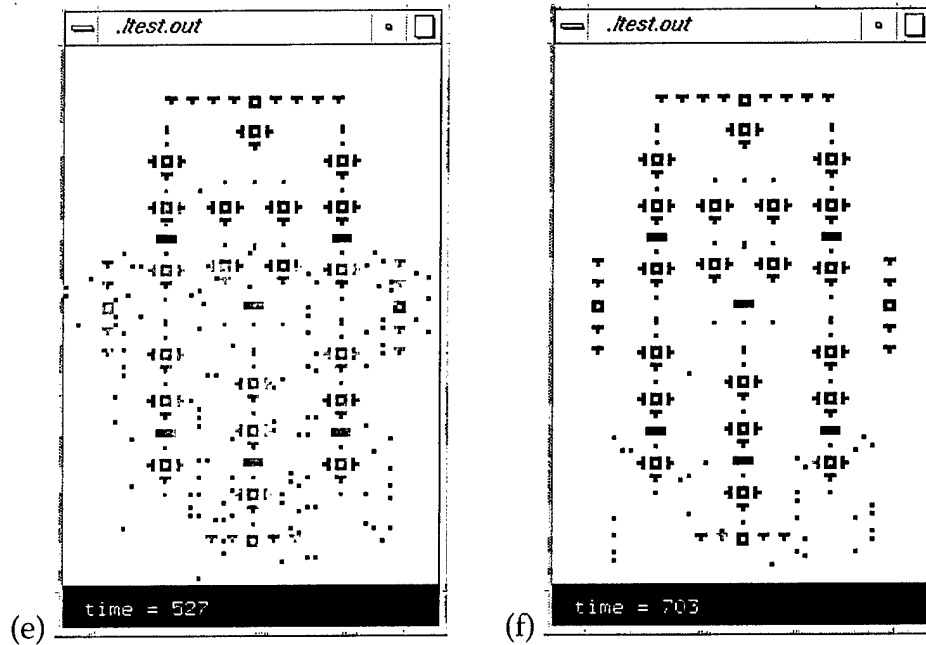


Figure 15. CA jammer bomb simulation snapshots: (e) time = 98 min; (f) time = 131 min (continued).

- Time = 68 min. As the lead security force reaches the end of the preplanted bomb field, an aerial barrage has recently commenced, resulting in an increased number of jammer bombs within the brigade. Many platforms operate under reduced fitness levels (high, medium, and very low) because they are either directly jammed or they lose communication channel continuity with neighboring platforms.
- Time = 98 min. Although most of the preplanted jammer bombs have expired or been destroyed (reverting most of the platforms in the front half of the brigade back to their original *very high* fitness states), the air-dropped bombs continue to degrade fitness levels of platforms in the back half of the brigade.
- Time = 131 min. Most of the brigade has moved beyond the air-dropped jammer bombs; fitness levels for these platforms have reverted back to *very high*, leaving only a few jammed or reduced-fitness platforms in the rear.

These snapshots were selected from a total of 225 time samples. They were generated within a single simulation run, where the uniform sampling rate is 1 time sample/45 s.

Figure 16 displays a plot of the time-dependent fraction of jammed network nodes, $\langle h^{stress}(t_n) \rangle$, resultant from the jammer bomb IO scenario. This scenario was run over multiple instances (where an instance refers to a single-trial stochastic initial placement of jammer bombs within the 2-D CA lattice) using the set of 20 sequential random seeds $\{0, 1, 2, \dots, 19\}$.^{*} The time series in this plot rises and then falls in a quasi-linear fashion (except for an exponential-like tail), with numerous stair-step regions indicating short-term equilibrium jamming conditions. In this time series, $\langle h^{stress}(t_n) \rangle$ seems to stabilize for finite intervals of time until advancing/retreating jammer bombs force the fraction of jammed nodes to adjust accordingly. Finally, the maximal value of $\langle h^{stress}(t_n) \rangle$ (0.21, which occurs at about 68 min into the scenario) roughly corresponds to the point of the maximal jammer bomb population within the CA brigade. This is also the point where preplanted bombs have completed their entry into the brigade from the front; thus, all new incoming bombs are stochastically introduced via the aerial barrage phase of the scenario.

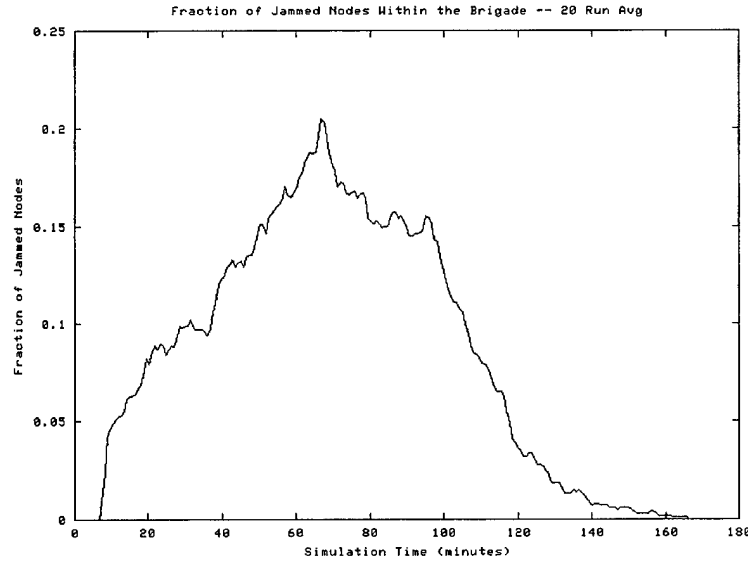


Figure 16. Fraction of jammed nodes as a function of scenario time t_n , for the CA brigade in the jammer bomb scenario.

Figure 17 is a plot of the dynamic globally averaged SA connectivity for the CA brigade, $\langle C(t_n) \rangle$, which is averaged over 20 simulation trials using the standard set of random seeds, where

$$\langle C(t_n) \rangle = \frac{1}{612} * \sum_{i=1}^{612} C_i(t_n) \quad (21)$$

^{*} Since this sequence of 20 random seeds will also be used in subsequent jammer bomb simulation runs for consistency purposes, it will henceforth be referred to as the *standard set of random seeds*.

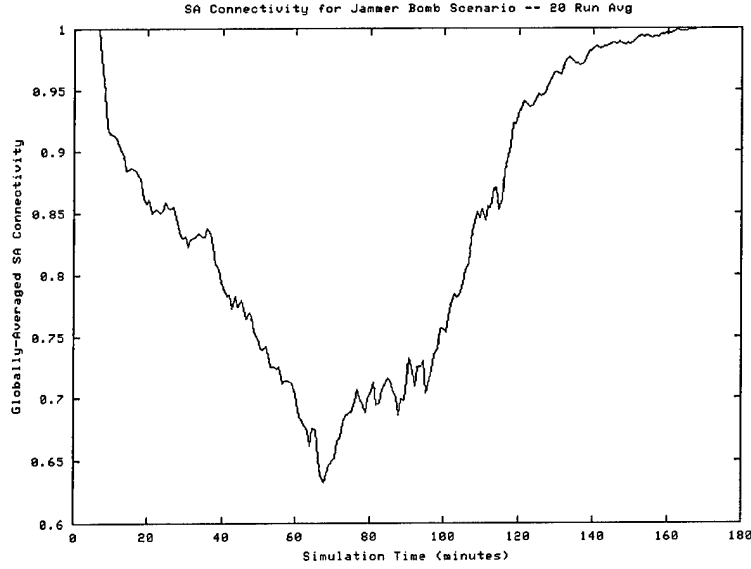


Figure 17. Dynamic globally averaged SA connectivity as a function of scenario time t_n , for the CA brigade in the jammer bomb scenario.

for each trial. Interestingly, the time series in this plot approximately resembles a mirror-image reflection of the $\langle h^{stress}(t_n) \rangle$ time series depicted in Figure 16, where the minimal value of $\langle C(t_n) \rangle$ (0.633) occurs concurrently with the maximal value of $\langle h^{stress}(t_n) \rangle$. As with the $\langle h^{stress}(t_n) \rangle$ time series, the SA connectivity stabilizes for finite intervals of time until advancing/retreating local jamming environments force $\langle C(t_n) \rangle$ to adjust accordingly.

Figure 18 shows plots of $P_{very\ high}(t_n)$, $P_{high}(t_n)$, $P_{medium}(t_n)$, $P_{low}(t_n)$, and $P_{very\ low}(t_n)$ as a function of scenario time t_n , where again $P_{fitness\ state}(t_n)$ is the fraction of nodes in a fitness state at time t_n . Each of the time series plots show how platform nodes simultaneously transition from the unperturbed *very high* fitness state down to either the *high* or *very low* fitness states for most nodes (with the remaining nodes transitioning to either the *medium* or *low* states). Then, there is an exponential return to the original *very high* state as the brigade moves beyond the jammer bombs. In particular, the $P_{very\ low}(t_n)$ time series, which rises and then falls in a fairly linear fashion (except for the exponential tail), accounts for those nodes whose SA connectivity is mainly impacted by local jamming. On the other hand, the $P_{high}(t_n)$ time series accounts for nodes which are mainly impacted by the loss of communication channel continuity with nonlocal neighbor nodes. Finally, for comparison purposes, Figure 19 displays the data from all of the 20 simulation

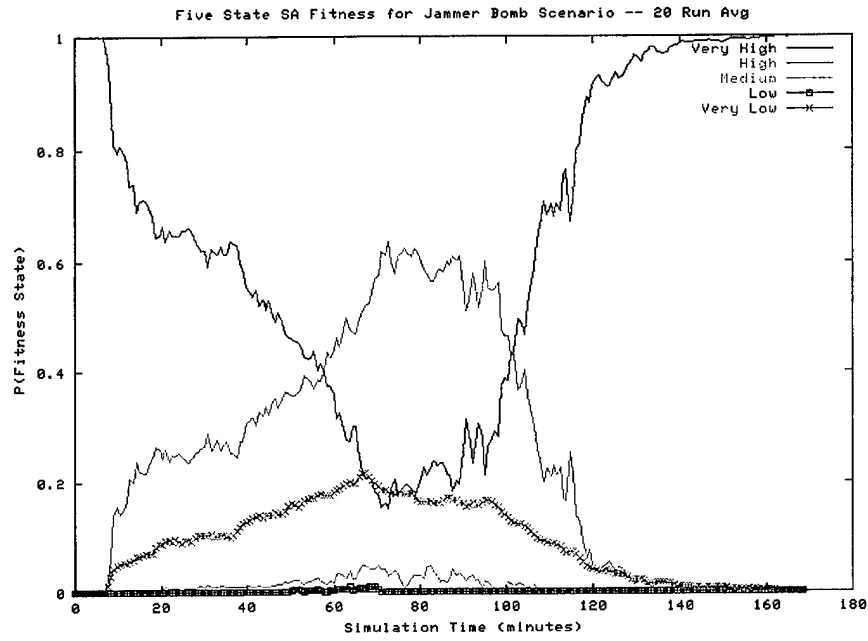


Figure 18. $P_{very\ high}(t_n)$, $P_{high}(t_n)$, $P_{medium}(t_n)$, $P_{low}(t_n)$, and $P_{very\ low}(t_n)$ as a function of scenario time t_n , for the CA brigade in the jammer bomb scenario.

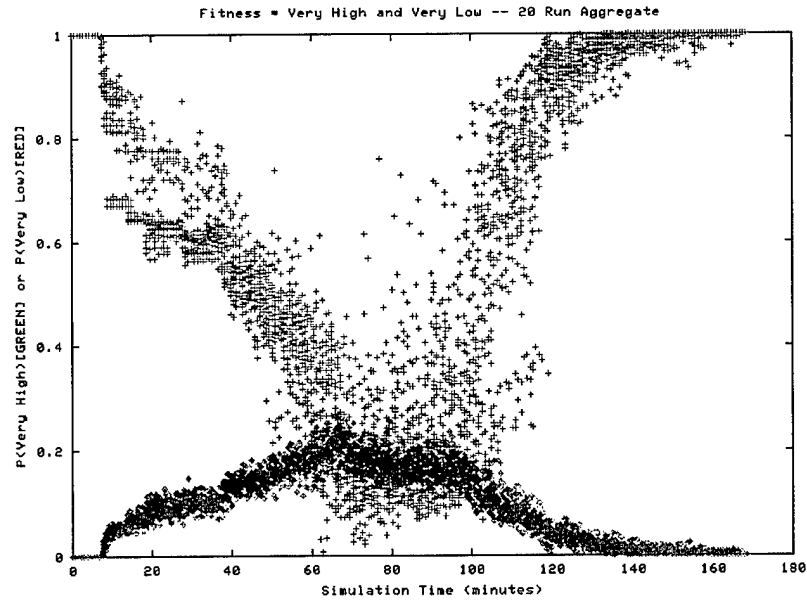


Figure 19. Superposition of $P_{very\ high}(t_n)$ (green points) and $P_{very\ low}(t_n)$ (red points) time series data from all 20 simulation trials.

trials for both the $P_{very\ high}(t_n)$ (green points) and $P_{very\ low}(t_n)$ (red points) time series, whose average values were plotted in Figure 18. Thus, in Figure 19, each of the green and red points represent, for one simulation instance, the fraction of nodes in the very high and very low fitness states, respectively, measured at a discrete simulation time step.

Figures 20 and 21 illustrate the coarse-grained state and temporal entropies

$H(t_n)$ and $H(t_n, t_{n-1})$, respectively, for the jammer bomb scenario. As introduced in section 2.6.4, “entropy” is a measure of the disorder/volatility within a multi-element system which can range from 0 up to 1. In both figures, the entropy is sampled according to three different node categories.

- *Global.* The sample population consists of all nodes within the brigade structure.
- *CSMA net.* The independent sample populations are separate battalion-level CSMA net communities (including the brigade-area CSMA net); after calculation, the net-specific entropies are averaged to come up with a single value.
- *Local net.* The independent sample populations are separate local SINCGARS net communities (including individual EPLRS-equipped self-servers, which are treated as local nets consisting of one member); after calculation, the net-specific entropies are again averaged to come up with a single value.

Both state and temporal entropies are sampled as a function of discrete simulation time step and averaged over 20 simulation runs using the standard set of random seeds.

The state entropy time series depicted in Figure 20 illustrate the progressive variation in successive nodal SA fitness levels as distributed among the five fitness states for global, CSMA net, and local net populations. In this context, $H(t_n) = 0$ indicates that all nodes within a sampling population occupy the same fitness state, while $H(t_n) = 1$ indicates that the nodes are equally distributed among the five possible fitness states. For the time series plots shown in the figure, the state entropy of the global population first jumps to a quasi-stable level between 0.4 and 0.5 until $t_n = 78$ –80 min, at which point the state entropy slightly decreases to a level between 0.35 and 0.4, and finally decreases to a level of 0 once all node vehicles have moved sufficiently beyond the last jammer bombs. On the other hand, the time series reflecting averaged state entropy for CSMA and local net populations both steadily reach peak levels of 0.35 and 0.28 at the transition time step $t_n = 78$ –80 min, at which point they steadily decrease down to a level of 0. Note that although both time series share very similar time profiles, the local net state entropy is always slightly less than that for the CSMA

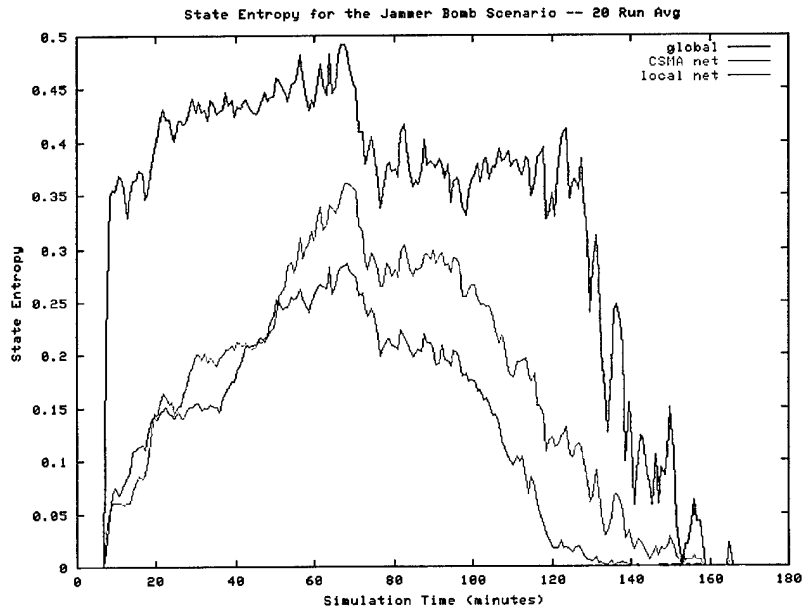


Figure 20. State entropy as a function of scenario time t_n , for the CA brigade in the jammer bomb scenario.

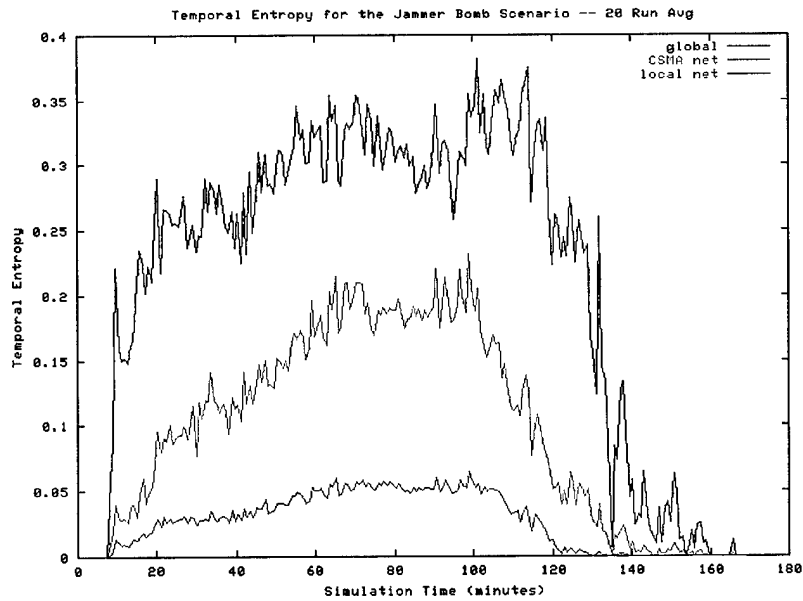


Figure 21. Temporal entropy as a function of scenario time t_n , for the CA brigade in the jammer bomb scenario.

nets. Taken together, the three time series describe a situation where SA fitness levels distributed across the nodes within a local SINCGARS net (and, to a lesser extent, across the nodes within a CSMA net community) are roughly coherent at each point in time, while SA levels distributed across the entire brigade structure are moderately varied.

The temporal entropy time series depicted in Figure 21 illustrates the persistence (or lack of) of nodal fitness levels across single time step transitions throughout the scenario run time. In this context, $H(t_n, t_{n-1}) = 0$ indicates that all nodes within a sampling population remain in the same fitness state from t_{n-1} to t_n , while $H(t_n, t_{n-1}) = 1$ indicates that the nodes are equally likely to transition to any of the five fitness states over this single-step time interval. As with the state entropy time series, the temporal entropy time series for each sampling population evolve to maximum levels, and then subsequently decrease back to a level of 0. However, in this case, the time series for the different sample populations are fairly spread out, with global, CSMA net, and local net temporal entropies achieving maximal levels of 0.38, 0.23, and 0.06, respectively. Taken together, the three temporal entropy time series illustrate an increasing trend in SA fitness level variability as a function of increasing sampling population size, from very stable SA levels across local nets up to moderately unstable SA fitness levels across the global brigade nodal population.

In the case of SA connectivity, stress-induced perturbations act to drive the global network state vector to various points within the SA connectivity phase space (as previously introduced in section 2.6.3). However, it is very difficult to explore this high-dimensional phase space for attractors; instead, we look for a *collective structure* within the coarse-grained fitness space by studying the various $P_{fitness}(t_n)$ time series that are generated during a simulation. This structure is said to demonstrate *non-trivial collective behavior* (NTCB), which was first reported by Chaté and Manneville as observed in the global dynamics of four-dimensional cellular automata (Chaté and Manneville 1991; Chaté and Manneville 1992).

Within the context of a return map, collective structure can be defined as a set of one or more points representing stationary $P_{fitness}$ levels towards which perturbative stress sources collectively drive the entire population of networked nodes. Since an IO threat scenario can posit an *open system*, where perturbative energy is stochastically introduced into a simulation from locations outside of the CA lattice, the interaction dynamics between network nodes and external stress sources can at best be deterministic *and* discontinuous. It should thus be noted that the discontinuous nature of the $P_{fitness}(t_n)$ time series will result in collective structures which can be "smeared" across variable-sized regions of fitness state space.

Figure 22 displays a return map associated with the jammer bomb scenario; this map was created with the $P_{very\ high}(t_n)$ (green points) and $P_{very\ low}(t_n)$ (red points)

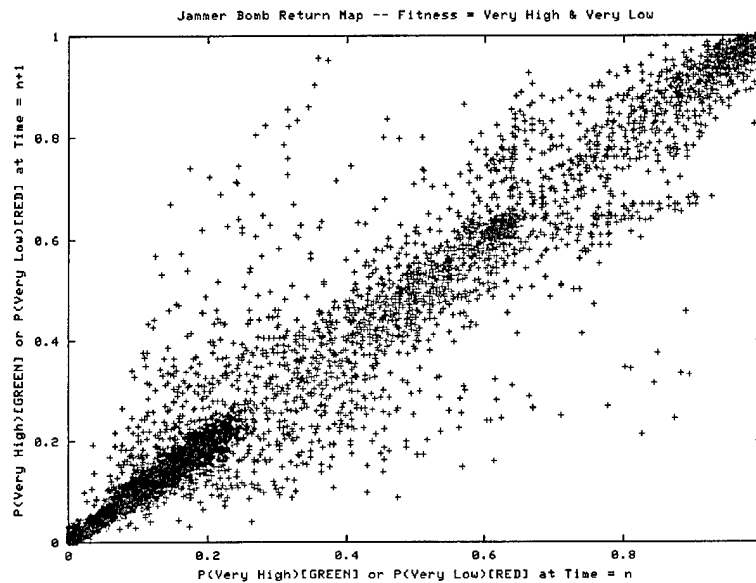


Figure 22. Return map for the $P_{very\ high}(t_n)$ (green points) and $P_{very\ low}(t_n)$ (red points) time series.

data previously shown in Figure 19. This return map illustrates the time series dynamics in a recursive rather than successive format. While the red points (clustered along a diagonal line with slope = 1) reinforce the linear behavior of the $P_{very\ low}(t_n)$ time series, the green points display the complex emergent behavior of the $P_{very\ high}(t_n)$ time series in what is clearly a collective structure. There are indications of what appears to be multimodal dynamics (symmetric branching along the diagonal line*) within certain intervals of $P_{very\ high}(t_n)$.

Although the collective structure associated with the $P_{very\ high}(t_n)$ time series resembles a period doubling bifurcation arising within a 1-D chaotic map (Ott 1993), the jammer bomb scenario clearly defines an open system (illustrated by the dispersive "smearing" of $P_{very\ high}(t_n)$ data points). Thus, the collective phenomena depicted in the figure cannot strictly qualify as deterministic chaos because the global system state at t_0 (i.e., initial positions of preplanted jammer bombs relative to the brigade) does not alone guarantee a determinable global SA connectivity state vector for $t_n > t_0$.

3.2.3 Echelon-Specific Nodal Sampling

By adjusting the scope of the node sampling populations within the CA model, it is possible to separately measure the time-series response of network nodes at the brigade, battalion, and combined company/platoon echelon levels

* This symmetric return map structure is very likely resultant from the vertical-axis spatial symmetry in the simple movement to contact brigade structure (Figure 3).

throughout the CA brigade. For example, Figures 23 and 24 display the dynamic fraction of jammed nodes and the globally averaged SA onnectivity, respectively, specifically sampled from brigade, battalion, and combined company/platoon echelon node populations using the standard set of random seeds. In Figure 23, the combined company/platoon echelon time series response is very similar to that of the global node population (see Figure 16), while battalion and brigade echelon time series display lower levels and/or later onset of significant levels of $\langle h^{stress}(t_n) \rangle$. This latter response results from many of the battalion and brigade nodes being positioned apart from the front and flanks of the advancing brigade structure. Figure 24 depicts similar relative behavior amongst the node populations, with brigade and combined company/ platoon nodes exhibiting very similar dynamic levels of SA connectivity as that calculated for the entire network node population (see Figure 17), while battalion nodes exhibit slightly higher levels of SA connectivity throughout the jammer bomb scenario.

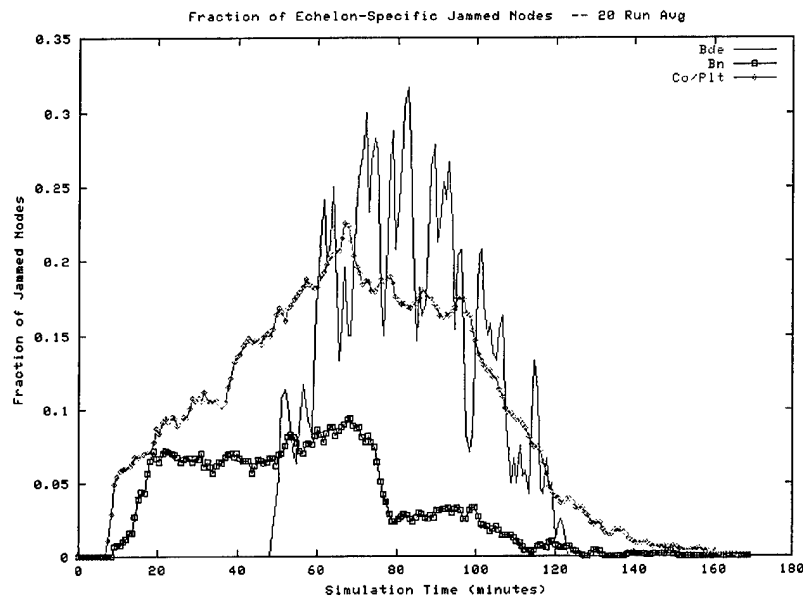


Figure 23. Fraction of jammed nodes as a function of scenario time t_n , for brigade (Bde), battalion (Bn), and combined company/platoon (Co-Plt) echelon nodes.

3.2.4 Sensitivity Investigation of Model Parameters

The CA model is constructed such that various sensitivity analyses can be performed on parameters, such as AO neighborhood size, jammer bomb residual lifetime, and jamming neighborhood size. In this section, the sensitivity of model parameters is demonstrated by varying the conditions of the jammer bomb IO scenario discussed in sections 3.2.1 and 3.2.2. Note that all simulation results presented here are generated by using the standard set of random seeds previously introduced in section 3.2.2.

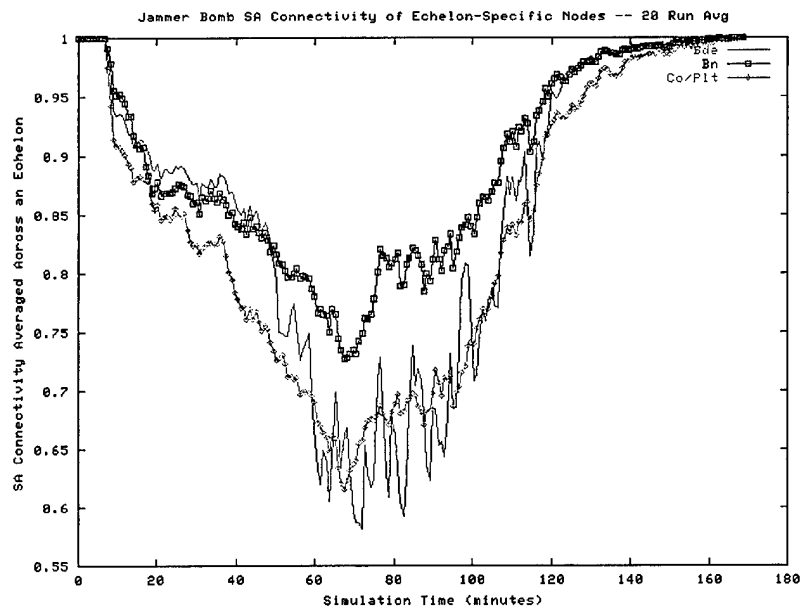


Figure 24. Globally averaged SA connectivity as a function of scenario time t_n for brigade (Bde), battalion (Bn), and combined company/platoon (Co-Plt) echelon nodes.

3.2.4.1 Reduced AO Neighborhood Scenario

In the first modified scenario, the AO neighborhoods originally associated with brigade, battalion, company, and platoon echelon nodes within the CA brigade (Figure 4) are each decreased by a factor of 0.5 in both x- and y-directions (thus decreasing total neighborhood areas by a factor of 0.25). These re-scaled AO neighborhoods are illustrated in Figure 25. As a result of these reduced neighborhoods, the connectivity distribution $P(N)$ for the CA brigade has also adjusted accordingly, as illustrated in Figure 26. It is interesting to note that this new connectivity distribution approximately resembles a power-law distribution (except for the large number of nodes with $N = 495$) characteristic of scale-free networks. In these networks, the connectivity distribution is described by $P(N) \sim N^{-\alpha}$, where $2.1 \leq \alpha \leq 4.0$ (Barabási and Albert 1999).

Figure 27 displays a plot comparing the time-dependent, globally averaged SA connectivity resultant from both the original jammer bomb and reduced AO neighborhood scenarios (a comparative plot of $\langle h^{stress}(t_n) \rangle$ vs. t_n is not presented since jammer bomb parameters remain unaltered from the original scenario). As is clearly demonstrated in the figure, uniformly reducing all AO neighborhood areas by a factor of 0.25 only serves to increase average SA connectivity levels by 10% or less. This result demonstrates the nonlinear response of overall SA network topology to changes in echelon-specific connectivity ranges.

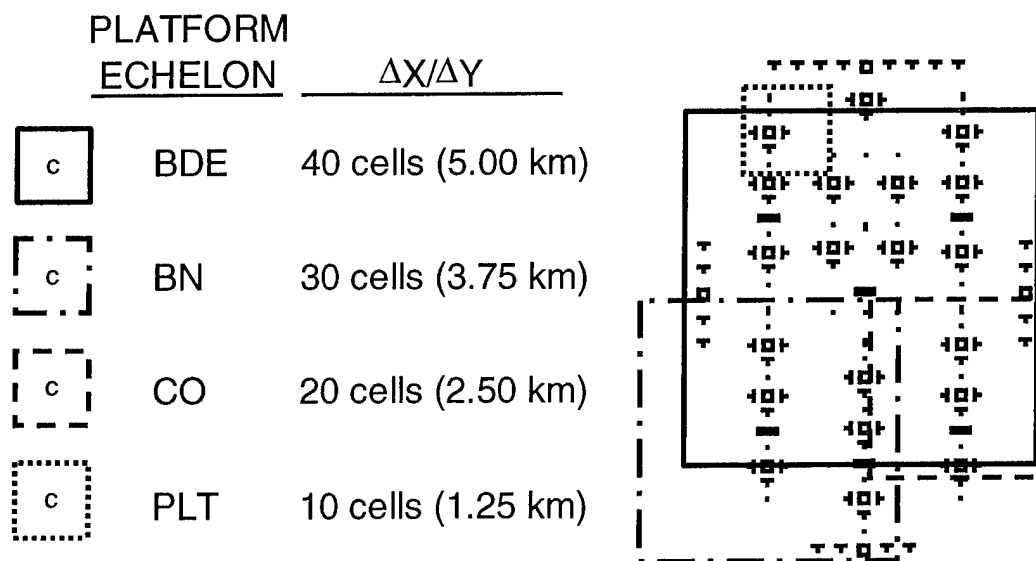


Figure 25. Reduced AO neighborhood scaling within the CA brigade.

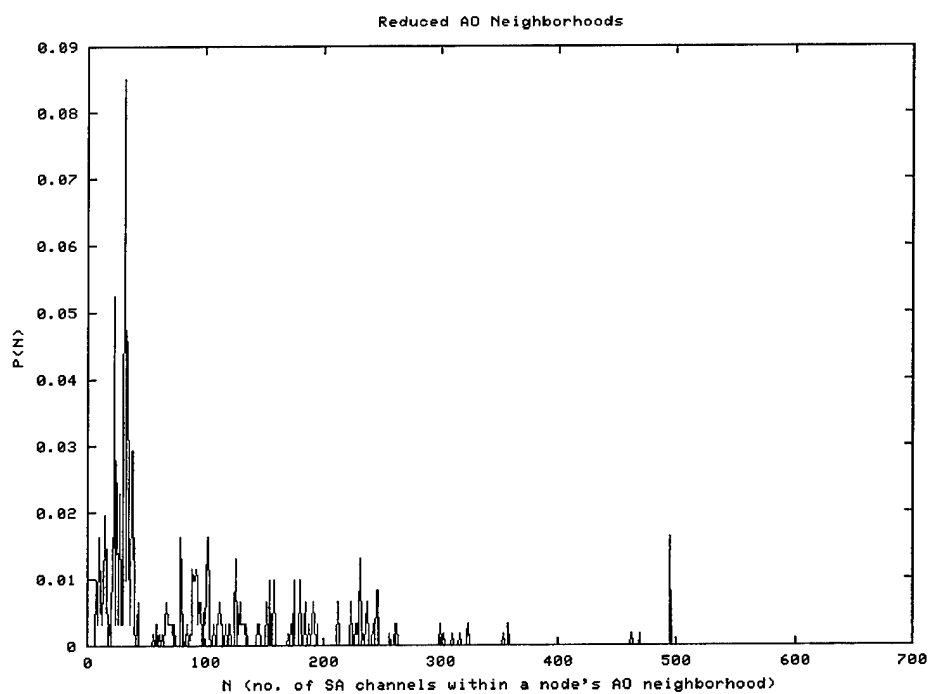


Figure 26. Connectivity distribution $P(N)$ for the CA brigade with reduced AO neighborhoods.

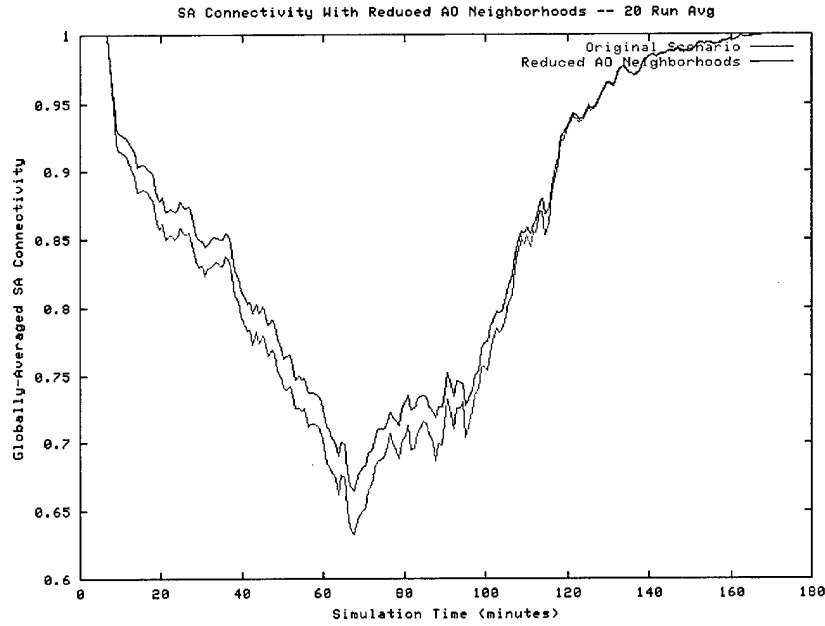


Figure 27. Globally averaged SA connectivity as a function of scenario time t_n , for both the original jammer bomb and reduced AO neighborhood scenarios.

Figure 28 displays time series plots of the fitness metrics $P_{very\ high}(t_n)$, $P_{high}(t_n)$, $P_{medium}(t_n)$, $P_{low}(t_n)$, and $P_{very\ low}(t_n)$ from the reduced AO neighborhood scenario. As with the SA connectivity time series depicted in Figure 27, the $P_{very\ high}(t_n)$ and $P_{high}(t_n)$ time series display a slight increase and decrease (respectively) in dynamic levels by factors between 0.05 and 0.10 when compared with similar measures from the original jammer bomb scenario (see Figure 18). The remaining three time series, however, essentially remain unaltered from the previous scenario results. This result demonstrates that the most significant response to modifying SA network connectivity ranges manifests in nonlocal interactions (i.e., those nodes which are not directly jammed but nevertheless suffer connectivity losses due to jammed neighbor nodes).

Figures 29 and 30 display state and temporal entropies, respectively, as a function of time step t_n for the reduced AO neighborhood scenario. In the case of measurements made with respect to the global nodal population, dynamic levels of both state and temporal entropies have significantly increased from those resultant from the original jammer bomb scenario, especially during the first 60 min of the scenario time window. This is likely due to a reduction in correlation between internodal SA connectivity levels, which results from decreased overlapping of AO neighborhoods associated with neighboring nodes. State and temporal entropies measured from CSMA and local net populations, however, essentially remain unchanged from the previous scenario.

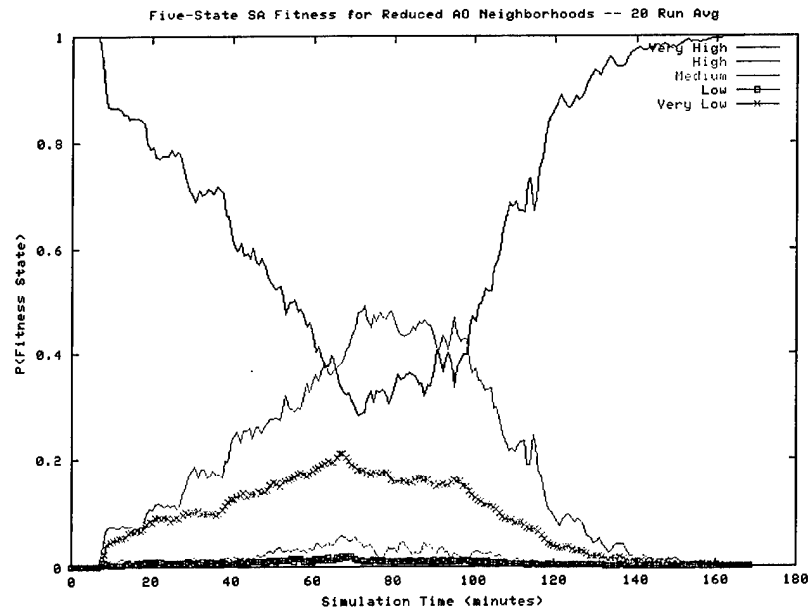


Figure 28. $P_{very\ high}(t_n)$, $P_{high}(t_n)$, $P_{medium}(t_n)$, $P_{low}(t_n)$, and $P_{very\ low}(t_n)$ as a function of scenario time t_n , for the reduced AO neighborhood scenario.

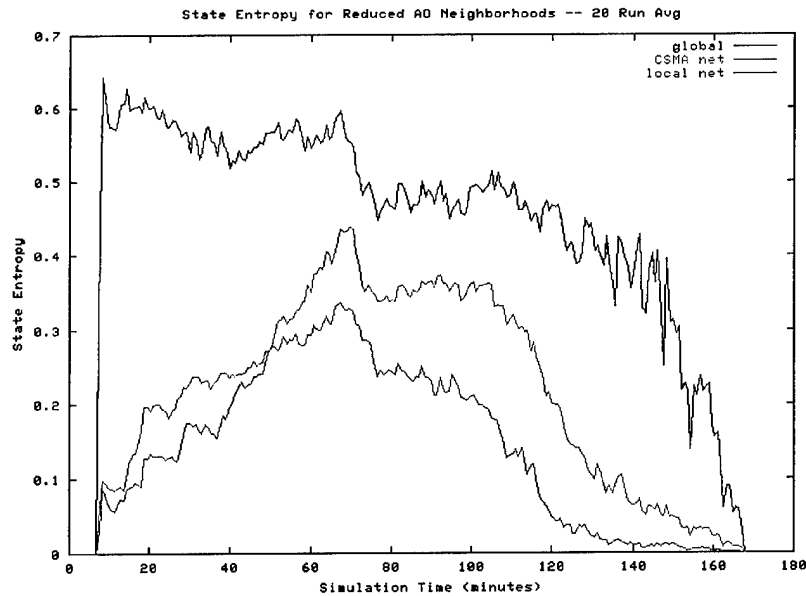


Figure 29. State entropy as a function of scenario time t_n , for the reduced AO neighborhood scenario.

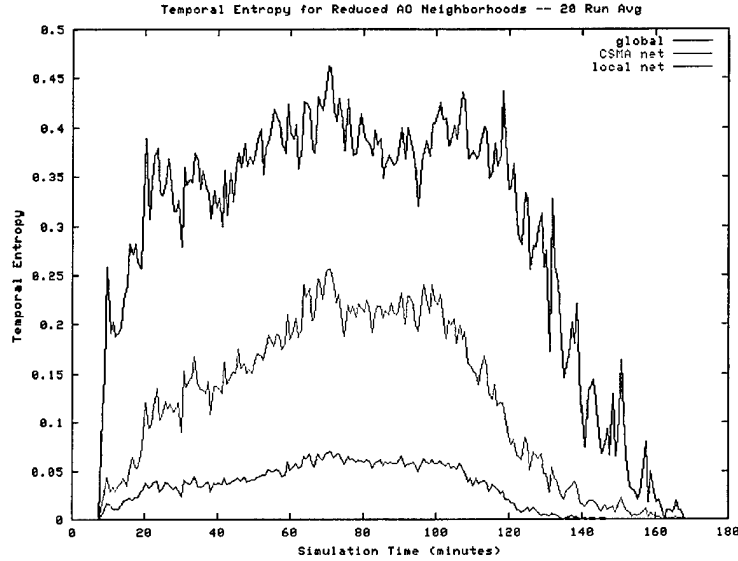


Figure 30. Temporal entropy as a function of scenario time t_n , for the reduced AO neighborhood scenario.

Figures 31 and 32 display the data from all of the 20 simulation trials of the reduced AO neighborhood scenario for both $P_{very\ high}(t_n)$ (green points) and $P_{very\ low}(t_n)$ (red points) in standard time series and return map formats, respectively. In Figure 31, each of the time series resembles its counterpart from the original jammer bomb scenario (Figure 19), except that the branching observed in the $P_{very\ high}(t_n)$ time series from the original scenario has disappeared. Similarly, the return map associated with the $P_{very\ high}(t_n)$ data depicted in Figure 32 has lost the definitive multimodal branching structure observed in the $P_{very\ high}(t_n)$ return map from the original scenario (Figure 22). These results demonstrate that the overall dynamic response of the stressed global networked system becomes more linear as the number of internodal connections is reduced.

3.2.4.2 Reduced Jammer Bomb Lifetime Scenario

In the second modified scenario, the jammer bomb scenario is next modified as follows:

- During the first phase of the scenario (i.e., preplanted bomb field), jammer bomb functional lifetimes are reduced by a factor of 0.5, now ranging from 18.75 to 26.25 min after a bomb is initially inserted into a plant cell.
- During the second phase of the scenario (i.e., aerial barrage), jammer bomb functional lifetimes are also reduced by a factor of 0.5, so that bombs now uniformly expire 37.5 min after being inserted into a target cell.

Note that the AO neighborhoods for all network nodes are restored to the original configuration, as depicted in Figure 4.

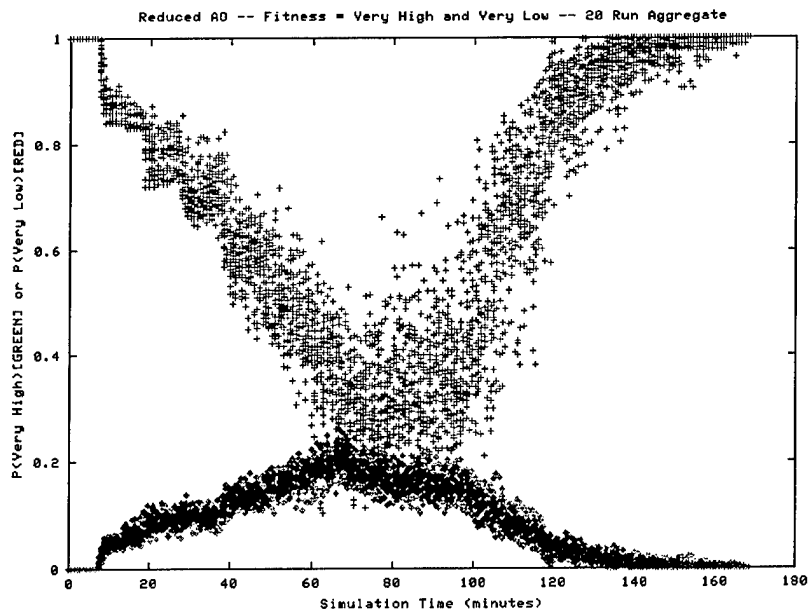


Figure 31. Superposition of $P_{very\ high}(t_n)$ (green points) and $P_{very\ low}(t_n)$ (red points) time series data from all 20 simulation trials from the reduced AO neighborhood scenario.

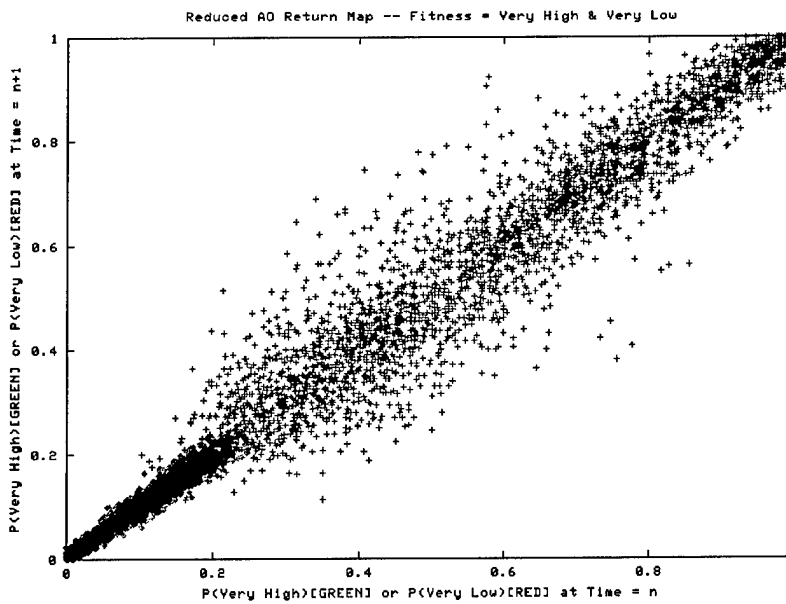


Figure 32. Return map for the $P_{very\ high}(t_n)$ (green points) and $P_{very\ low}(t_n)$ (red points) time series from the reduced AO neighborhood scenario.

Figure 33 displays a plot of the time-dependent fraction of jammed network nodes resultant from the reduced bomb lifetime scenario. The impact of shorter-lived jammer bombs is clearly illustrated in this figure, which compares $\langle h^{stress}(t_n) \rangle$ resultant from the current scenario with the time series response from the original scenario. Here, it is seen that the response is identical out until $t_n = 18.75$ min, the point where preplanted bombs begin to expire. From that point in time until about $t_n = 100$ min, the difference between the two time series ranges from about 25% to 50%, where reduced bomb lifetime significantly reduces $\langle h^{stress}(t_n) \rangle$. Of particular interest is an interval of apparently quasi-equilibrium behavior within the modified scenario time series (where $\langle h^{stress}(t_n) \rangle$ seems to temporarily stabilize), commencing at about $t_n = 20$ min (right after preplanted bombs begin to expire) and continuing until $t_n = 48$ min (the commencement of the aerial barrage phase). Finally, the improved network performance is less significant from $t_n = 100$ min to the end of the scenario, at which point most of the preplanted bombs in the original scenario have expired and the brigade gradually moves beyond the remaining bombs delivered during the recently concluded aerial barrage.

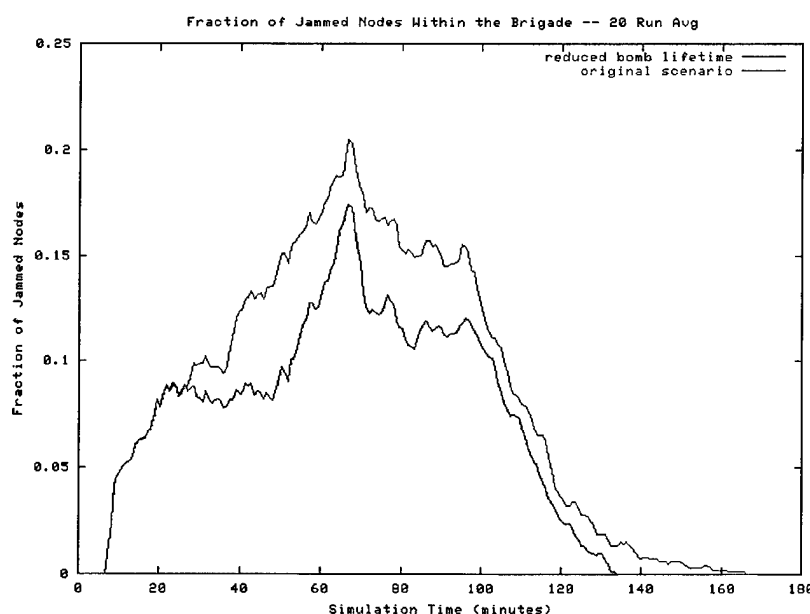


Figure 33. Fraction of jammed nodes as a function of scenario time t_n , for both the original and reduced jammer bomb lifetime scenarios.

Figures 34–37 depict time series of the globally averaged SA connectivity, five-state $P_{fitness}(t_n)$ metrics, state entropy, and temporal entropy, respectively, associated with the reduced bomb lifetime scenario. Each of these time series behaves very similarly to its counterpart from the original scenario, with the notable ubiquitous exception of the interval of quasi-equilibrium behavior first

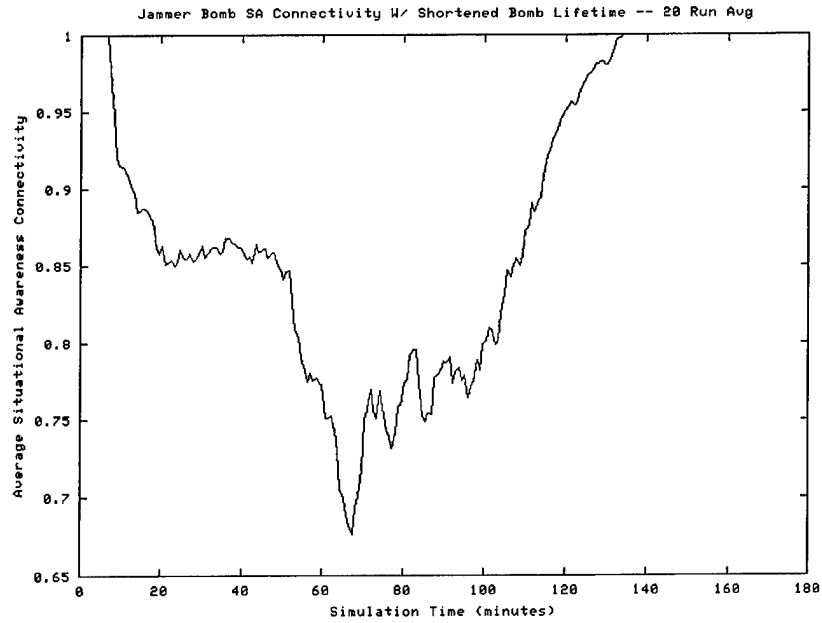


Figure 34. Globally averaged SA connectivity as a function of scenario time t_n , for the reduced jammer bomb lifetime scenario.

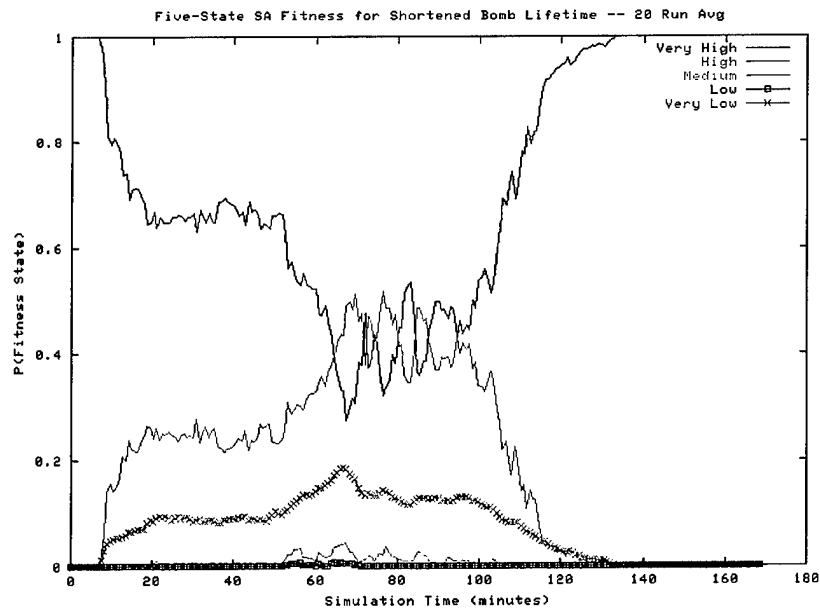


Figure 35. $P_{very\ high}(t_n)$, $P_{high}(t_n)$, $P_{medium}(t_n)$, $P_{low}(t_n)$, and $P_{very\ low}(t_n)$ as a function of scenario time t_n , for the reduced jammer bomb lifetime scenario.

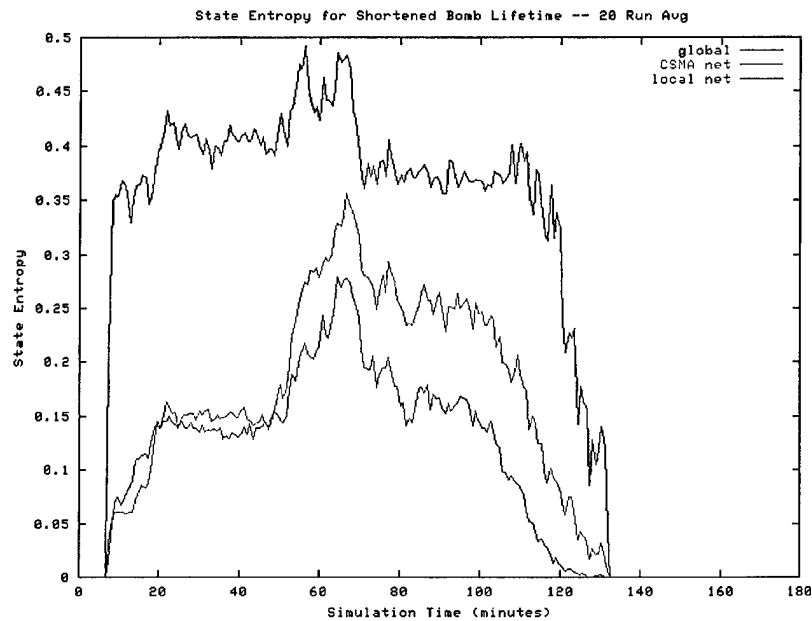


Figure 36. State entropy as a function of scenario time t_n for the reduced jammer bomb lifetime scenario.

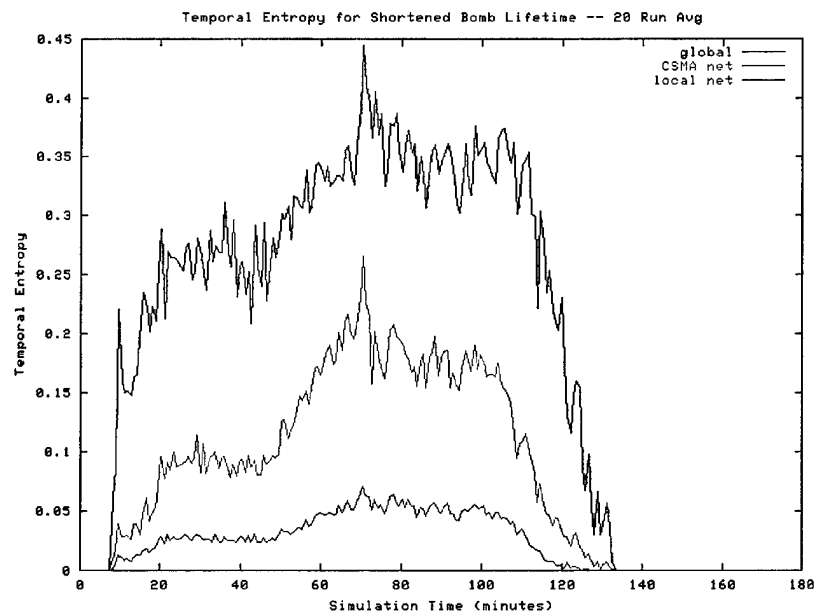


Figure 37. Temporal entropy as a function of scenario time t_n for the reduced jammer bomb lifetime scenario.

pointed out in the $\langle h^{stress}(t_n) \rangle$ time series from the modified scenario (Figure 33). In particular, the $P_{very\ high}(t_n)$, $P_{high}(t_n)$, and $P_{very\ low}(t_n)$ time series in Figure 35, the CSMA and local net state entropy time series in Figure 36, and the local net temporal entropy time series in Figure 37 are all remarkably level throughout this time interval, with the two state entropy time series essentially overlapping (indicating near-identical distributions of SA connectivity levels throughout both CSMA and local net nodal populations). Taken together, these plots reinforce the conclusion that SA connectivity levels have achieved a near-equilibrium condition by $t_n = 20$ min (especially within battalion-level and local SINCGARS net communities), which is perpetuated until the commencement of the aerial barrage phase of the jammer bomb scenario at $t_n = 48$ min.

Figures 38 and 39 display all data from the 20 simulation trials of the reduced jammer bomb lifetime scenario for both $P_{very\ high}(t_n)$ (green points) and $P_{very\ low}(t_n)$ (red points) in standard time series and return map formats, respectively. Again, each of the time series in Figure 38 resembles its counterpart from the original jammer bomb scenario, except that the branching observed in the $P_{very\ high}(t_n)$ time series from the original scenario has now sharply differentiated into two relatively level branches (with some additional distributed points resulting from variable bomb lifetimes) within the $t_n = 20$ min to 48 min time interval discussed previously. Similarly, the return map associated with the $P_{very\ high}(t_n)$ data depicted in Figure 39 retains the definitive multimodal branching structure first observed in the $P_{very\ high}(t_n)$ return map from the original scenario. In addition, the $P_{very\ high}(t_n)$ time series in Figure 38 exhibits more significant point dispersion from about $t_n = 70$ min to $t_n = 110$ min than is seen in the original scenario response, as well as a more abbreviated tail from $t_n = 110$ min to the end of the scenario. This behavior is illustrated from another perspective in the return map (Figure 39), where the lower branching structure appears to exhibit a wider spread than that observed in the return map associated with the original scenario. These results seem to indicate that the overall dynamic network response within the deterministic first phase of the jammer bomb scenario becomes more stable as jammer bomb lifetimes are reduced; this equilibrium then vanishes as the second phase commences (which is a characteristic of the discontinuous dynamics of this phase). Due to the emergent nature of this behavior, however, it is not appropriate to conclude that the first phase equilibrium would continue to manifest itself given even shorter bomb lifetimes without running further simulations.

3.2.4.3 Combined Moore Neighborhood/Increased Bomb Density Scenario

In the third modified scenario, the original jammer bomb scenario presented in sections 3.2.1 and 3.2.2 is modified as follows:

- The jamming neighborhood of all jammer bombs is reduced from 36 down to 8 neighboring cells.

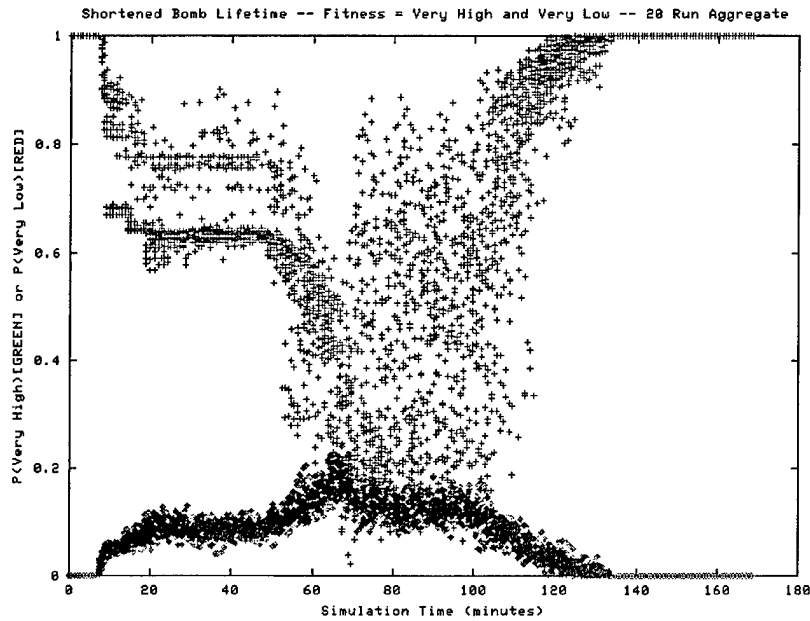


Figure 38. Superposition of $P_{very\ high}(t_n)$ (green points) and $P_{very\ low}(t_n)$ (red points) time series data from all 20 simulation trials from the reduced jammer bomb lifetime scenario.

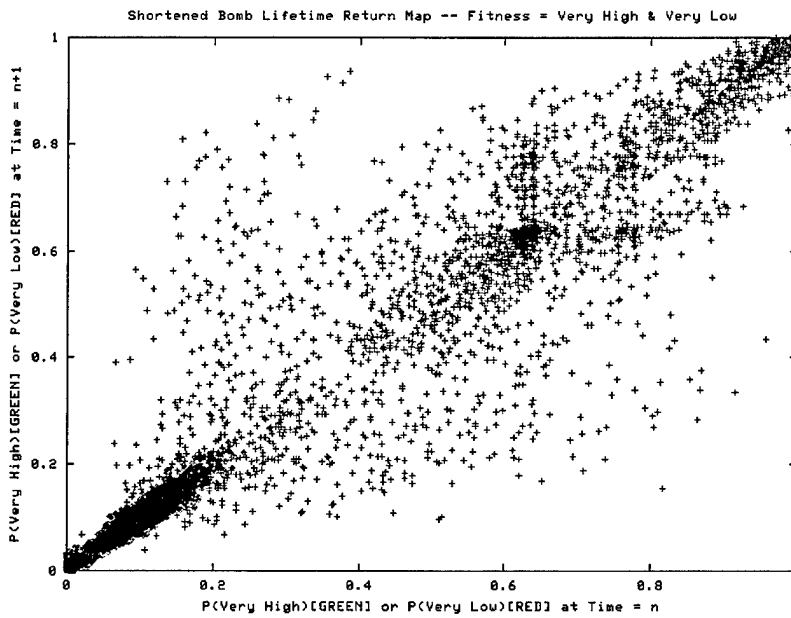


Figure 39. Return map for the $P_{very\ high}(t_n)$ (green points) and $P_{very\ low}(t_n)$ (red points) time series from the reduced jammer bomb lifetime scenario.

- The average density of preplanted jammer bombs is increased by a factor of five (i.e., from 480 to 2,400 bombs per simulation trial).
- The average density of air-dropped jammer bombs is increased by a factor of 10 (i.e., from 163 to 1,630 bombs per simulation trial).

Figure 40 depicts the modified jamming neighborhood as defined in the previous paragraph. As is clearly illustrated in the figure, the modified CA neighborhood includes only those neighboring cells that share either an edge or vertex with the center cell. This type of nine-cell configuration is conventionally known as a *Moore neighborhood* (see Figure 2[b]).

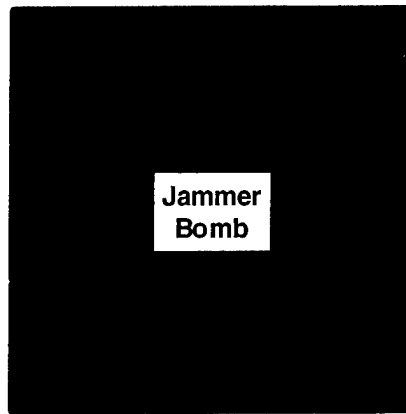


Figure 40. Reduced jamming neighborhood consisting of eight neighbor cells.

Figures 41 and 42 display plots of the time-dependent fraction of jammed network nodes and globally averaged SA connectivity resultant from the modified IO scenario, respectively. This modified scenario was run using the standard set of random seeds. Although the average net jamming area coverage per simulation trial has increased from the original to modified scenarios by factors of 1.2 and 2.4 for preplanted and air-dropped bombs, respectively, the associated changes in the fraction of jammed nodes and average SA connectivity are clearly nonlinear. Figure 41 illustrates this point by comparing the time-series response of the fraction of jammed nodes for both the original and modified jammer bomb scenarios. The high-frequency structure which emerges (in both plots) from $t_n = 8$ min to $t_n = 68$ min is due to a combination of a maximally allowable preplanted bomb population (where every one of the dark gray cells originally shown in Figure 14 is occupied by a jammer bomb during each advancing time step in the first scenario phase) and a jamming range which doesn't necessarily jam both the primary and alternate servers in a radio net (which was the case in the original scenario).

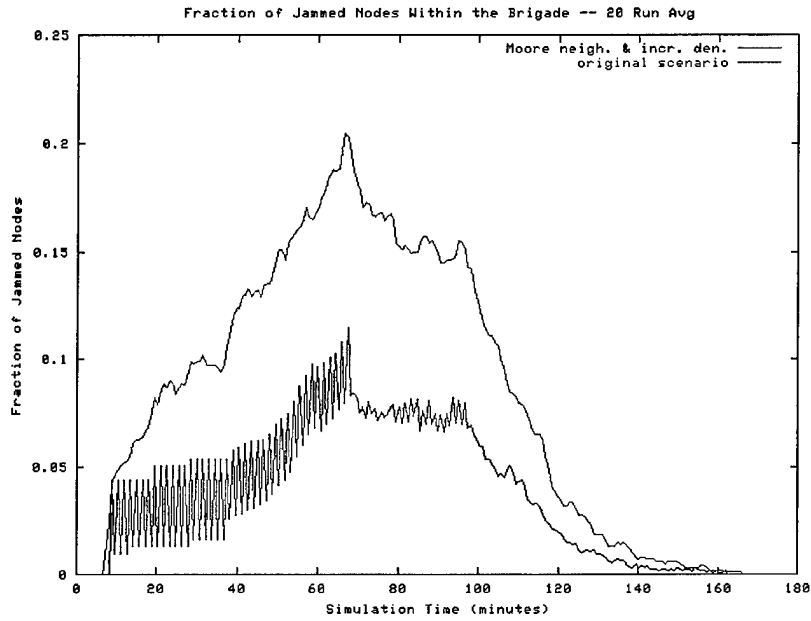


Figure 41. Comparison of fraction of jammed nodes as a function of time t_n for both the original jammer bomb and combined Moore neighborhood/increased bomb density scenarios.

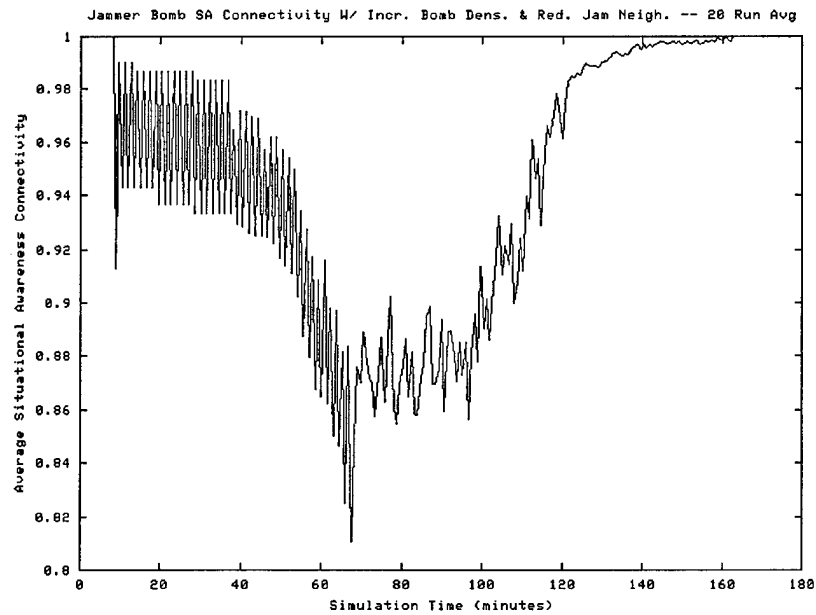


Figure 42. Dynamic globally averaged SA connectivity for the combined Moore neighborhood/increased bomb density scenario.

Figure 43 displays time series plots of the fitness metrics $P_{very\ high}(t_n)$, $P_{high}(t_n)$, $P_{medium}(t_n)$, $P_{low}(t_n)$, and $P_{very\ low}(t_n)$ from the combined Moore neighborhood/increased bomb density scenario. Here, the high-frequency structure observed in Figures 41 and 42 reemerges in the $P_{very\ high}(t_n)$ and $P_{very\ low}(t_n)$ time series out to $t_n = 68$ min. Then, it abruptly disappears when the $P_{high}(t_n)$ time series emerges, and from this time onward, the dynamics appear to follow a trend similar to that exhibited in the original jammer bomb scenario (see Figure 18). The $P_{very\ high}(t_n)$ and $P_{high}(t_n)$ time series exhibit an approximate reflective symmetry, while the $P_{very\ low}(t_n)$ time series decreases amplitude in a quasi-linear fashion. Because this abrupt transition in the time series dynamics is not evident in the original scenario, it reinforces the fact that global emergent behavior in a distributed interdependent system is usually very sensitive to any changes in model parameters.

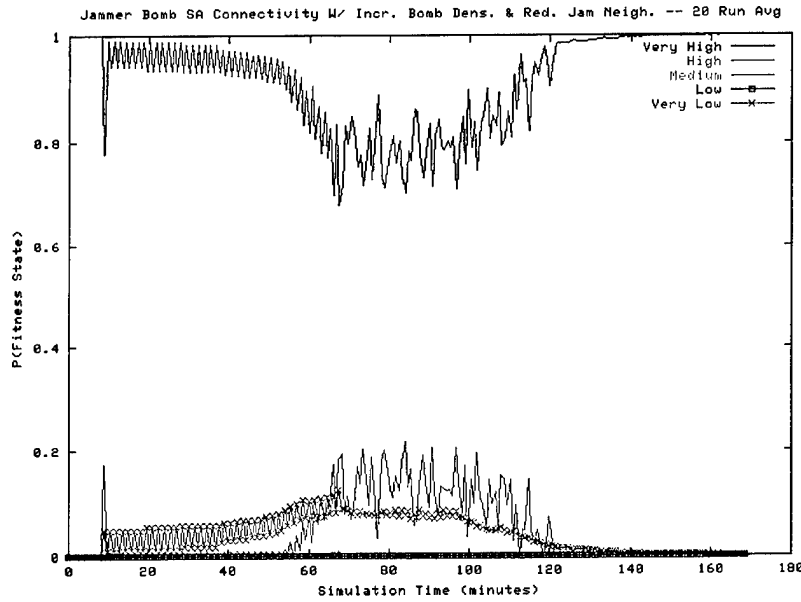


Figure 43. $P_{very\ high}(t_n)$, $P_{high}(t_n)$, $P_{medium}(t_n)$, $P_{low}(t_n)$, and $P_{very\ low}(t_n)$ as a function of time t_n , for the combined Moore neighborhood/increased bomb density scenario.

Figures 44 and 45 display state and temporal entropies, respectively, as a function of time step t_n for the combined Moore neighborhood/increased bomb density scenario. The high-frequency structure characteristic of this scenario again manifests itself within the plots in both figures from $t_n = 8$ min to $t_n = 68$ min. However, there is also an abrupt increase in amplitude in both state and temporal entropy time series plots occurring at about $t_n = 55$ min. This phenomenon is very likely catalyzed by the commencement of the aerial barrage phase of the scenario, where the steady-state stress supplied by the preplanted jammer bombs transitions (within the time interval $t_n = 48$ min to $t_n = 55$ min)

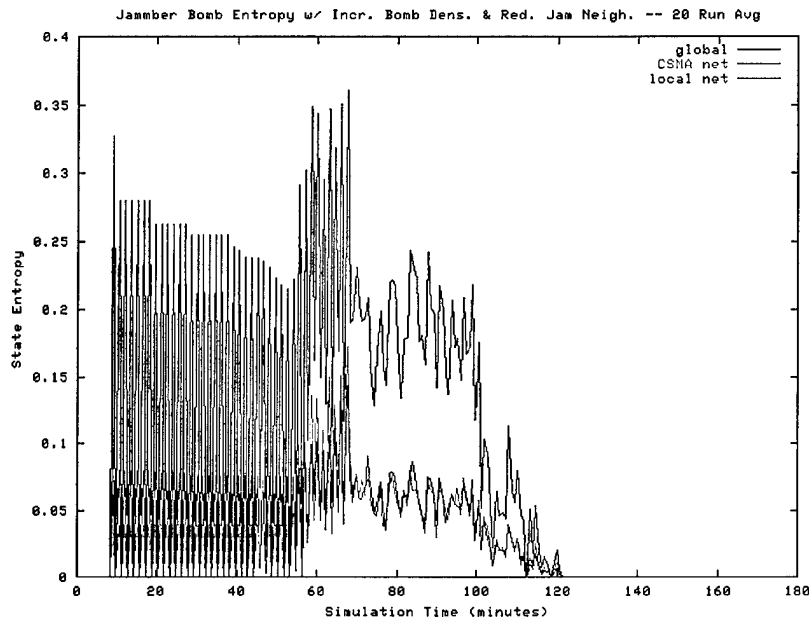


Figure 44. State entropy as a function of time t_n , for the combined Moore neighborhood/increased bomb density scenario.

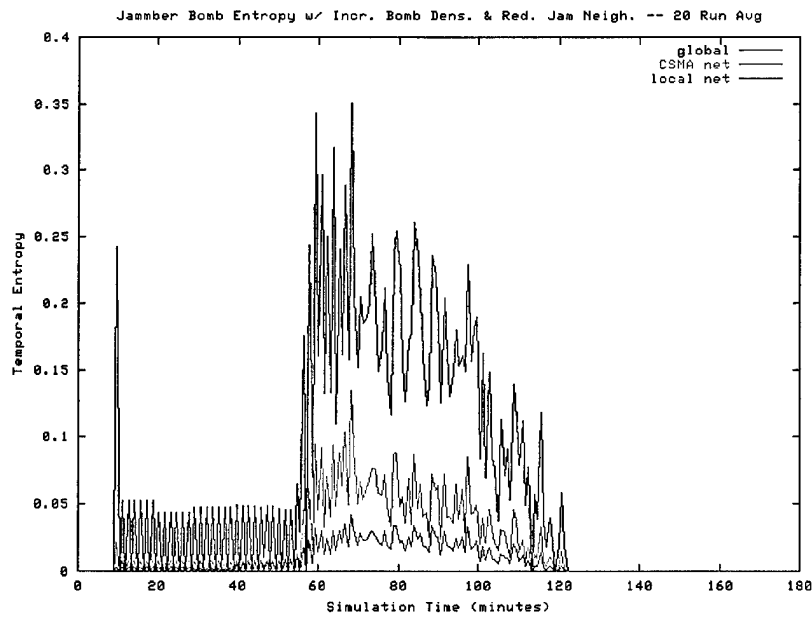


Figure 45. Temporal entropy as a function of time t_n , for the combined Moore neighborhood/increased bomb density scenario.

into a form stochastic in nature. This is especially noticeable in Figure 45, where the temporal entropies jump from low-amplitude (≤ 0.05 for brigade, CSMA net, and local net sampling populations) periodic time series to the higher amplitude and noisy format previously observed in the other jammer bomb scenario variants.

Figures 46 and 47 display the data from all of the 20 simulation trials for both $P_{very\ high}(t_n)$ (green points) and $P_{very\ low}(t_n)$ (red points) in standard time series and return map formats, respectively. In Figure 46, both time series exhibit a gradually increasing two-state periodic behavior from $t_n = 8$ min out to about $t_n = 40$ min within the scenario. This behavior then transitions into a dynamics exhibiting data clustering for both $P_{very\ high}(t_n)$ and $P_{very\ low}(t_n)$ (similar to what was observed in the original scenario and plotted in Figure 19), plus two additional new clusters for $P_{very\ high}(t_n)$ in the probability range from 0.3 to 0.5. As with the return maps associated with the previously discussed jammer bomb scenario variants, the return map shown in Figure 47 displays symmetry relative to the diagonal, with slope = 1 for both $P_{very\ high}(t_n)$ and $P_{very\ low}(t_n)$; in particular, the symmetric behavior of $P_{very\ high}(t_n)$ indicates that this particular time series exhibits quasi-periodic dynamics, where levels jump back and forth between the upper and middle data clusters. Thus, platform nodes principally impacted by local jamming (represented by the $P_{very\ low}(t_n)$ time series) consistently exhibit *very low* situational awareness. Other nodes, whose situational awareness is mainly impacted by communication loss with nonlocal neighbor nodes (represented by the $P_{very\ high}(t_n)$ and $P_{high}(t_n)$ time series), oscillate between *very high* and *high* levels of situational awareness.

3.2.5 Jammer Bomb Susceptibility Gradient

Up to this point, it has been assumed that the jamming neighborhood of a jammer bomb is isotropic. In other words, a node platform is assumed jammed if the Euclidian distance between the centers of the k^{th} jammer bomb and i^{th} node cells $d_{ik} < 3.2$ cell lengths. This assumption is predicated upon another simplifying assumption: that each node platform is *equally* susceptible to jamming from the notional jammer bomb stress source. A more realistic assumption might be that different types of node platforms each have their own specific *susceptibility threshold* to jamming, which is based on the incident power received by a node radiating from a jamming source. Thus, the original jammer bomb IO scenario is modified to account for variable platform susceptibility thresholds.

In this modification to the jammer bomb scenario, each bomb is equipped with a set of concentric, embedded jamming neighborhoods, as depicted in Figure 48. Each of the seven neighborhoods shown in this figure is defined by a unique d_{ik}

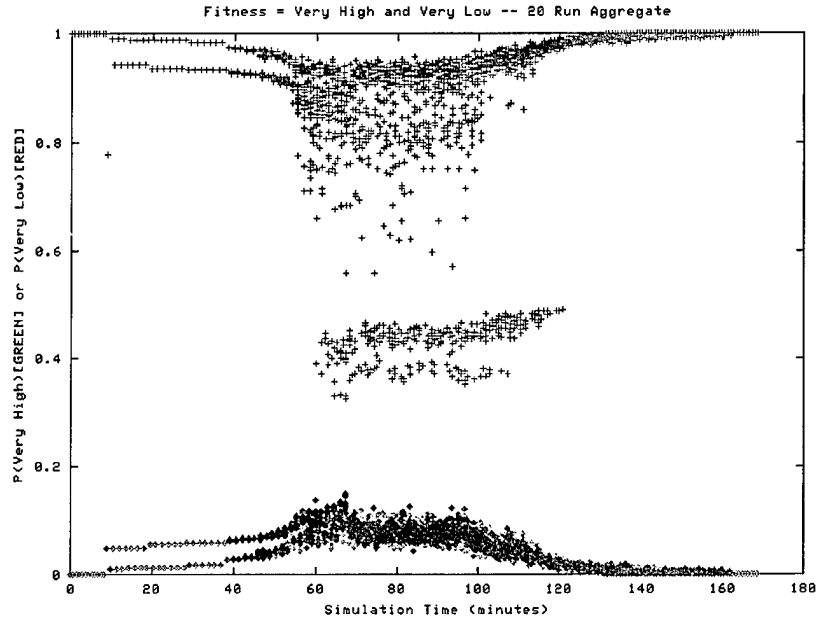


Figure 46. Superposition of $P_{very\ high}(t_n)$ (green points) and $P_{very\ low}(t_n)$ (red points) time series data from the combined Moore neighborhood/increased bomb density scenario.

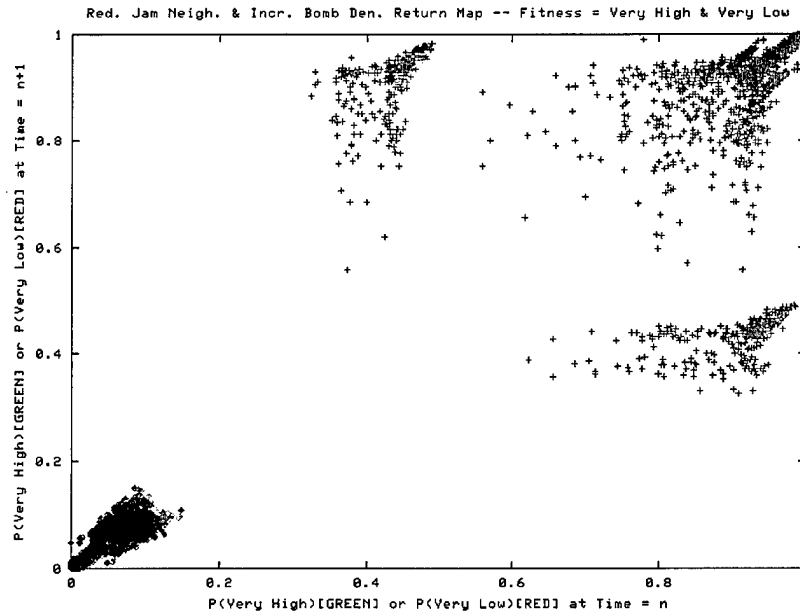


Figure 47. Return map for the $P_{very\ high}(t_n)$ (green points) and $P_{very\ low}(t_n)$ (red points) time series from the combined Moore neighborhood/increased bomb density scenario.

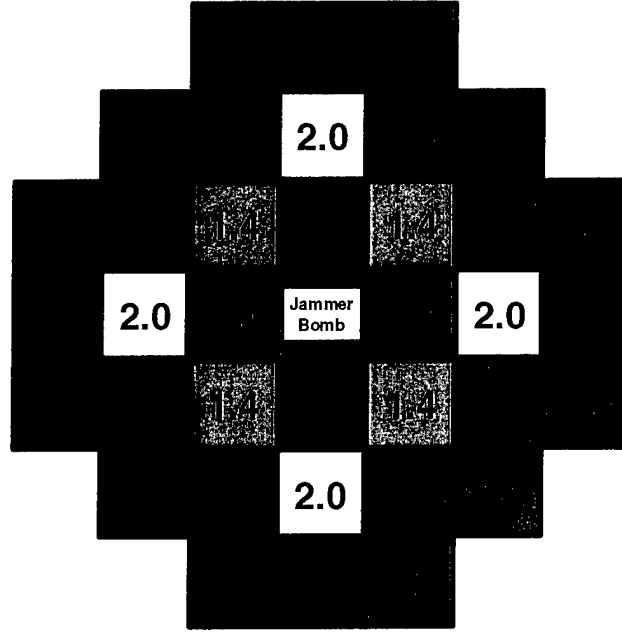


Figure 48. The seven concentric jamming neighborhoods used to define the discrete susceptibility gradient.

(where the i^{th} node can reside in any of the 36 lattice cells surrounding the k^{th} jammer bomb cell at the center) indicated by the numbers written within each cell, where $d_{ik} = 1.0, 1.4, 2.0, 2.2, 2.8, 3.0, 3.2$. In addition, each node within the brigade is assigned (based on sampling from a uniform random distribution) one of seven possible jamming susceptibility thresholds $S_{jamming}$, where $S_{jamming} = d_{ik}$ (thus associating a susceptibility threshold with a specific jamming neighborhood). Once assigned a susceptibility threshold, each node then also has an associated jamming resistance or “hardness” threshold $H_{jamming}$, where $H_{jamming} = 1/S_{jamming}$ and $0.3125 \leq H_{jamming} \leq 1.000$. Then, a node is jammed if and only if $S_{jamming} \geq d_{ik}$ (or equivalently, $H_{jamming} \leq d_{ik}$). The collection of concentric jamming neighborhoods, taken in conjunction with the set of threshold values for nodal susceptibility $S_{jamming}$, defines a discrete *susceptibility gradient*. It should be noted that this gradient is a notional construct that qualitatively represents a form of nodal jamming susceptibility which monotonically decreases as a function of increasing d_{ik} (where the received jammer power $P^{received} \propto 1/d_{ik}^{-2}$).

Figures 49–51 display the fraction of jammed nodes, globally averaged SA connectivity, and five-state fitness time series, respectively, averaged across multiple simulation runs (using the standard set of random seeds). The results depicted in these three plots are very similar in amplitude to those associated with the combined Moore neighborhood/increased bomb density scenario (see Figures 41–43 in section 3.2.4.3). The most obvious difference between the

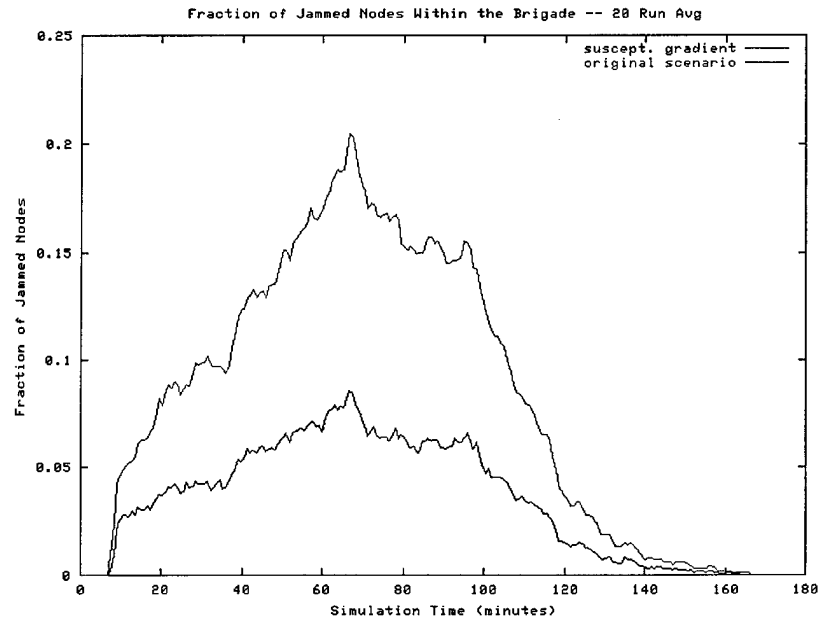


Figure 49. Comparison of fraction of jammed nodes as a function of time within the susceptibility gradient and original jammer bomb scenarios.

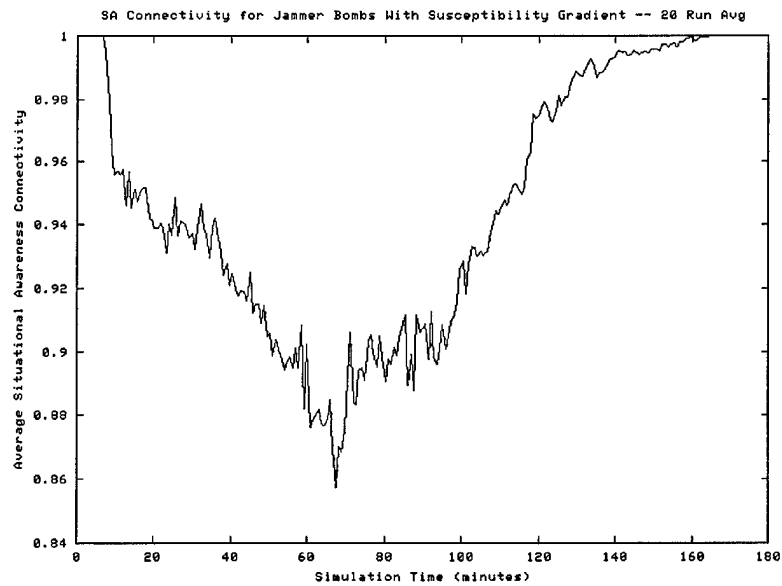


Figure 50. Dynamic globally averaged SA connectivity for the susceptibility gradient scenario.

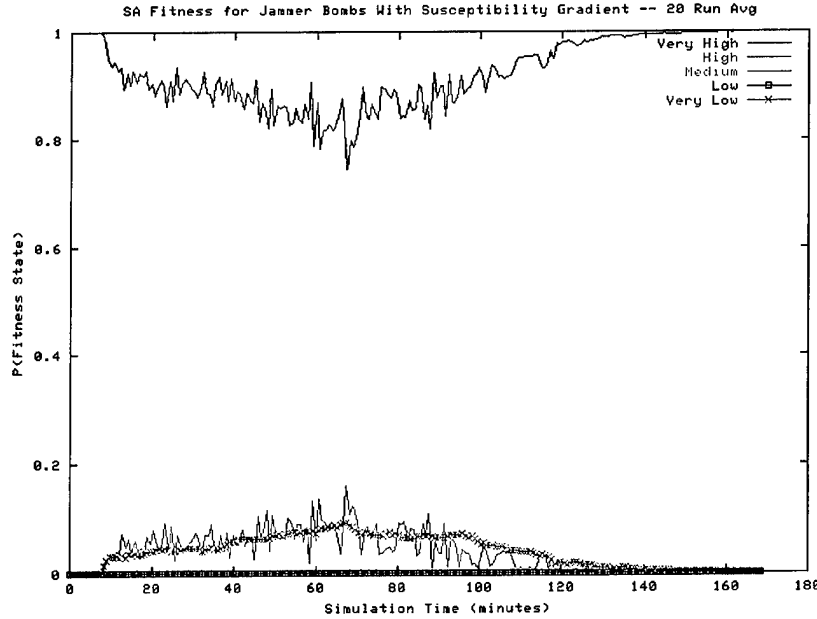


Figure 51. $P_{very\ high}(t_n)$, $P_{high}(t_n)$, $P_{medium}(t_n)$, $P_{low}(t_n)$, and $P_{very\ low}(t_n)$ as a function of time t_n for the susceptibility gradient scenario.

the two sets of time series plots is the lack in the susceptibility gradient plots of the periodic high-frequency structure observed in the first half of the previous scenario plots. However, from about $t_n = 68$ min to the termination of the scenario, plots from both scenarios demonstrate similar aperiodic low-amplitude structure. In the case of the current scenario, this aperiodic structure is likely due to the variation in susceptibility threshold levels amongst the jammed nodes, where locally clustered nodes within a jammer bomb neighborhood can no longer be assumed to be concurrently jammed.

Figures 52 and 53 display the state and temporal entropy time series, respectively, for the susceptibility gradient scenario, again averaged across simulation runs using the standard set of random seeds. As with similar plots associated with the original, reduced AO neighborhood, and reduced jammer bomb lifetime scenarios, both state and temporal entropies associated with the current scenario demonstrate distinct separation between brigade, CSMA net and local net sampling populations. Entropy amplitude levels are also comparable to those observed in each of the previous scenarios (including the combined Moore neighborhood/increased bomb density scenario). One, difference, however, is the tendency for the susceptibility gradient scenario entropies (especially those associated with the global brigade sampling population) to suddenly drop significantly in magnitude at about $t_n = 68$ to

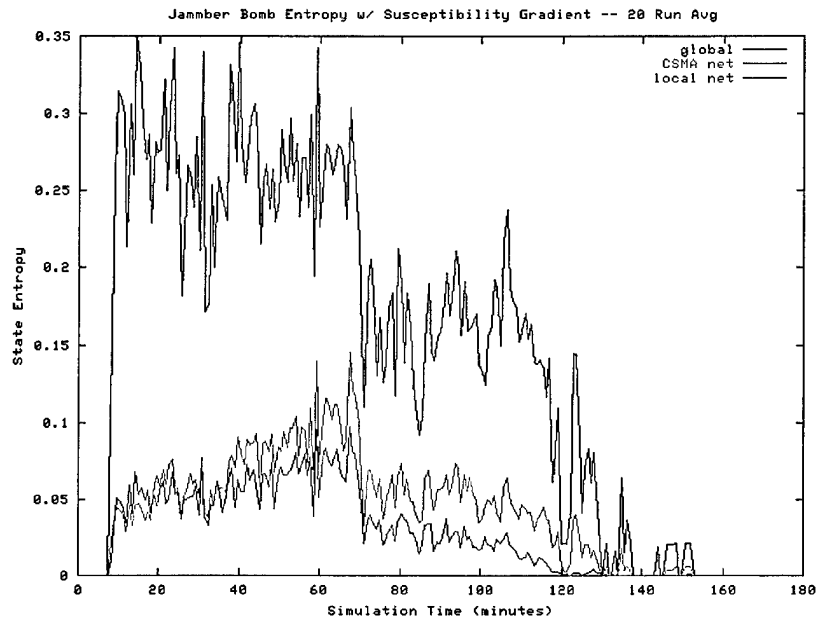


Figure 52. State entropy as a function of time t_n for the susceptibility gradient scenario.

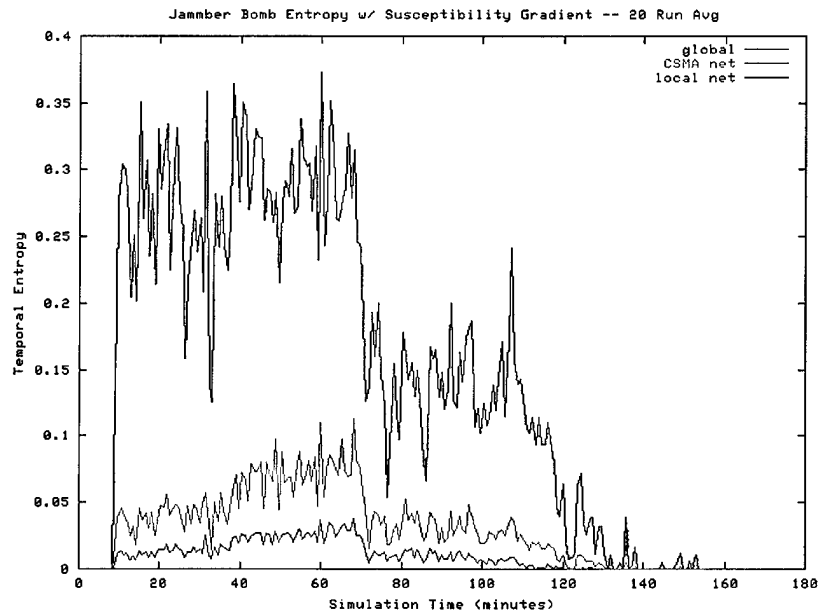


Figure 53. Temporal entropy as a function of time t_n for the susceptibility gradient scenario.

70 min. This sharp dropoff is clearly associated with the previously noted point in time within each jammer bomb scenario variant where maximal bomb density within the brigade structure is reached and then is persistently reduced, indicating a slight increase in ordered behavior as bombs switch from entering to exiting the brigade territory.

Figures 54 and 55 display all susceptibility gradient simulation trial data for both $P_{very\ high}(t_n)$ (green points) and $P_{very\ low}(t_n)$ (red points) in standard time series and return map formats, respectively. There is some general resemblance between these plots and corresponding plots associated with the combined Moore neighborhood/increased bomb density scenario (Figures 46 and 47). In both scenarios, most of the data within the time series plots demonstrate higher and lower levels of $P_{very\ high}(t_n)$ and $P_{very\ low}(t_n)$, respectively, than were observed within the other jammer bomb scenario variants. Also, the return maps associated with both the current and combined Moore neighborhood/increased bomb density scenarios demonstrate the metamorphosis of the diagonal cluster (and branching subclusters) of $P_{very\ high}(t_n)$ data points observed in other scenario return maps into smaller clusters symmetric with respect to the slope = 1 diagonal. However, the crisply defined periodic behavior that emerged in the early portion of the time series plot from the previous scenario has been replaced by two data clusters extending from $t_n = 8$ min out to about $t_n = 68$ (i.e., the point of maximum average jammer bomb density within the brigade). This change is also reflected in the collective structure depicted in Figure 55, where the three elongated clusters associated with the $P_{very\ high}(t_n)$ time series from the previous scenario (Figure 47) have metamorphosized into five compressed, ovoid-like clusters, which are again symmetric relative to the slope = 1 diagonal. The new data-smeared structure that emerges within the susceptibility gradient scenario plots is likely resultant from the previously mentioned variable susceptibility threshold levels assigned across the nodes within the brigade.

3.2.6 Disussion of Results

Results from each of the jammer bomb scenarios examined in this section indicate the following set of shared characteristics.

- The dynamic fraction of jammed node within the brigade, $\langle I^{stress}(t_n) \rangle$, increases (with small non-monotonic fluctuations) to a maximum value occurring at $t_n = 68$ min, remains in a state of quasi-equilibrium until about $t_n = 95$ min, and then steadily decreases back down to a level of 0.
- The dynamic SA connectivity averaged across all nodes within the brigade, $\langle C(t_n) \rangle$, decreases (again with small non-monotonic fluctuations) down to a minimum value again occurring at $t_n = 68$ min, again remains in a state of quasi-equilibrium until about $t_n = 95$ min, and then steadily increases back up to a level of 1.

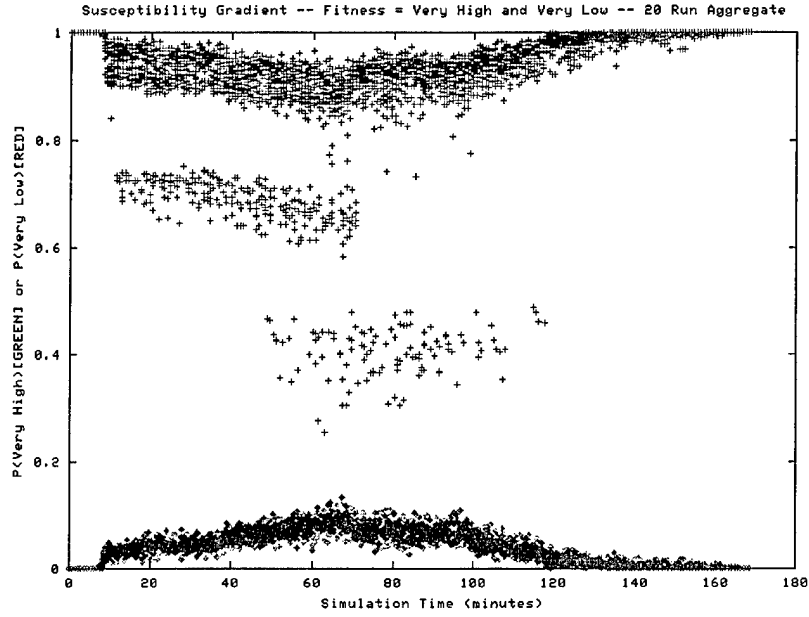


Figure 54. Superposition of $P_{very\ high}(t_n)$ (green points) and $P_{very\ low}(t_n)$ (red points) time series data from the susceptibility gradient scenario.

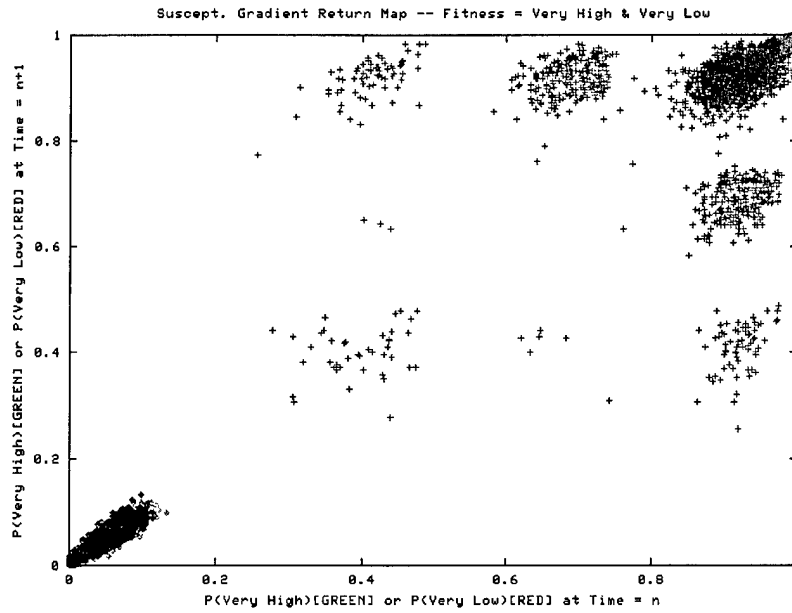


Figure 55. Return map for the $P_{very\ high}(t_n)$ (green points) and $P_{very\ low}(t_n)$ (red points) time series from the susceptibility gradient scenario.

- The SA fitness $P_{very\ low}(t_n)$ time series roughly track the scenario-respective $\langle h^{stress}(t_n) \rangle$ time series in terms of metric amplitude.
- The SA fitness $P_{very\ high}(t_n)$ time series roughly track the scenario-respective $\langle C(t_n) \rangle$ time series in terms of relative time series profile (with differences in metric amplitude).
- The SA fitness $P_{high}(t_n)$ time series profile is a mirror-image reflection of the respective SA fitness $P_{very\ high}(t_n)$ time series profile.
- The dynamic state entropy $H(t_n)$ and temporal entropy $H(t_n, t_{n-1})$ demonstrate separation between global brigade, CSMA net, and local net node populations (although $H(t_n)$ for CSMA and local net populations within a scenario often will track one another very closely).

Thus, in all scenario variants, the dynamic response of the brigade SA network tracks the invasive IO stress, as jammer bombs diffuse into and then out of the brigade structure. Also, this dynamic response is clearly partitioned between nodes which are directly jammed and other nodes which suffer SA connectivity degradation through jammed neighbors.

However, the jammer bomb scenario variants are distinguishable by their distinctive emergent behaviors as displayed within their respective $P_{very\ high}(t_n)$ time series. Although variations in AO neighborhoods and jammer bomb lifetimes lead to differences in collective structure as displayed within the return maps, the most noticeable factor observed within jammer bomb simulations appears to be the size of the jammer bomb neighborhood and its impact on $P_{very\ high}(t_n)$. The original 36-cell jamming neighborhood tends to drive the network to respond in a roughly linear fashion, with variable degrees of data spreading during the stochastic second phase of a simulation (as witnessed in both the $P_{very\ low}[t_n]$ and $P_{very\ high}[t_n]$ data). The smaller 8-cell Moore neighborhood, on the other hand, produces a more abrupt and discontinuous response in $P_{very\ high}(t_n)$ (but with little change in $P_{very\ low}[t_n]$). Finally, the susceptibility gradient scenario, which effectively applies a spectrum of jamming neighborhood sizes, results in network response almost as equally abrupt and discontinuous as seen with the 8-cell jamming neighborhoods. This suggests that the greater the jamming range of a stress source and the more ubiquitous this type of source (i.e., the higher the average susceptibility of network nodes), the more coordinated the overall network response (i.e., the less resistant is the network to associated perturbations to SA connectivity).

It is interesting to speculate on the possible existence of a specific jamming neighborhood size that results in a phase transition relative to $P_{very\ high}(t_n)$ (and $P_{high}[t_n]$ by association). This type of phenomenon, which has been observed in network models of complex systems (Sawhill and Kauffman 1996; Luque and Sole 1997), is analogous to a thermodynamic phase transition. In thermodynamics, a material's characteristic properties can suddenly change

when its temperature is raised or lowered by a small amount; the temperature at which the change in properties occurs is called the critical temperature T_c . Similarly, within the context of an SA network operating within a scenario with uniform jammer bombs (i.e., all bombs consistently radiate at the same power level for a fixed lifetime) and susceptibility gradients, one could define the *critical jamming susceptibility threshold* $S_{c,jamming}$ relative to the average network susceptibility threshold $\langle S_{jamming} \rangle$, where

$$\langle S^{jamming} \rangle = \frac{1}{N} \sum_{i=1}^N S_i, \quad (22)$$

N = total number of network nodes within the brigade, and S_i = susceptibility threshold of the i^{th} node. Then, given the existence of a phase transition relative to $P_{very\ high}(t_n)$, two network SA connectivity phases might emerge: (1) an uncorrelated decoherent collective structure arising when $\langle S_{jamming} \rangle < S_{c,jamming}$, and (2) a quasi-linear coherent collective structure arising when $\langle S_{jamming} \rangle \geq S_{c,jamming}$. Since nodal susceptibility thresholds are directly correlated to a range of embedded jamming neighborhoods, $S_{c,jamming}$ would be associated with an average jamming neighborhood size somewhere between 4 and 36 cells (where $1.0 < \langle S_{jamming} \rangle < 3.2$; see section 3.2.5).

4. Conclusions

A CA model has been developed for the analysis of the dynamics of situational awareness network topology (relative to a receiving platform) which emerges on a global scale within a digitized brigade exposed to IO stress. Two different forms of stress were modeled, including (1) a mean field stress applied uniformly to all brigade nodes and (2) RF jammer bombs exhibiting local fields within the brigade structure. In particular, analysis of the fitness state data generated by the jammer bomb IO scenarios has demonstrated that local threat/target interactions affect inter-platform dependencies simultaneously on various scales. If a portion of the CA brigade (such as a battalion) were to be geographically separated from the rest of the brigade, levels of SA connectivity throughout both of the separated parts would be impacted as a function of node platform echelon, so that jamming in one of the separated parts would affect nodal situational awareness (especially in higher echelon platforms) in the other part differently than if the parts were unified. Thus, the emergent behavior of the stressed brigade network must be analyzed at the holistic macroscale level.

The spatio-temporal dynamics of other types of dispersed IO stress sources which modify nodal transmit/receive functionality are likely to create similar modes of emergent global behavior in situational awareness levels which cannot be anticipated by studying isolated platforms or even local area networks

(LANs). Given that the threat/target interaction dynamics of an IO stress source can be discretized in space and time at the mesoscale level, it is a fairly straightforward process to program the stress source into the CA model within the context of a scenario. There is also no limitation on the number of different types of IO stress sources which can be programmed into a single scenario given that the discrete spatio-temporal interaction dynamics are known, although the complexity of multi-threat scenario sensitivity analyses can rapidly increase as a function of inter-threat synergy.

5. References

- Bak, P. *How Nature Works: The Science of Self-Organized Criticality*. New York: Copernicus, 1996.
- Bak, P., and K. Sneppen. "Punctuated Equilibrium and Criticality in a Simple Model of Evolution." *Physical Review Letters*, vol. 71, pp. 4083–4086, 1993.
- Barabási, A. L., and R. Albert. "Emergence of Scaling in Random Networks." *Science*, vol. 286, pp. 509–511, 15 October 1999.
- Bar-Yam, Y. *Dynamics of Complex Systems*. Reading, MA: Perseus Books, pp. 420–471, 1997.
- Bothner, P. Personal communication. U.S. Army Research Laboratory, Aberdeen Proving Ground, MD, April 2000.
- Chaté, H., and P. Manneville. "Evidence of Collective Behavior in Cellular Automata." *Europhysics Letters*, vol. 14, pp. 409–413, 1991.
- Chaté, H., and P. Manneville. "Collective Behaviors in Spatially Extended Systems With Local Interactions and Synchronous Updating." *Progress in Theoretical Physics*, vol. 87, no. 1, pp. 1–60, 1992.
- Eckart, J. D. "Cellang 2.0: Language Reference Manual." *ACM SIGPLAN Notices*, vol. 27, no. 8, August 1992a.
- Eckart, J. D. "A Cellular Automata Simulation System: Version 2.0." *ACM SIGPLAN Notices*, vol. 27, no. 8, August 1992b.
- Galam, S. "Rational Group Decision Making: A Random Field Ising Model at $T = 0$." *Physica A*, vol. 238, p. 66, 1997.
- Guzie, G. L. "Vulnerability Risk Assessment." pp. 21–32, ARL-TR-1045, White Sands Missile Range, NM, June 2000.
- Luque, B., and R. V. Sole. "Phase Transitions in Random Networks: Simple Analytic Determination of Critical Points." *Physical Review E*, vol. 55, no. 4, pp. 257–260, 1997.
- Newman, M. E. J., and K. Sneppen. "Avalanches, Scaling, and Coherent Noise." *Physical Review E*, vol. 54, pp. 6226–6231, 1996.
- Ott, E. *Chaos in Dynamical Systems*. New York: Cambridge University Press, pp. 23–68, 1993.

- Sawhill, B. K., and S. A. Kauffman. "Phase Transitions in Logic Networks." Proceedings of the Workshop on Control Mechanisms for Complex Systems: Issues of Measurement and Semiotic Analysis, Las Cruces, NM, December 1996.
- U.S. Army Communication Electronics Command. "Next Generation Performance Model Situational Awareness Design Document Version 1.5." Command and Control Division, Ft. Monmouth, NJ, January 1999.
- U.S. Army Training and Doctrine Command. "Operational Requirements Document (ORD) for Force XXI Battle Command Brigade-and-Below (FBCB2), Version 5.1 (Change 2)." Ft. Monmouth, NJ, 6 March 1998.
- U.S. Department of the Army. *The Armored and Mechanized Infantry Brigade*. FM 71-3, Washington, DC, January 1996a.
- U.S. Department of the Army. *Information Operations*. FM 100-6, Washington, DC, August 1996b.
- U.S. Department of the Army. *Tactics, Techniques, and Procedures for the Tactical Internet (Version 6)*. FM 24-32, Washington, DC, November 1999.
- von Neumann, J. *Theory of Self-Reproducing Automata*. Edited and completed by A. Banks, Urbana, IL: University of Illinois Press, 1966.
- Wolfram, S. "Twenty Problems in the Theory of Cellular Automata." *Physica Scripta*, vol. T9, pp. 170-183, 1985.
- Wuerz, MAJ R. Personal communication. U.S. Army Research Laboratory, Aberdeen Proving Ground, MD, March 2000.
- zum Brunnen, R. L., C. D. McDonald, P. R. Stay, M. W. Starks, and A. L. Barnes. "Information Operations Vulnerability/Survivability Assessment (IOVSA): Process Structure." pp. 19-21, ARL-TR-2250, U.S. Army Research Laboratory, Aberdeen Proving Ground, MD, June 2000.

Appendix A. Dynamics of Situational Awareness Message Servers

The dissemination of situational awareness (SA) messages throughout a digitized brigade involves the coordinated interaction of several different types of servers, including (a) carrier sense multiple access (CSMA) servers, (b) multisource group (MSG) servers, and (c) self servers (see section 1.2.1 in the main report for more details on the operational roles of these servers). In order to properly simulate the process of SA message dissemination throughout the brigade, the dynamics of both CSMA and MSG servers must be modeled to account for both message size and transmission channel baud rates. This includes both messages broadcast upward out of a particular net as well as other incoming messages which must be broadcast across a local net once received by a server. In all cases, the disseminated SA messages will represent K5.1 variable message format (VMF) messages, which report the positions of friendly platform nodes within the brigade network. Finally, an SA message is limited to a single bit in order to simulate server queue loading only (as opposed to measuring message network transit times).

Figure A-1 illustrates the dynamics of outbound SA message processing within a CSMA server. First, the server receives a message over its local single channel ground airborne radio system (SINCGARS) net (this message is also received by all other members of the net). Next, the server determines a course of action based on the total number of messages currently stored within its queue (this number can include a self-generated SA message in addition to received messages).

- If only one message is queued, the server will wait 4 s, transmit the message over its CSMA net via broadcast, and then reset an embedded timer which measures message waiting time within the queue.
 - If two messages are queued, the server will combine the messages into a CSMA net communication transmission unit (CTU) containing two bits and transmit the CTU immediately, and then reset its message timer.
 - If more than two messages are queued, the server will combine two messages into a CSMA net CTU and transmit the CTU immediately.
- (1) If there is now one message left in the queue, the server will wait four seconds, transmit the message, and then reset its message timer.
 - (2) If there are two or more messages left in the queue, the server will wait one second, and then recheck the number of queued messages (thus restarting the whole process).

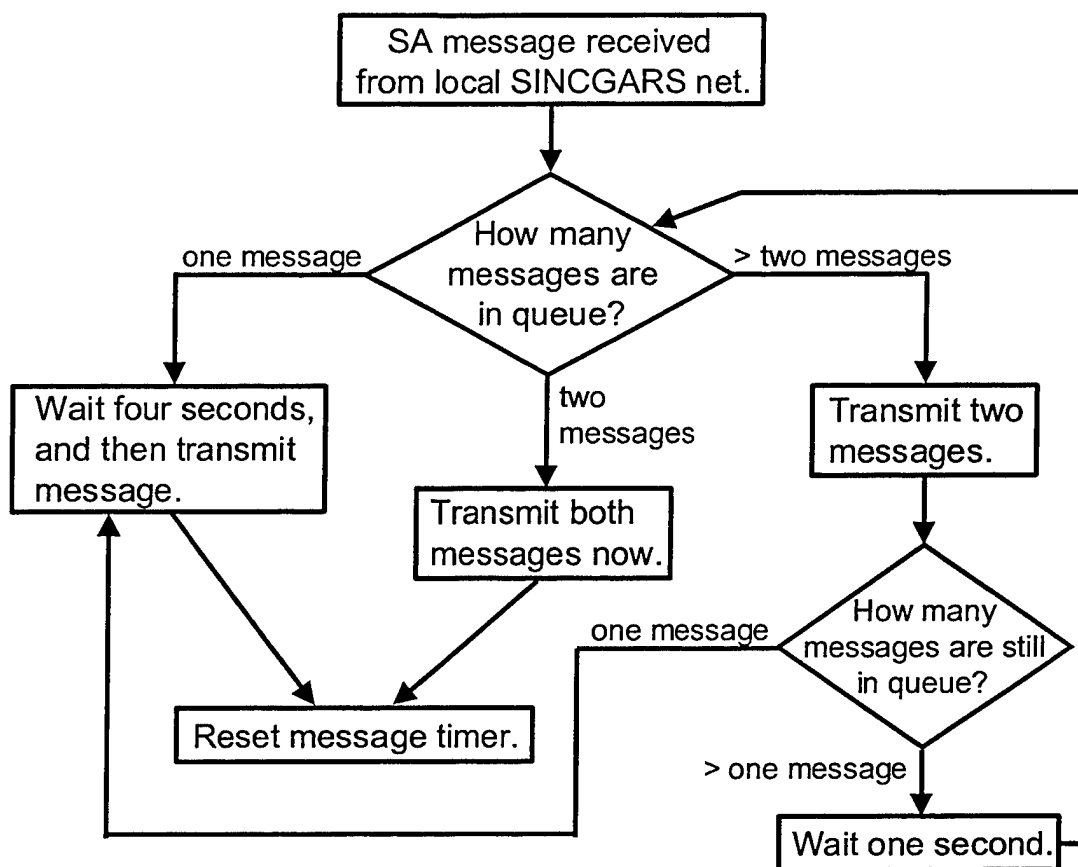


Figure A-1. Dynamics of outbound SA message processing within a CSMA server.

This process is restarted every time the CSMA server receives one or more new SA messages.

Figure A-2 illustrates the dynamics of SA message processing within an MSG server (which only sends messages outbound across the brigade-wide MSG net). First, the server receives a message over its CSMA net (this message is also received by all other enhanced position location reporting system (EPLRS)-equipped members of the net). As with the CSMA server described previously the MSG server determines a course of action based on the total number of currently queued messages.

- If one to four messages are queued, the server will wait 4 s, transmit the messages over the MSG net via broadcast, and then reset its message timer.
- If five messages are queued, the server will combine the messages into an MSG net CTU containing five bits, transmit the CTU immediately, and then reset its message timer.

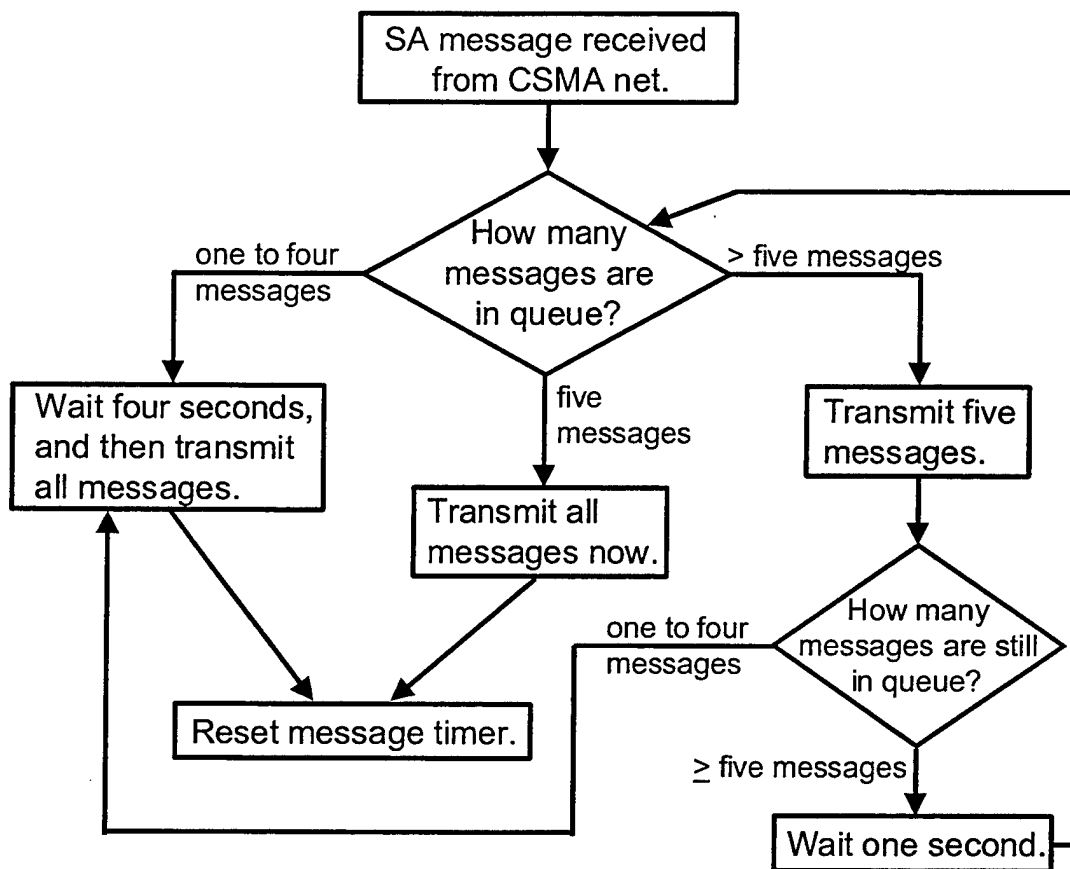


Figure A-2. Dynamics of outbound SA message processing within an MSG server.

- If more than five messages are queued, the server will combine five messages into a MSG net CTU and transmit the CTU immediately.
- (1) If there are now one to four messages left in the queue, the server will wait 4 s, transmit the messages, and then reset its message timer.
- (2) If there are five or more messages left in the queue, the server will wait 1 s, and then recheck the number of queued messages (thus restarting the whole process).

As with the CSMA server, this process is restarted every time the MSG server receives one or more new SA messages.

There is a simplifying assumption made within this MSG server model. Basically, it is assumed that an MSG net CTU can be transmitted within a 1-s time interval, which is the duration of a simulation time step using the mean field model (see section 3.1). In reality, given that (1) such a CTU is 600 bits in

length and (2) the transmission baud rate over the MSG net is 480 bits per second (bps), only 0.8 of a CTU can actually be transmitted during one simulation time step. Thus, the server queue saturation effects induced by GPS spoofing will actually evolve at a faster rate than predicted by the mean field model; i.e., staggered reduction in average SA connectivity levels; see Figure 12 in section 3.1 of the main report).

Lastly, Figure A-3 illustrates the dynamics of incoming SA message processing within a CSMA server. In this instance, messages are received over the server's CSMA net and the brigade-wide MSG net, and are subsequently rebroadcast over the server's local SINCGARS net. First, the server receives a message (which is also received by all other EPLRS-equipped members of the net over which the message was broadcast). Again, the server determines a course of action based on the total number of currently queued messages.

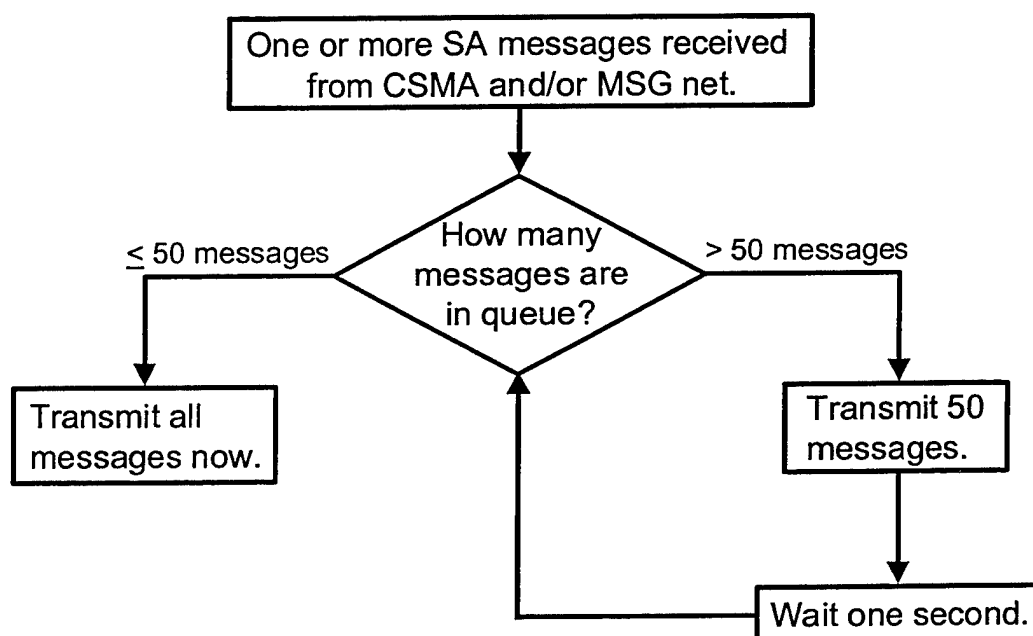


Figure A-3. Dynamics of incoming SA message processing within a CSMA server.

- If 1–50 messages are queued, the server will immediately transmit all of the messages over the local SINCGARS net via broadcast.
- If more than 50 messages are queued, the server will transmit 50 of the messages, wait 1 s, and then recheck the number of queued messages (thus restarting the whole process).

The 50 message/time step limit is due to (1) the data transmission baud rate associated with a SINCGARS net is assumed to be 16,000 bps and (2) the size of an SA message broadcast over a SINCGARS net is 320 bits. Thus, $16,000 \text{ bps} / 320 \text{ bits per message} = 50 \text{ messages transmitted per second}$. Of course, since SINCGARS net voice/data C2 messages are not considered in this model, the actual SA message transmission rate would likely be considerably lower than this estimate. Finally, this process is restarted every time the CSMA server receives one or more new incoming SA messages.

INTENTIONALLY LEFT BLANK.

**Appendix B. Cellang Source Code for the Situational
Awareness Network Model With Jammer
Bomb Stress Sources**

This appendix appears in its original form, without editorial change.

INTENTIONALLY LEFT BLANK.

```

# sa_net_jam --
# Situational awareness dissemination network with jammer bombs.
#
# This CA models a tactical digital communication network at the multiple
# BRIGADE level, where the network architecture is based on a generic
# Tactical Internet (TI) implementation. The model thus allows for data
# communication within a single tactical BRIGADE. The current version of
# this code models a spatial/geographical network architecture where all
# nodes move forward at the same fixed speed, so that network spatial
# configuration is essentially static (actually, nodes -- mounted on vehicles
# -- do not move forward, but rather invasive enemy "jammer" bombs move
# downward; see below). Cells represent different types of TI nodes within
# the architecture (i.e. clients and servers); these nodes represent
# networked Army systems at either the BRIGADE, BATTALION, COMPANY, or
# PLATOON military echelon levels.
#
# The model runs as a cyclic process with *FOUR* phases per simulation
# cycle (i.e. 1 four-phase cycle = 1 realtime increment):
#
# Phase 1: New or "advancing" jammer bombs move into empty cells.
#
# Phase 2: Perturb the state of a TI node cell from functional to
#           dysfunctional if a jammer bomb lies within a symmetric
#           jamming neighborhood relative to the cell.
#
# Phase 3: Determine the operational status of the cell's local and CSMA
#           servers (this status will be self-referential if the reference
#           cell is itself a local or CSMA server). The status report
#           includes only the current functionality of both servers. This
#           information will be used in Phase 4 in order to determine the
#           reference cell's tactical internet connectivity.
#
# Phase 4: Evaluate the operational fitness (fit/degraded/unfit) of all TI
#           node cells based on (i) self-function and (ii) neighbor cell
#           function within broadcast neighborhood (i.e. tactical internet
#           connectivity). Determine whether jammer bombs advance into
#           adjacent empty cells, cease to operate (due to battery drainage),
#           or are destroyed.
#
2 dimensions of
    # Output either
    #
    # (a) the SA connectivity (0 to 1000) which is then divided by 1000
    # within the data filter program, or
    #
    # (b) the SA communication fitness as either VERY HIGH (5),
    # HIGH (4), MEDIUM (3), LOW (2), or VERY LOW (1).
    #
    output of 0..1000    # NOTE: currently set up to output SA connectivity
    #output of 0..5

```

```

# The xpos and ypos fields record the absolute x and y positions,
# respectively, of a cell relative to the 2D CA lattice.
#
xpos of 0..128
ypos of 0..128

# 0 = Empty cell
# 1 = TI node system with SINCGARS only (no EPLRS)
# 2 = TI node system with SINCGARS AND EPLRS -- capable of
#       functioning as local stub net server
# 3 = Self server (i.e. SINCGARS/EPLRS-equipt system with
#       no other systems in local stub net; must act as its
#       own local stub net server).
# 4 = CSMA Primary SA server (pre-planned)
# 5 = CSMA Alternate SA server (pre-planned)
# 6 = Jammer bomb
# 7 = Empty cell which absorbs entering jammer bombs.
#
type of 0..7

# The echelon field identifies the military echelon of a TI node.
# Possible echelons are
#
# 4 --> BRIGADE-level (e.g. Brigade Commander)
# 3 --> BATTALION-level (e.g. Battalion TOC S3)
# 2 --> COMPANY-level (e.g. Company First Sargent)
# 1 --> PLATOON-level (e.g. Platoon Wing Man)
# 0 --> not a TI node (i.e. an empty cell)
#
echelon of 0..4

# fitness = 5 (VERY HIGH);
# fitness = 4 (HIGH);
# fitness = 3 (MEDIUM);
# fitness = 2 (LOW);
# fitness = 1 (VERY LOW).
#
fitness of 0..10

# The function field indicates the current RECEIVE functionality
# of the system hardware/software required for TI connectivity,
# where
#       1 = system is currently able to transmit AND receive
#           SA messages;
#       0 = system is currently able to transmit BUT unable to
#           receive SA messages.
# Thus, only message reception is adversely affected by a jammer
# bomb in the vicinity of a cell.
#
function of 0..1

# The server_local field indicates a server that is dynamically
# selected to function as a local stub net server. Any SINCGARS/
# EPLRS-equipt system may act as a local stub net server (i.e.
# cell types 2, 4, or 5 -- type 3 is already a server by
# definition and doesn't require this field). The local stub net
# server is a TI node that maintains and disseminates SA messages

```

```

# to and from other client nodes within a local SA routing
# neighborhood, and also broadcasts SA data generated within its
# local stub net outward across the associated CSMA net.
#
#         0 = non-server
#         1 = server.
#
server_local of 0..1

# The server_csma field indicates a pre-determined active CSMA
# net server; this function may be assumed by either the primary or
# alternate CSMA servers (cell types 4 and 5, respectively). The
# CSMA server receives SA data from its local CSMA net and then
# re-broadcasts this data across the BRIGADE-wide MSG net.
#
#         0 = non-server
#         1 = server.
#
# Since CSMA servers across different BATTALIONS (including the
# BRIGADE-AREA unit; see below) must exchange "broadcast shares"
# between themselves for the privilege of broadcasting, the token
# field is used to identify the CSMA server which currently holds
# the authority to broadcast.
#
server_csma of 0..1

# The bn field identifies the BATTALION or CSMA community to
# which a TI node belongs. A BRIGADE-AREA unit, which maintains
# its own CSMA SA net, is undistinguishable from a subordinate
# BATTALION within the context of this model.
#
bn of 0..10

# The stub field identifies the local STUB NET to which a TI node
# belongs. A local stub net is itself contained within an BATTALION,
# which is in turn contained within a BRIGADE. Thus, the 5th STUB NET
# within the 3rd BATTALION is identified by the cell fields
#
#         cell.stub = 5
#         cell.bn = 3.
#
stub of 0..50

# The target field designates an empty cell (type = 0) which can either
# (a) become occupied by a previously-positioned jammer bomb (with
# probability P1) or (b) serve as a target drop point for a jammer bomb
# dropped by an over-flying aircraft (with probability P2). In these
# cases, cell.target = 1 or cell.target = 2, respectively.
#
target of 0..2

# The jammer_timer field keeps track of the lifetime of a jammer bomb.
# This field is passed as an agent to represent a bomb moving downward
# relative to the BRIGADE structure.
#
jammer_timer of 0..300

```

```

# The connect field represents a fraction which measures, for a
# reference cell, the number of currently functional cells which can
# potentially provide INCOMING SA data to the reference cell divided
# by the total number of such cells (including both functional and
# dysfunctional cells). Since this code is limited to integer opera-
# tions, the above fraction is represented as the integer
# connect = (fraction * 1000)modulo 0 (i.e. rounded down to next
# integer), where 0 <= connect <= 1000.
#
connect of 0..1000

# The server_status field encodes the current operational status of a
# cell's local stub net server and/or CSMA server (if the cell itself
# is a local/CSMA server, then the report status is self-referential).
# This field is set/reset for each TI node cell during phase 2 of a
# simulation cycle and then queried during phase 3 of the same cycle
# in order to calculate a cell's connectivity measure. The array
# elements of the server_status field are
#
# cell.server_status[0]: current local server function (0/1);
#
# cell.server_status[1]: current CSMA server function (0/1).
#
server_status[] for 2 of 0..1

agent of
# The bomb and timer fields indicates a message sent from a bomb-
# occupied cell to an adjacent empty cell into which the bomb will
# "move." The bomb agent field indicates the presence of a jammer
# bomb; the timer agent field contains the same information as
# cell.jammer_timer.
#
bomb of 0..1
timer of 0..300

# The transfer_local field indicates a message sent from a
# dysfunctional local stub net server to a dynamically-selected
# alternate server within the local stub net. A field value of "1"
# instructs the alternate server to assume the role of active stub net
# server.
#
transfer_local of 0..1

# The transfer_csma field indicates a message sent from a dysfunctional
# primary CSMA server to a pre-determined alternate/standby server
# within the same BATTALION/CSMA community. A field value
# of "1" instructs the alternate server to assume the role of active
# CSMA server. A subsequent message with a field value of "0"
# (sent from the primary to alternate server upon functional recovery
# of the primary) instructs the alternate server to relinquish its
# server role.
#
transfer_csma of 0..1

end

```

```

# Set the maximum neighborhood search ranges to cover the entire cell
# universe based on the pre-set constants x_range and y_range. This takes
# advantage of the toroidal curvature of the cell universe.
#
const imin := -47      # 1 - (x range of automata/2)
const imax := 48       # x range of automata / 2
const jmin := -63      # 1 - (y range of automata/2)
const jmax := 64       # y range of automata / 2

# Set up generic BATTALION-wide neighborhood search ranges to cover 1/4th of
# the entire cell universe.
#
const ilow := -24      # (imin/2) - 1
const ihigh := 25      # (imax/2) + 1
const jlow := -32      # (jmin/2) - 1
const jhigh := 33      # (jmax/2) + 1

# Operational fitness thresholds:
#
#   fitness > high --> VERY HIGH
#   fitness <= high & fitness > medium --> HIGH
#   fitness <= medium & fitness > low --> MEDIUM
#   fitness <= low & fitness > very_low --> LOW
#   fitness <= very_low --> VERY LOW
#
#
const high := 900
const medium := 600
const low := 400
const very_low := 100

# Apply a geographic distance filter to all TI node cells within the
# BRIGADE.
#
const bde_filter := 80
const bn_filter := 60
const co_filter := 40
const plt_filter := 20

# Set the lifetime of a jammer bomb (all bombs are assumed to have the same
# lifetime).
#
const lifetime := 100

# Define a symmetric jamming neighborhood relative to a center cell. By
# symmetry arguments, the center cell may either be a jamming source (where
# neighbors are jammed targets) or a jammed target (with at least one
# neighbor as a jamming source).
#
jamming[] for 36 := [0, 1], [1, 1], [1, 0], [1, -1], [0, -1], [-1, -1],
                  [-1, 0], [-1, 1], [0, 2], [1, 2], [2, 2], [2, 1], [2, 0],
                  [2, -1], [2, -2], [1, -2], [0, -2], [-1, -2], [-2, -2],
                  [-2, -1], [-2, 0], [-2, 1], [-2, 2], [-1, 2], [0, 3],
                  [1, 3], [3, 1], [3, 0], [3, -1], [1, -3], [0, -3],
                  [-1, -3], [-3, -1], [-3, 0], [-3, 1], [-1, 3]

if time%4 = 0 & cell.type = 0 then

```

```

# Cell is a potential site for a jammer bomb and
# time = 0, 4, 8, 12, 16, ...
#
if cell.target = 1 then
    # A potential pre-planted jammer bomb.
    #
    P_planted := random%1000
    if P_planted < 67 then
        # There is a one-in-fifteen chance that a targeted
        # cell will contain a pre-planted jammer bomb with
        # a remaining lifetime of 70 simulation cycles (i.e.
        # 280 timesteps).
        #
        cell.type := 6
        cell.jammer_timer := 30
    elseif P_planted >= 67 & P_planted < 133 then
        # There is a one-in-fifteen chance that a targeted
        # cell will contain a pre-planted jammer bomb with
        # a remaining lifetime of 60 simulation cycles (i.e.
        # 240 timesteps).
        #
        cell.type := 6
        cell.jammer_timer := 40
    elseif P_planted >= 133 & P_planted < 200 then
        # There is a one-in-fifteen chance that a targeted
        # cell will contain a pre-planted jammer bomb with
        # a remaining lifetime of 50 simulation cycles (i.e.
        # 200 timesteps).
        #
        cell.type := 6
        cell.jammer_timer := 50
    end
elseif cell.target = 2 then
    # A potential air-dropped jammer bomb.
    #
    P_dropped := random%1000
    if P_dropped < 5 then
        # There is a one-in-two-hundred chance that a targeted
        # cell will receive an air-dropped jammer bomb.
        cell.type := 6
        cell.jammer_timer := 0
    end
end

type := cell.type

# Check for a jammer bomb moving into the cell from a previously-
# occupied ground position.
#
new_bomb_timer := cell.jammer_timer
forall bomb:agent
    # Move a bomb into the cell if appropriate.
    #
    if bomb.bomb then
        if type = 0 then
            cell.type := 6
            cell.jammer_timer := bomb.timer + 1
        end
    end
end

```

```

        elsif type = 6 then
            cell.jammer_timer := bomb.timer + 1
            when bomb.timer < new_bomb_timer
        end
    end
end

elsif time%4 = 0 & cell.type = 6 then
    # Move a jammer bomb out of its previous location.
    #
    cell.type := 0

elsif time%4 = 1 & cell.type > 0 & cell.type < 6 then
    # Cell is a tactical internet network node and
    # time = 1, 5, 9, 13, 17, ...
    #
    # Jam the cell's radios (cutting off tactical internet data reception)
    # if there is a jammer bomb present within a jamming neighborhood.
    #
    function := 1
    forall i
        if jamming[i].type = 6 then
            function := 0
            exit
        end
    end
    cell.function := function

elsif time%4 = 2 & cell.type > 0 & cell.type < 6 then
    # Cell is a TI node and time = 2, 6, 10, 14, 18, ...
    #
    cell_bn := cell.bn
    cell_stub := cell.stub
    type := cell.type

    # Establish the current status of the cell's local stub net and CSMA
    # servers (this information is self-referential if the cell is itself
    # either an active local or CSMA server).
    #
    forall i:ilow..ihigh
        forall j:jlow..jhigh
            neighbor := [i, j]
            neighbor_function := neighbor.function
            neighbor_bn := neighbor.bn
            neighbor_stub := neighbor.stub
            server_local := neighbor.server_local
            server_csma := neighbor.server_csma
            if neighbor_bn = cell_bn & neighbor_stub =
                cell_stub & server_local then
                # Neighbor is the cell's local server.
                server_status_0 := neighbor_function
            elsif neighbor_bn = cell_bn & server_csma then
                # Neighbor is the cell's CSMA server.
                server_status_1 := neighbor_function
            end
        end
    end
end
end

```

```

if cell.type = 2 & cell.server_local &
    (cell.function = 0 | cell.function = 2) then
    # Transfer the local stub net server role to the
    # first available/functional EPLRS-equip system (i.e.
    # a type 2 cell) found in a search of the stub net/
    # local SA community.
    #
    cell.server_local := 0
    if cell.echelon = 1 then
        agent(0, 0, 1, 0) -> [0, 1] when [0, 1].type = 2 &
            [0, 1].echelon = 1 &
            [0, 1].function
        -> [1, 0] when [1, 0].type = 2 &
            [1, 0].echelon = 1 &
            [1, 0].function
        -> [0, -1] when [0, -1].type = 2 &
            [0, -1].echelon = 1 &
            [0, -1].function
        -> [-1, 0] when [-1, 0].type = 2 &
            [-1, 0].echelon = 1 &
            [-1, 0].function
        -> cell otherwise
    elseif cell.echelon = 2 then
        agent(0, 0, 1, 0) -> [0, 2] when [0, 2].type = 2 &
            [0, 2].echelon = 2 &
            [0, 2].function
        -> [2, 0] when [2, 0].type = 2 &
            [2, 0].echelon = 2 &
            [2, 0].function
        -> [0, -2] when [0, -2].type = 2 &
            [0, -2].echelon = 2 &
            [0, -2].function
        -> [-2, 0] when [-2, 0].type = 2 &
            [-2, 0].echelon = 2 &
            [-2, 0].function
        -> cell otherwise
    elseif cell.echelon = 3 then
        agent(0, 0, 1, 0) -> [0, 1] when [0, 1].type = 2 &
            [0, 1].echelon = 3 &
            [0, 1].function
        -> [1, 1] when [1, 1].type = 2 &
            [1, 1].echelon = 3 &
            [1, 1].function
        -> [1, 0] when [1, 0].type = 2 &
            [1, 0].echelon = 3 &
            [1, 0].function
        -> [1, -1] when [1, -1].type = 2 &
            [1, -1].echelon = 3 &
            [1, -1].function
        -> [0, -1] when [0, -1].type = 2 &
            [0, -1].echelon = 3 &
            [0, -1].function
        -> [-1, -1] when [-1, -1].type = 2 &
            [-1, -1].echelon = 3 &
            [-1, -1].function
        -> [-1, 0] when [-1, 0].type = 2 &

```

```

        [-1, 0].echelon = 3 &
        [-1, 0].function
    -> [-1, 1] when [-1, 1].type = 2 &
        [-1, 1].echelon = 3 &
        [-1, 1].function
    -> cell otherwise
elseif cell.echelon = 4 then
    agent(0, 0, 1, 0) -> [0, 1] when [0, 1].type = 2 &
        [0, 1].echelon = 4 &
        [0, 1].function
    -> [1, 1] when [1, 1].type = 2 &
        [1, 1].echelon = 4 &
        [1, 1].function
    -> [1, 0] when [1, 0].type = 2 &
        [1, 0].echelon = 4 &
        [1, 0].function
    -> [1, -1] when [1, -1].type = 2 &
        [1, -1].echelon = 4 &
        [1, -1].function
    -> [0, -1] when [0, -1].type = 2 &
        [0, -1].echelon = 4 &
        [0, -1].function
    -> [-1, -1] when [-1, -1].type = 2 &
        [-1, -1].echelon = 4 &
        [-1, -1].function
    -> [-1, 0] when [-1, 0].type = 2 &
        [-1, 0].echelon = 4 &
        [-1, 0].function
    -> [-1, 1] when [-1, 1].type = 2 &
        [-1, 1].echelon = 4 &
        [-1, 1].function
    -> cell otherwise
end
elseif (cell.type = 4 | cell.type = 5) & cell.server_csma &
    (cell.function = 0 | cell.function = 2) then
    # Transfer the CSMA server role to the standby CSMA
    # server if it is currently functional.
    #
    cell.server_csma := 0
    if cell.echelon = 2 then
        agent(0, 0, 0, 1) -> [2, 0] when ([2, 0].type = 4 |
            [2, 0].type = 5) &
            [2, 0].echelon = 2 &
            [2, 0].function
        -> [-2, 0] when ([-2, 0].type = 4 |
            [-2, 0].type = 5) &
            [-2, 0].echelon = 2 &
            [-2, 0].function
        -> cell otherwise
    elseif cell.echelon = 3 then
        agent(0, 0, 0, 1) -> [1, 0] when ([1, 0].type = 4 |
            [1, 0].type = 5) &
            [1, 0].echelon = 3 &
            [1, 0].function
        -> [-1, 0] when ([-1, 0].type = 4 |
            [-1, 0].type = 5) &
            [-1, 0].echelon = 3 &

```

```

                                [-1, 0].function
                                -> cell otherwise
elseif cell.echelon = 4 then
    agent(0, 0, 0, 1) -> [1, 0] when ([1, 0].type = 4 |
                                [1, 0].type = 5) &
                                [1, 0].echelon = 4 &
                                [1, 0].function
                                -> [-1, 0] when ([-1, 0].type = 4 |
                                [-1, 0].type = 5) &
                                [-1, 0].echelon = 4 &
                                [-1, 0].function
                                -> cell otherwise
end
end

cell.server_status[0] := server_status_0
cell.server_status[1] := server_status_1

elseif time%4 = 3 & cell.type > 0 & cell.type < 6 then
    # Cell is a TI node and time = 3, 7, 11, 15, 19, ...
    #
    cell_xpos := cell.xpos
    cell_ypos := cell.ypos
    cell_function := cell.function
    type := cell.type
    bn := cell.bn
    stub := cell.stub
    server_local := cell.server_local
    server_csma := cell.server_csma
    cell_local_function := cell.server_status[0]
    cell_csma_function := cell.server_status[1]

    # Calculate a cell's inbound connectivity to the tactical internet
    # (i.e. examine the cell as a RECEIVER of SA messages) and the
    # correlated SA fitness.
    #
    echelon := cell.echelon
    number_total := 0
    number_function := 0
    forall i:imin..imax forall j:jmin..jmax
        neighbor := [i, j]
        xpos := neighbor.xpos
        ypos := neighbor.ypos
        neighbor_local_function := neighbor.server_status[0]
        neighbor_csma_function := neighbor.server_status[1]
        delta_x_1 := cell_xpos - xpos
        delta_x_2 := xpos - cell_xpos
        delta_y_1 := cell_ypos - ypos
        delta_y_2 := ypos - cell_ypos
        delta_x := delta_x_1 when delta_x_1 > 0
                := delta_x_2 when delta_x_2 > 0
                := 0 otherwise
        delta_y := delta_y_1 when delta_y_1 > 0
                := delta_y_2 when delta_y_2 > 0
                := 0 otherwise
        if echelon = 1 then
            within := 1 when delta_x < plt_filter

```

```

        & delta_y < plt_filter
        := 0 otherwise
    elsif echelon = 2 then
        within := 1 when delta_x < co_filter
        & delta_y < co_filter
        := 0 otherwise
    elsif echelon = 3 then
        within := 1 when delta_x < bn_filter
        & delta_y < bn_filter
        := 0 otherwise
    elsif echelon = 4 then
        within := 1 when delta_x < bde_filter
        & delta_y < bde_filter
        := 0 otherwise
    else
        within := 0
    end

    number_total := number_total + 1 when within & (i != 0 | j != 0)
    & neighbor.type > 0 & neighbor.type < 6

    if within then
        if type = 1 then
            # Cell is a SINGARS-only-equip member of a stub net.
            #
            if neighbor.type = 1 then
                # Neighbor only has a SINGARS radio on board (no \
                # EPLRS).
                #
                if neighbor.bn = bn & neighbor.stub = stub then
                    # Neighbor is in the same local stub as the
                    # reference cell.
                    #
                    number_function := number_function + 1
                    when (i != 0 | j != 0)

                elsif neighbor.bn = bn & neighbor.stub != stub then
                    # Neighbor is in the same CSMA net as the reference
                    # cell, but in a different local stub net. Thus, the
                    # communication channel between the neighbor and cell
                    # also depends on functionality of both the
                    # neighbor's and cell's local servers.
                    #
                    number_function := number_function + 1
                    when (i != 0 | j != 0) &
                    cell_local_function
                    & neighbor_local_function

                else
                    # Neighbor and reference cell are in different CSMA
                    # nets. Thus, the communication channel between the
                    # neighbor and cell also depends on functionality of
                    # (i) the neighbor's local server, (ii) the
                    # neighbor's CSMA server, and (iii) the cell's local
                    # server.
                    #
                    number_function := number_function + 1
                    when (i != 0 | j != 0) &
                    cell_local_function &

```

```

neighbor_local_function &
neighbor_csma_function

end

elsif neighbor.type = 2 | neighbor.type = 3 |
neighbor.type = 4 | neighbor.type = 5 then
# Neighbor is equipt with an EPLRS data radio.
#
if neighbor.bn = bn & neighbor.stub = stub &
neighbor.type != 3 then
# Neighbor is in the same local stub as the
# reference cell.
#
number_function := number_function + 1
when (i != 0 | j != 0)

elsif neighbor.bn = bn & neighbor.stub != stub then
# Neighbor is in the same CSMA net as the reference
# cell, but in a different local stub net.
#
if (neighbor.type = 2 & !neighbor.server_local) |
neighbor.type = 4 | neighbor.type = 5 then
# The communication channel between the
# neighbor and cell depends on
# functionality of the cell's local
# server and the neighbor's local server.
#
number_function := number_function + 1
when (i != 0 | j != 0) &
cell_local_function
& neighbor_local_function
elsif (neighbor.type = 2 & neighbor.server_local) |
neighbor.type = 3 then
# The communication channel between the neighbor and
# cell depends on functionality of the cell's local
# server.
#
number_function := number_function + 1
when (i != 0 | j != 0) &
cell_local_function
end

else
# Neighbor and reference cell are in different CSMA nets.
#
if (neighbor.type = 2 & !neighbor.server_local) |
neighbor.type = 4 | neighbor.type = 5 then
# The communication channel between the neighbor and
# cell depends on functionality of (i) the cell's
# local server, (ii) the neighbor's local server, and
# (iii) the neighbor's CSMA server.
#
number_function := number_function + 1
when (i != 0 | j != 0) &
cell_local_function
& neighbor_local_function &
neighbor_csma_function

```

```

        elsif (neighbor.type = 2 & neighbor.server_local) |
            neighbor.type = 3 then
            # The communication channel between the neighbor and
            # cell depends on functionality of the cell's local
            # server and neighbor's CSMA server.
            #
            number_function := number_function + 1
                                when (i != 0 | j != 0) &
                                cell_local_function
                                & neighbor_csma_function
        end
    end
end

elsif type = 2 | type = 3 | type = 4 | type = 5 then
    # Cell is any EPLRS-equipt TI node.
    #
    if neighbor.type = 1 then
        # Neighbor only has a SINCGARS radio on board (no EPLRS).
        #
        if neighbor.bn = bn & neighbor.stub = stub then
            # Neighbor is in the same local stub as the reference
            # cell.
            #
            number_function := number_function + 1
                                when (i != 0 | j != 0)

            elsif neighbor.bn = bn & neighbor.stub != stub then
                # Neighbor is in the same CSMA net as the reference
                # cell, but in a different local stub net. Thus, the
                # communication channel between the neighbor and cell
                # also depends on functionality of the neighbor's
                # local server.
                #
                number_function := number_function + 1
                                    when (i != 0 | j != 0) &
                                    neighbor_local_function
            else
                # Neighbor and reference cell are in different CSMA
                # nets. Thus, the communication channel between the
                # neighbor and cell also depends on functionality of
                # (i) the neighbor's local server and (ii) the
                # neighbor's CSMA server.
                #
                number_function := number_function + 1
                                    when (i != 0 | j != 0) &
                                    neighbor_local_function
                                    & neighbor_csma_function
            end
        end

    elsif neighbor.type = 2 | neighbor.type = 3 | neighbor.type
        = 4 | neighbor.type = 5 then
        # Neighbor is equipt with an EPLRS data radio.
        #
        if neighbor.bn = bn & neighbor.stub = stub then
            # Neighbor is in the same local stub as the reference
            # cell.

```

```

#
    number_function := number_function + 1
                        when (i != 0 | j != 0)

elseif neighbor.bn = bn & neighbor.stub != stub then
# Neighbor is in the same CSMA net as the reference
# cell, but in a different local stub net.
#
if (neighbor.type = 2 & !neighbor.server_local) |
    neighbor.type = 4 | neighbor.type = 5 then
# The communication channel between the neighbor and
# cell depends on functionality of the neighbor's
# local server.
#
    number_function := number_function + 1
                        when (i != 0 | j != 0) &
                            neighbor_local_function

elseif (neighbor.type = 2 & neighbor.server_local) |
    neighbor.type = 3 then
# The communication channel between the neighbor and
# cell depends only on cell functionality.
#
    number_function := number_function + 1
                        when (i != 0 | j != 0)
end

else
# Neighbor and reference cell are in different CSMA
# nets.
#
if (neighbor.type = 2 & !neighbor.server_local) |
    neighbor.type = 4 | neighbor.type = 5 then
# The communication channel between the neighbor and
# cell depends on functionality of (i) the neighbor's
# local server and (ii) the neighbor's CSMA server.
#
    number_function := number_function + 1
                        when (i != 0 | j != 0) &
                            neighbor_local_function &
                                neighbor_csma_function

elseif (neighbor.type = 2 & neighbor.server_local) |
    neighbor.type = 3 then
# The communication channel between the neighbor and
# cell depends on functionality of the neighbor's
# CSMA server.
#
    number_function := number_function + 1
                        when (i != 0 | j != 0) &
                            neighbor_csma_function
end

end

end
end
end
end

```

```

connect := (number_function*1000) / number_total when
           cell_function
           := 0 otherwise

fitness := 5 when connect > high
          := 4 when connect <= high &
                connect > medium
          := 3 when connect <= medium &
                connect > low
          := 2 when connect <= low &
                connect > very_low
          := 1 when connect <= very_low
          := 0 otherwise

# Re-assign server roles as appropriate. This is done at the end of
# the phase 3 timestep so as not to effect current cell message
# reception fitness metrics. Thus, the actual role transfer is
# effectively delayed until the next simulation cycle (i.e. the next
# time step).
#
local_transfer := 0
csma_transfer := 0

forall message:agent
    # Acknowledge receipt of server role transfer authorization
    # messages. The actual role transfer is delayed until the
    # next simulation cycle (i.e. the next timestep).
    local_transfer := 1 when type = 2 & message.transfer_local = 1
    csma_transfer := 1 when (type = 4 | type = 5) &
        message.transfer_csma = 1
end

# Assign server roles to designated cells based on reception of
# transfer authorization messages. The receiver of the server
# authorization message SHOULD be currently functional.
#
server_local := 1 when type = 2 & local_transfer
server_csma := 1 when (type = 4 | type = 5) & csma_transfer

cell.server_local := server_local
cell.server_csma := server_csma
cell.connect := connect
cell.fitness := fitness

cell.output := connect
# cell.output := fitness when cell.type > 0 & cell.type < 6
# := 0 otherwise

elsif time%4 = 3 & cell.type = 6 then
    # Cell is a jammer bomb and time = 3, 7, 11, 15, 19, ...
    #
    jammer_timer := cell.jammer_timer
    new_position := [0, -1]
    if jammer_timer < lifetime & new_position.type = 0 then
        # Send a message to move the jammer bomb downward
        # one lattice cell.

```

```
        #
        agent(1, jammer_timer, 0, 0) -> [0, -1]
    end
end
```

Appendix C. Data Filter Source Code

This appendix appears in its original form, without editorial change.

INTENTIONALLY LEFT BLANK.

```

/* filter_jam.c

This data filtering program should ONLY be used with the Cellang code
modeling the various jammer bomb IO scenarios.

*/

#include <stdio.h>

/***** Model Constants *****/

/* Is the number of time steps in each complete phase of the computation. */
#define TIME_PHASES      4

/* The total maximum number of tactical internet nodes in the simulation. */
#define MAX_NODES        612

/* Determines the consecutive number of complete time phase computations to
keep.
*/
#define TIME_PERIOD      1

/* The total number of tactical internet echelon levels in the simulation. */
#define NUM_OF_ECHELONS  4

/* The total number of possible nodal fitness states in the simulation. */
#define NUM_OF_NODE_STATES 5

/* Various constants for dealing with different sampling windows/partitions.
*/
#define NUM_OF_BN_NETS    7
#define NUM_OF_STUB_NETS  26

/* This is the maximum of NUM_OF_BN_NETS and NUM_OF_STUB_NETS. */
#define MAX_NUM_OF_SUBNETS 26

/***** Data Structures *****/

typedef struct {
    int vhigh_nodes[TIME_PERIOD];
    int high_nodes[TIME_PERIOD];
    int med_nodes[TIME_PERIOD];
    int low_nodes[TIME_PERIOD];
    int vlow_nodes[TIME_PERIOD];
} echelon;

static echelon fitness_count[NUM_OF_ECHELONS];

/* This structure maintains the total amount of time that each node has thus
far spent in a particular state.
*/
typedef struct {
    int vhigh_time, high_time, med_time, low_time, vlow_time;

```

```

} node_fitness_times;

static node_fitness_times node_times[MAX_NODES];

/* This structure maintains the current fitness state of each node and the
   consecutive time to present that the node has been in that state.  E.g.

       time_in_state[3].state == 2 and time_in_state[3].time == 5

   means that the 4th node has been in state 2 for the past 5 time periods.
*/
typedef struct {
    int state;
    int time;
} node_time_in_state;

static node_time_in_state time_in_state[MAX_NODES];

/* This structure maintains the state that each node was in during the most
   recent TIME_PERIOD previous time steps/phases.  The zeroth (0th) entry
   is the most recent (i.e. current) state.
*/
static int last_states[MAX_NODES][TIME_PERIOD];

/* This structure indicates the position within the CA universe of each
   node as well as its echelon level, the battalion to which it belongs, and
   the stub net it is a part of.
*/
typedef struct {
    int x, y;
    int echelon;
    int bn_net;
    int stub_net;
} info_for_node;

static info_for_node node_info[MAX_NODES];

/* Counts the number of nodes belonging to each echelon level.  Where

       echelon_count[3] = # Brigade nodes
                       2  = # Battalion nodes
                       1  = # Company nodes
                       0  = # Platoon nodes
*/
static int echelon_count[NUM_OF_ECHELONS];

/***** Data Gathering *****/

extern int dim_size(int dim);
extern long int get_field (int field, int dim_0_index, int dim_1_index);
extern unsigned long int get_time (void);

static unsigned long int time;

static int x_dim_size, y_dim_size;
static int number_of_nodes = 0;

```

```

#define OUTPUT_FIELD    0
#define TYPE_FIELD      3
#define ECHELON_FIELD   4
#define BN_FIELD        11
#define STUB_FIELD      12

/* Input array files that the CA takes the x and y coordinates for the
   nodes from.
*/
#define X_NODE_COORD    "nodes_x"
#define Y_NODE_COORD    "nodes_y"

/* Initializes the data structures in preparation for getting the nodal
   data at the end of each time phase.  0 is returned if operations are
   okay and should continue, 1 is returned if there is a problem.
*/
static int init_data(void) {
    int i, j;
    FILE *x_coord_file, *y_coord_file;

    /* Initialization at time == 0 */
    x_dim_size = dim_size(0);
    y_dim_size = dim_size(1);

    memset(echelon_count, 0, NUM_OF_ECHELONS*sizeof(int));
    memset(last_states, 0, MAX_NODES*TIME_PERIOD*sizeof(int));

    x_coord_file = fopen(X_NODE_COORD, "r");
    y_coord_file = fopen(Y_NODE_COORD, "r");

    /* Make sure the files got opened. */
    if (x_coord_file == NULL) {
        fprintf(stderr, "FILTER ERROR: Unable to open the file '%s'\n",
            X_NODE_COORD);
        return 1;
    }
    if (y_coord_file == NULL) {
        fprintf(stderr, "FILTER ERROR: Unable to open the file '%s'\n",
            Y_NODE_COORD);
        return 1;
    }

    /* Get position and echelon values for each node. */
    while (1) {
        int x, y, echelon;
        int x_scan, y_scan;

        x_scan = fscanf(x_coord_file, " %d", &x);
        y_scan = fscanf(y_coord_file, " %d", &y);

        if (x_scan <= 0 || y_scan <= 0) break;
        else if (x_scan + y_scan == 1) {
            /* A different number of coordinates was
               placed in each file.
            */
            fprintf(stderr, "FILTER ERROR: Number of value"

```

```

        " in coordinate files '" X_NODE_COORD
        "' and '" Y_NODE_COORD "' are not the "
        "same.\n");
    return 1;
} else if (x < 0 || x > dim_size(0)) {
    fprintf(stderr, "FILTER ERROR: The value of the x"
        " coordinate from '" X_NODE_COORD "' at"
        " position %d is out of bounds.\n",
        number_of_nodes+1);
    return 1;
} else if (y < 0 || y > dim_size(1)) {
    fprintf(stderr, "FILTER ERROR: The value of the y"
        " coordinate from '" Y_NODE_COORD "' at"
        " position %d is out of bounds.\n",
        number_of_nodes+1);
    return 1;
}

if (number_of_nodes == MAX_NODES) {
    fprintf(stderr, "FILTER ERROR: Increase the size of"
        " MAX_NODES in filter.c and recompile.\n");
    return 1;
}

node_info[number_of_nodes].x = x;
node_info[number_of_nodes].y = y;
echelon = get_field (ECHELON_FIELD, x, y);

/* Check for a legal echelon value. */
if (echelon < 1 || echelon > NUM_OF_ECHELONS) {
    fprintf(stderr, "FILTER ERROR: Echelon value (%d) for "
        " node at (%d, %d) is out of range.\n",
        echelon, x, y);
    return 1;
}

node_info[number_of_nodes].echelon = echelon-1;

echelon_count[echelon-1]++;

node_times[number_of_nodes].vhigh_time
    = node_times[number_of_nodes].high_time
    = node_times[number_of_nodes].med_time
    = node_times[number_of_nodes].low_time
    = node_times[number_of_nodes].vlow_time = 0;

node_info[number_of_nodes].bn_net = get_field (BN_FIELD, x, y);

node_info[number_of_nodes].stub_net =
    get_field (STUB_FIELD, x, y);

/* A node cell should NEVER be in state 0, thus
   whatever state it is in will not match and the
   data gathering loop below will set the correct
   initial value.
*/
time_in_state[number_of_nodes].state = 0;

```

```

        time_in_state[number_of_nodes].time = 0;

        number_of_nodes++;
    }
    fclose(x_coord_file);
    fclose(y_coord_file);

    /* Zero out all elements of the echelon time phase counts. */
    for (i = 0; i < NUM_OF_ECHELONS; i++)
        for (j = 0; j < TIME_PERIOD; j++) {
            fitness_count[i].vhigh_nodes[j] = 0;
            fitness_count[i].high_nodes[j] = 0;
            fitness_count[i].med_nodes[j] = 0;
            fitness_count[i].low_nodes[j] = 0;
            fitness_count[i].vlow_nodes[j] = 0;
        }

    return 0;
}

/* Gets the data at the end of each complete time phase. 0 is returned if
operations are okay and should continue, 1 is returned if there is a
problem.
*/
static int get_data(void) {
    int i, j;

    /* Fitness counts for each echelon during this time step. */
    int vhigh_count[NUM_OF_ECHELONS],
        high_count[NUM_OF_ECHELONS],
        med_count[NUM_OF_ECHELONS],
        low_count[NUM_OF_ECHELONS],
        vlow_count[NUM_OF_ECHELONS];

    /* Initialize echelon fitness counts. */
    memset(vhigh_count, 0, NUM_OF_ECHELONS*sizeof(int));
    memset(high_count, 0, NUM_OF_ECHELONS*sizeof(int));
    memset(med_count, 0, NUM_OF_ECHELONS*sizeof(int));
    memset(low_count, 0, NUM_OF_ECHELONS*sizeof(int));
    memset(vlow_count, 0, NUM_OF_ECHELONS*sizeof(int));

    /* Get the data for each node. */
    for (i = 0; i < number_of_nodes; i++) {
        int x = node_info[i].x;
        int y = node_info[i].y;
        int echelon = node_info[i].echelon;

        int fitness = (int) get_field (OUTPUT_FIELD, x, y);
        /* This works ONLY when the cell.output field (i.e.,
        field 0) in the Cellang code mirrors the cell.fitness
        field.
        */
        switch (fitness) {
            case 5: /* VERY HIGH */
                node_times[i].vhigh_time++;
                vhigh_count[echelon]++;
                break;

```

```

        case 4: /* HIGH */
            node_times[i].high_time++;
            high_count[echelon]++;
        break;
        case 3: /* MEDIUM */
            node_times[i].med_time++;
            med_count[echelon]++;
        break;
        case 2: /* LOW */
            node_times[i].low_time++;
            low_count[echelon]++;
        break;
        case 1: /* VERY LOW */
            node_times[i].vlow_time++;
            vlow_count[echelon]++;
        break;
    }

    /* Shift the remembered states and add the latest one. */
    for (j = TIME_PERIOD-1; j > 0; j--)
        last_states[i][j] = last_states[i][j-1];
    last_states[i][0] = time_in_state[i].state;

    if (time_in_state[i].state == fitness)
        time_in_state[i].time++;
    else {
        time_in_state[i].state = fitness;
        time_in_state[i].time = 1;
    }
}

/* Shift the fitness counts for the previous time steps and then
   include the count for the current time step.
*/
for (j = 0; j < NUM_OF_ECHELONS; j++) {
    for (i = TIME_PERIOD-1; i > 0; i--) {
        fitness_count[j].vhigh_nodes[i] =
            fitness_count[j].vhigh_nodes[i-1];
        fitness_count[j].high_nodes[i] =
            fitness_count[j].high_nodes[i-1];
        fitness_count[j].med_nodes[i] =
            fitness_count[j].med_nodes[i-1];
        fitness_count[j].low_nodes[i] =
            fitness_count[j].low_nodes[i-1];
        fitness_count[j].vlow_nodes[i] =
            fitness_count[j].vlow_nodes[i-1];
    }

    fitness_count[j].vhigh_nodes[0] = vhigh_count[j];
    fitness_count[j].high_nodes[0] = high_count[j];
    fitness_count[j].med_nodes[0] = med_count[j];
    fitness_count[j].low_nodes[0] = low_count[j];
    fitness_count[j].vlow_nodes[0] = vlow_count[j];
}

return 0;
}

```

```

/***** Filter Configuration File *****/

/* This is the file which contains the specification for what type of filter
   to do and the kind of graph specifications to generate for feeding to
   gnuplot.
*/
#define FILTER_SPEC_FILE      "filter.conf"

/* By default, the first filter (i.e. "filter_1") will be performed. */
static int desired_filter = 1;

/***** Specific Filters *****/

#include <math.h>

/* The filter_level determines which kinds of nodes will be considered in
   building certain statistics.

      Level      Meaning
      -----
      0          All nodes
      1          Only brigade nodes
      2          Only battalion nodes
      3          Only platoon and company nodes
      4          CSMA net communities are averaged together
      5          Local stub net communities are averaged together

*/
static int filter_level = 0;

/* Basic zeroth filter.  Outputs the fraction of jammed nodes within a
   simulation cycle.  NOTE: This filter will work correctly ONLY when
   the cell.output field in the Cellang code mirrors the complemented
   value of the cell.function field.  In other words,

      cell.output := 1 - cell.function.

*/
static void filter_0 (void) {
    int i, total = 0, node_count = 0;

    /* Add up the # of nodes for the current time step/phase.
    */
    for (i = 0; i < number_of_nodes; i++) {
        if (filter_level == 1 && node_info[i].echelon != 3)
            continue;
        else if (filter_level == 2 && node_info[i].echelon != 2)
            continue;
        else if (filter_level == 3 && node_info[i].echelon != 0
                 && node_info[i].echelon != 1)
            continue;

        total += time_in_state[i].state;
        node_count++;
    }
}

```

```

        printf("%d %lf\n", time/TIME_PHASES,
               (double) total / (double) node_count);
    }

/* Basic first filter. Outputs the situational awareness connectivity
   within a simulation cycle. NOTE: This filter will work correctly ONLY
   when the cell.output field in the Cellang code mirrors the value of the
   cell.connect field.
*/
static void filter_1 (void) {
    int i, total = 0, node_count = 0;

    /* Add up the # of nodes for the current time step/phase.
    */
    for (i = 0; i < number_of_nodes; i++) {
        if (filter_level == 1 && node_info[i].echelon != 3)
            continue;
        else if (filter_level == 2 && node_info[i].echelon != 2)
            continue;
        else if (filter_level == 3 && node_info[i].echelon != 0
                 && node_info[i].echelon != 1)
            continue;

        total += time_in_state[i].state;
        node_count++;
    }

    printf("%d %lf\n", time/TIME_PHASES,
           (double) total / (double) node_count / 1000.0);
}

/* Basic second filter. Outputs total and fraction of cells in each of five
   fitness states for the current time step.
*/
static void filter_2 (void) {

    int i, j, temp;
    int step[NUM_OF_NODE_STATES], node_count = 0;

    /* Initialize arrays. */
    memset(step, 0, NUM_OF_NODE_STATES*sizeof(int));

    /* Add up the # of nodes in each fitness category for each echelon
       for the current time step/phase.
    */
    for (i = 0; i < NUM_OF_ECHELONS; i++) {
        if (filter_level == 1 && i != 3)
            continue;
        else if (filter_level == 2 && i != 2)
            continue;
        else if (filter_level == 3 && i != 0 && i != 1)
            continue;

        step[4] += fitness_count[i].vhigh_nodes[0];
        step[3] += fitness_count[i].high_nodes[0];
        step[2] += fitness_count[i].med_nodes[0];
        step[1] += fitness_count[i].low_nodes[0];
    }
}

```

```

        step[0] += fitness_count[i].vlow_nodes[0];
    }
    node_count = step[0] + step[1] + step[2] + step[3] + step[4];

    printf(" %d", time/TIME_PHASES;
    printf(" %lf", (double) step[4] / (double) node_count);
    printf(" %lf", (double) step[3] / (double) node_count);
    printf(" %lf", (double) step[2] / (double) node_count);
    printf(" %lf", (double) step[1] / (double) node_count);
    printf(" %lf\n", (double) step[0] / (double) node_count);
}

/* Basic third filter. This produces both the state and temporal entropies.
The helper function "calc_entropies" makes it easier to do the averaging
over both bn and stub networks. Sub_net is the network number to
calculate the entropy for if the filter_level is either 4 or 5.
*/

static calc_entropies(double *state, double *temporal, int sub_net);

static void filter_3 (void) {
    double state_entropy, temporal_entropy;
    int i;

    if (filter_level == 4) {
        /* Average the entropies over all bn nets. */

        double total_state = 0.0, total_temporal = 0.0;
        for (i = 0; i < NUM_OF_BN_NETS; i++) {
            calc_entropies(&state_entropy, &temporal_entropy, i);
            total_state += state_entropy;
            total_temporal += temporal_entropy;
        }
        state_entropy = total_state / NUM_OF_BN_NETS;
        temporal_entropy = total_temporal / NUM_OF_BN_NETS;
    } else if (filter_level == 5) {
        /* Average the entropies over all stub nets. */

        double total_state = 0.0, total_temporal = 0.0;
        for (i = 0; i < NUM_OF_STUB_NETS; i++) {
            calc_entropies(&state_entropy, &temporal_entropy, i);
            total_state += state_entropy;
            total_temporal += temporal_entropy;
        }
        state_entropy = total_state / NUM_OF_STUB_NETS;
        temporal_entropy = total_temporal / NUM_OF_STUB_NETS;
    } else
        calc_entropies(&state_entropy, &temporal_entropy, 0);

    /* Print the results. */
    printf("%d %lf %lf\n", time/TIME_PHASES,
        -state_entropy, -temporal_entropy);
}

static double last_P[NUM_OF_NODE_STATES][MAX_NUM_OF_SUBNETS];

/* Helper function (and the real guts) for filter_3. */

```

```

static calc_entropies(double *state, double *temporal, int sub_net) {

    double state_entropy = 0, P[NUM_OF_NODE_STATES];

    double temporal_entropy = 0,
        P_cond[NUM_OF_NODE_STATES][NUM_OF_NODE_STATES];

    int j, k, step[NUM_OF_NODE_STATES], node_count = 0;

    /* Initialize arrays. */
    memset(step, 0, NUM_OF_NODE_STATES*sizeof(int));
    memset(P, 0, NUM_OF_NODE_STATES*sizeof(double));
    memset(P_cond, 0,
NUM_OF_NODE_STATES*NUM_OF_NODE_STATES*sizeof(double));

    /* Add up the # of nodes in each fitness category for each echelon,
       but only for the current time step/phase.
    */
    for (k = 0; k < number_of_nodes; k++) {
        if (filter_level == 1 && k != 3)
            continue;
        else if (filter_level == 2 && k != 2)
            continue;
        else if (filter_level == 3 && k != 0 && k != 1)
            continue;
        else if (filter_level == 4 && node_info[k].bn_net != sub_net)
            continue;
        else if (filter_level == 5 && node_info[k].stub_net != sub_net)
            continue;

        step[0] += (time_in_state[k].state == 0); /* VERY LOW */
        step[1] += (time_in_state[k].state == 1); /* LOW */
        step[2] += (time_in_state[k].state == 2); /* MEDIUM */
        step[3] += (time_in_state[k].state == 3); /* HIGH */
        step[4] += (time_in_state[k].state == 4); /* VERY HIGH */
    }

    node_count = step[0] + step[1] + step[2] + step[3] + step[4];

    /* Calculate the state entropy. */
    for (k = 0; k < NUM_OF_NODE_STATES; k++) {
        P[k] = (step[k] == 0 ? 0 : (double)step[k]/(double)node_count);
        state_entropy += log(pow(P[k], P[k]));
    }
    state_entropy = state_entropy/log((double) NUM_OF_NODE_STATES);

    /* Calculate the conditional probabilities. */
    for (j = 0; j < NUM_OF_NODE_STATES; j++)
        for (k = 0; k < NUM_OF_NODE_STATES; k++) {

            /* Count the number of nodes now in state k that were in
               state j during the last time step/phase.
            */
            int i, k_count = 0, last_j_count = 0;
            for (i = 0; i < number_of_nodes; i++) {
                if (filter_level == 1 && node_info[i].echelon != 3)

```

```

        continue;
    else if (filter_level == 2 && node_info[i].echelon != 2)
        continue;
    else if (filter_level == 3
             && node_info[i].echelon != 0
             && node_info[i].echelon != 1)
        continue;
    else if (filter_level == 4
             && node_info[i].bn_net != sub_net)
        continue;
    else if (filter_level == 5
             && node_info[i].bn_net != sub_net)
        continue;

    if (time_in_state[i].state == NUM_OF_NODE_STATES-k
        && last_states[i][0] == NUM_OF_NODE_STATES-j)
        k_count++;

    if (last_states[i][0] == NUM_OF_NODE_STATES-j)
        last_j_count++;
}

/* NOTE: If last_P[j] = 0 then it MUST be the case that
   k_count = 0, in which case the conditional probability
   is simply set to 0 (rather than attempting the faulty
   division by 0).
*/
P_cond[k][j] = (last_j_count == 0 ?
                0 :
                (double)k_count / (double)last_j_count);
}

/* Calculate the temporal entropy. */
for (j = 0; j < NUM_OF_NODE_STATES; j++)
for (k = 0; k < NUM_OF_NODE_STATES; k++) {

    temporal_entropy += log(pow(P_cond[k][j],
                               last_P[j][sub_net] * P_cond[k][j]));
}
temporal_entropy = temporal_entropy/log((double) NUM_OF_NODE_STATES);

/* Report the results. */
*state = state_entropy;
*temporal = temporal_entropy;

/* Update last_P for the next time step/phase calculations. */
for (k = 0; k < NUM_OF_NODE_STATES; k++)
    last_P[k][sub_net] = P[k];
}

/* Basic fourth filter. Produces return maps for all five fitness level
time series. Each line of output for a given time step/phase
represents the fraction of nodes within each state for the current time
step as well as for the previous time step. This filter produces no
output for T=0.
*/
static void filter_4 (void) {

```

```

int i, step[NUM_OF_NODE_STATES], node_count = 0;
    static double last_vhigh_frac = 0,
                  last_high_frac = 0,
                  last_med_frac = 0,
                  last_low_frac = 0,
                  last_vlow_frac = 0;

/* Initialize arrays. */
memset(step, 0, NUM_OF_NODE_STATES*sizeof(int));

/* Add up the # of nodes in each fitness category for each echelon,
   but only for the current time step/phase.
*/
for (i = 0; i < NUM_OF_ECHELONS; i++) {
    if (filter_level == 1 && i != 3)
        continue;
    else if (filter_level == 2 && i != 2)
        continue;
    else if (filter_level == 3 && i != 0 && i != 1)
        continue;

    step[4] += fitness_count[i].vhigh_nodes[0];
    step[3] += fitness_count[i].high_nodes[0];
    step[2] += fitness_count[i].med_nodes[0];
    step[1] += fitness_count[i].low_nodes[0];
    step[0] += fitness_count[i].vlow_nodes[0];
}
    node_count = step[0] + step[1] + step[2] + step[3] + step[4];

/* No output for T=0. */
if (time > 0) {
    printf("%d", time/TIME_PHASES);
    printf(" %.2f", (double) step[4] / (double) node_count);
    printf(" %f", last_vhigh_frac);
    printf(" %.2f", (double) step[3] / (double) node_count);
    printf(" %f", last_high_frac);
    printf(" %.2f", (double) step[2] / (double) node_count);
    printf(" %f", last_med_frac);
    printf(" %.2f", (double) step[1] / (double) node_count);
    printf(" %f", last_low_frac);
    printf(" %.2f", (double) step[0] / (double) node_count);
    printf(" %f\n", last_vlow_frac);
}

    last_vhigh_frac = (double) step[4] / (double) node_count;
    last_high_frac = (double) step[3] / (double) node_count;
    last_med_frac = (double) step[2] / (double) node_count;
    last_low_frac = (double) step[1] / (double) node_count;
    last_vlow_frac = (double) step[0] / (double) node_count;
}

/* This is the main driver for filtering. */
int do_filter (void) {
    time = get_time();

    if (time == 0) {
        FILE *config;

```

```

        if (init_data()) return 1; /* Quit now if there is a problem. */

        /* Read the desired filter from the FILTER_SPEC_FILE. */
        if ((config = fopen (FILTER_SPEC_FILE, "r")) == NULL) {
            fprintf(stderr,
                "FILTER ERROR: Cannot open file '%s'\n",
                FILTER_SPEC_FILE);
            return 1;
        }
        if (1 != fscanf(config, " %d", &desired_filter)) {
            fprintf(stderr,
                "FILTER ERROR: No int at beginning of file '%s'\n",
                FILTER_SPEC_FILE);
            return 1;
        }
        if (1 != fscanf(config, " %d", &filter_level)) {
            filter_level = 0;
        }
        close(config);

        printf ("# FILTER %d\n# LEVEL %d\n",
            desired_filter, filter_level);
    }

    if (time%TIME_PHASES == 0) {
        if (get_data()) return 1; /* Quit now if there is a problem. */

        switch (desired_filter) {
            case 0: filter_0(); break;
            case 1: filter_1(); break;
            case 2: filter_2(); break;
            case 3: filter_3(); break;
            case 4: filter_4(); break;
            default:
                fprintf(stderr, "FILTER ERROR: Unknown " \
                    "filter kind '%d' encountered.\n",
                    desired_filter);
                return 1;
        }
    }

    /* Since nothing is done before the end of a time phase, do nothing. */

    return 0;
}

```

INTENTIONALLY LEFT BLANK.

NO. OF
COPIES ORGANIZATION

2 DEFENSE TECHNICAL
INFORMATION CENTER
DTIC OCA
8725 JOHN J KINGMAN RD
STE 0944
FT BELVOIR VA 22060-6218

1 HQDA
DAMO FDT
400 ARMY PENTAGON
WASHINGTON DC 20310-0460

1 OSD
OUSD(A&T)/ODDR&E(R)
DR R J TREW
3800 DEFENSE PENTAGON
WASHINGTON DC 20301-3800

1 COMMANDING GENERAL
US ARMY MATERIEL CMD
AMCRDA TF
5001 EISENHOWER AVE
ALEXANDRIA VA 22333-0001

1 INST FOR ADVNCD TCHNLGY
THE UNIV OF TEXAS AT AUSTIN
3925 W BRAKER LN STE 400
AUSTIN TX 78759-5316

1 US MILITARY ACADEMY
MATH SCI CTR EXCELLENCE
MADN MATH
THAYER HALL
WEST POINT NY 10996-1786

1 DIRECTOR
US ARMY RESEARCH LAB
AMSRL D
DR D SMITH
2800 POWDER MILL RD
ADELPHI MD 20783-1197

1 DIRECTOR
US ARMY RESEARCH LAB
AMSRL CI AI R
2800 POWDER MILL RD
ADELPHI MD 20783-1197

NO. OF
COPIES ORGANIZATION

3 DIRECTOR
US ARMY RESEARCH LAB
AMSRL CI LL
2800 POWDER MILL RD
ADELPHI MD 20783-1197

3 DIRECTOR
US ARMY RESEARCH LAB
AMSRL CI IS T
2800 POWDER MILL RD
ADELPHI MD 20783-1197

ABERDEEN PROVING GROUND

2 DIR USARL
AMSRL CI LP (BLDG 305)

NO. OF
COPIES ORGANIZATION

1 OASD C3I
J BUCHHEISTER
RM 3D174
6000 DEFENSE PENTAGON
WASHINGTON DC 20301-6000

1 OUSD AT STRT TAC SYS
DR SCHNEITER
RM 3E130
3090 DEFENSE PENTAGON
WASHINGTON DC 20301-3090

1 OUSD(A&T)/S&T AIR WARFARE
R MUTZELBURG
RM 3E139
3090 DEFENSE PENTAGON
WASHINGTON DC 20301-3090

1 OUSD(A&T)/S&T LAND
WARFARE
A VIILU
RM 3B1060
3090 DEFENSE PENTAGON
WASHINGTON DC 20310-3090

1 UNDER SECY OF THE ARMY
DUSA OR
ROOM 2E660
102 ARMY PENTAGON
WASHINGTON DC 20310-0102

1 ASST SECY ARMY
ACQSTN LOGISTICS & TECHLGY
SAAL ZD ROOM 2E673
103 ARMY PENTAGON
WASHINGTON DC 20301-0103

1 ASST SECY ARMY
ACQSTN LOGISTICS & TECHLGY
SAAL ZP ROOM 2E661
103 ARMY PENTAGON
WASHINGTON DC 20310-0103

1 ASST SECY ARMY
ACQSTN LOGISTICS & TECHLGY
SAAL ZS ROOM 3E448
103 ARMY PENTAGON
WASHINGTON DC 20310-0103

NO. OF
COPIES ORGANIZATION

1 OADCSOPS FORCE DEV DIR
DAMO FDZ
ROOM 31522
460 ARMY PENTAGON
WASHINGTON DC 20301-0460

1 HQDA
ODCSPER
DAPE MR RM 2C733
300 ARMY PENTAGON
WASHINGTON DC 20301-0300

1 US ARMY ARMAMENT RDEC
AMSTA AR TD
M FISETTE BLDG 1
PICATINNY ARSENAL NJ
07806-5000

1 US ARMY MISSILE RDEC
AMSMI RD
DR W MCCORKLE
REDSTONE ARSENAL AL
35898-5240

1 NATICK SOLDIER CENTER
SBCN T
P BRANDLER
KANSAS STREET
NATICK MA 01760-5056

1 US ARMY TANK AUTOMTV RDEC
AMSTA TR
J CHAPIN
WARREN MI 48397-5000

1 US ARMY INFO SYS ENGRG CMD
AMSEL IE TD
DR F JENIA
FT HUACHUCA AZ 85613-5300

1 US ARMY SIM TRNG INST CMD
AMSTI CG
DR M MACEDONIA
12350 RESEARCH PKWY
ORLANDO FL 32826-3726

1 US ARMY TRADOC
BATTLE LAB INTEGRATION
TECH & CONCEPTS DIR
ATCD B
FT MONROE VA 23651-5850

<u>NO. OF COPIES</u>	<u>ORGANIZATION</u>
1	US ARMY TRADOC ANL CTR ATRC W A KEINTZ WSMR NM 88002-5502
1	US ARMY RESEARCH OFFICE 4300 S MIAMI BLVD RESEARCH TRIANGLE PARK NC 27709
1	US ARMY RESEARCH LAB AMSRL SL C HOPPER WSMR NM 88002-5513
1	US ARMY RESEARCH LAB AMSRL SL E J PALOMO WSMR NM 88002-5513
1	US ARMY RESEARCH LAB AMSRL SL EA R FLORES WSMR NM 88002-5513
1	US ARMY RESEARCH LAB AMSRL SL EI J NOWAK FT MONMOUTH NJ 07703-5601

ABERDEEN PROVING GROUND

1	SBCCOM RDEC AMSSB RTD J ZARZYCKI 5183 BLACKHAWK RD APG MD 21010-5424
1	US ARMY DEV TEST COM CSTE DTC TT T APG MD 21005-5055
1	US ARMY EVALUATION CENTER CSTE AEC W HUGHES 4120 SUSQUEHANNA AVE APG MD 21005-3013

<u>NO. OF COPIES</u>	<u>ORGANIZATION</u>
<u>ABERDEEN PROVING GROUND (CONT)</u>	
1	US ARMY EVALUATION CENTER CSTE AEC SV L DELATTRE 4120 SUSQUEHANNA AVE APG MD 21005-3013
12	DIR USARL AMSRL SL DR WADE J BEILFUSS AMSRL SL B J FRANZ W WINNER AMSRL SL BA M RITONDO AMSRL SL BD J MORRISSEY AMSRL SL BE D BELY AMSRL SL BG A YOUNG AMSRL SL BN D FARENWALD AMSRL SL E M STARKS AMSRL SL EC E PANUSKA AMSRL SL EM J FEENEY

NO. OF
COPIES ORGANIZATION

3 INFO SCIENCES TEAM
PHYS SCIENCE LAB
DR R BERNSTEIN JR
DR R SMITH
DR M COOMBS
PO BOX 30002
LAS CRUCES NM
88003-8002

1 BOOZ ALLEN & HAMILTON INC
P MURPHY
4001 FAIRFAX DRIVE
SUITE 750
ARLINGTON VA 22203

1 CONVERGENT COMPUTING INC
J D ECKART
1260 RADFORD ST SUITE A
CHRISTIANSBURG VA 24073

5 US ARMY RESEARCH LAB
AMSRL SL EA
DR K MORRISON
DR P DJANG
D LANDIN
T READER
G GUZIE
WSMR NM 88002-5513

1 US ARMY RESEARCH LAB
AMSRL SL EI
P BOTHNER
FT MONMOUTH NJ
07703-5601

NO. OF
COPIES ORGANIZATION

ABERDEEN PROVING GROUND

1 US ARMY DEV TEST CMD
CSTE DTC TT T
APG MD 21005-5055

17 DIR USARL
AMSRL SL BE
P TANENBAUM
R SANDMEYER
J ANDERSON
R SAUCIER
AMSRL SL BN
E FIORAVANTE
AMSRL SL EA
D BAYLOR
B RUTH (10 CPS)
R ZUM BRUNNEN

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project(0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE December 2001		3. REPORT TYPE AND DATES COVERED Final, October 1999–September 2000
4. TITLE AND SUBTITLE Modeling the Emergent Dynamics of a Tactical Situational Awareness Network Within an Information Operations (IO) Environment			5. FUNDING NUMBERS 665604D670	
6. AUTHOR(S) Brian G. Ruth and J. Dana Eckart*				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) U.S. Army Research Laboratory ATTN: AMSRL-SL-EA Aberdeen Proving Ground, MD 21005-5068			8. PERFORMING ORGANIZATION REPORT NUMBER ARL-TR-2641	
9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES)			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES *Convergent Computing, Inc., 1260 Radford St., Suite A, Christiansburg, VA 24073.				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) A model developed to analyze the global emergent dynamics of situational awareness (SA) dissemination within a digitized brigade undergoing hostile information operations (IO) stress events is presented. Networked battlefield platforms are modeled as interconnected tactical internet nodes through the use of cellular automata. The cellular automata form a discrete spatially extended dynamical system consisting of a parallel networked lattice of computational cells in two dimensions. The global impact of localized IO stress events on SA dissemination can then be analyzed as a function of time by mapping the SA dissemination architecture onto the cellular automata lattice. The lattice cells can represent tactical internet client or server platforms. This model demonstrates the fluctuating patterns of situational "self-awareness" (i.e., a friendly platform's situational awareness of all other friendly platforms within its area of operations) which emerge across the digitized brigade on a global scale and also illustrates the emergent dynamics of SA dissemination within a "movement-to-contact" brigade structure.				
14. SUBJECT TERMS cellular automata, tactical communication network, situational awareness, information operations, emergent behavior			15. NUMBER OF PAGES 121	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL	

INTENTIONALLY LEFT BLANK.