A Framework to Control Emergent Survivability of Multi Agent Systems

Aaron Helsinger, Karl Kleinmann & Marshall Brinn
BBN Technologies
[ahelsing, kkleinmann, mbrinn]@bbn.com
The Problem

- DMAS are complex
  - By definition, many independent entities autonomously pursuing goals, spread out over an unreliable network
  - Application Function is itself *emergent*
- As with any complex system, **chaos is a fact of life**
- Predictability is impossible at the *micro* level
  - Multithreading, timing, etc.
  - The autonomy of agents exacerbates this, as does the network over which you distribute them.
- A DMAS can fail in many unpredictable ways.
  - No complex system can anticipate all problems, nor be impervious to all attacks.

- For widespread adoption, the agent community must provide confidence in DMAS systems to reliably perform under stress.
Emergent Survivability

- Our only hope is to
  - Limit the impact to the micro level, and
  - Keep the macro stable.
  - Make tradeoffs, or suffer catastrophic functionality loss.

- We engineer the system to tolerate degradation in some dimensions, while trying to maximize overall system performance.
  - Measure resources, application function, stresses, and survivability at runtime.
  - Build a hierarchy of control loops to measure performance at macro level and control behavior at micro level.

- The system can reason about its survivability in real time and adjust resources in the face of attacks at multiple levels, producing Emergent Survivability.

Degrade without Failing
1) Measure Performance

- Identify the dimensions of application function
  - E.g. Timeliness, correctness, completeness
  - Include survivability, e.g. integrity, accountability, robustness
  - Measure system resources, stresses, and performance

- Must define these correctly
  - If they are too micro, they will vary wildly.
  - If they measure the wrong quantities, they will not vary with the application performance

- Build sensors for collecting these data
  - In-band, lightweight, and real-time
  - See my AAMAS03 paper details

- Functions for weighting measures and producing a scalar overall system score
The key idea of our framework is to build a hierarchy

- Reasoning at the **macro** level
- Acting at the **micro** level
- Decisions are made close to the resources in contention or actions capable of addressing the issue,
  - Without being susceptible to minor chaotic variations.
- Succession of layers; *One layer’s micro is another layer’s macro*
- These levels are managed by a nested set of control loops.
UltraLog Program

- DARPA effort
- Integrated contributions of 15-20 companies and universities
- Show assessable wartime survivability
- Prototype application is military logistics
  - Real algorithms and organizations
  - Plan, transport, and execute 180 day deployment
  - FCS scenario
  - Resulting log plan has 250K+ individual elements representing demand and transport for 34K+ entities of 200+ types.
UltraLog Survivability Requirements

Program Goal (per original program description):
System will incur no greater than a 20% capabilities loss and a 30% performance loss under conditions of 45% information infrastructure loss, wartime loads, and directed information warfare.

- Stress, System Function and Degradation are Quantitative in Nature
- Three categories of stress
  - Loss (total or partial) of hardware capabilities (CPU, BW, Memory, Disk)
  - Significant increases in legitimate work to perform
  - Attempts to circumvent system integrity (confidentiality, authentication, authorization)

Survivability: Extent to which system function is maintained under stress.
The Cougaar Architecture

- Cougaar architecture is designed to support:
  - data intensive,
  - inherently distributed applications,
  - emphasizing scalability & configurability.

- Cougaar is:
  - 100% Java agent architecture
  - Expressly for building large distributed MAS
  - Around 400K lines of code.

- Prototype application:
  - Uses over 1092 agents
  - over a 9-LAN network of
  - over 85 machines. It is
  - Data- and compute- intensive,
  - Inherently distributed, and must
  - Plan and execute a logistics deployment.

- Developed under DARPA funding

- Cougaar is Open-Source (BSD-style license)

[http://www.cougaar.org]
Prototype Application MOPs

Swing Weights November 03

- Measure Performance
- Weight Measures
- Compute Overall Survivability Score
**Library of Adaptive Services**

- **Adaptive Robustness**
  - No single points of failure (SPOFs)
  - Automated recovery from resource loss
    - Planned or unplanned agent and machine loss
    - Proactive response to perceived threat
    - Lost network component (temporary or permanent)
  - Resource management
    - Load balancing
    - Load shedding

- **Adaptive Security**
  - Application software integrity:
    - Signed jars, Java security mgr
  - Data integrity:
    - Signed and encrypted messages
    - Signed and encrypted data files
  - Access control:
    - Maintain an identity and certificates for “Principles”
    - Policy-based access control of servlets, messages, and blackboard objects
UltraLog Control Hierarchy

- **Society**
  - Top level, with user input
  - Policy manager
  - Cross-community coordinator

- **Community**
  - Security, robustness, LAN communities & resources
  - Policy controlled, Defense Coordinator balances priorities

- **Host or JVM**
  - Host level resources managed by policy, Adaptivity Engine, coordinator

- **Agent**
  - Tailor local operations and goals
  - Adaptivity Engine reasons using a local book of plays, configuring local components
The Adaptivity Engine is the heart of the Agent or Node-level control loop.

An Adaptivity Engine in an agent or node will be run off a playbook that determines what operating modes and policies should be invoked on sub-components to achieve a desirable aggregate performance:

- Based on measurements of current and expected performance and situation.

A playbook represents rules for adaptivity actions based on performance regions. Examples:

- “Enter Operating mode X when CPU > X and RT-Performance=‘Falling Behind’”
- “Establish Policy ABC when THREATCON>=3”

The Adaptivity Engine at any given level needs to make periodic measurements, determine the current operating region and take appropriate action (control loop).
Adaptivity Engine Architecture

**Operating Mode Policy Manager**
- Sets Operating Modes for Components based on plays in Playbook and current Sensor Conditions
- Constrain Playbook based on 'leaf' OperatingMode policy direction
- Read Playbook

**Adaptivity Engine**
- Publish changes to operating modes
- Publish InterAgentOperatingModePolicies
- Get Condition by name
- Get OperatingMode by name

**Blackboard**
- Publish real-time sensor conditions: Load, Resource Availability, THREATCON, Possibly, Current Settings, Current Performance
- Operating Modes (Knob Settings)
- RelayLP

**Sensors**
- Publish InterAgent Operating Mode Policies

**Processing Components (Plugins)**
- Get OperatingMode by name

**Condition Service**
- Get Condition by name

**Playbook Manager**
- Read Playbook
- Constrain Playbook Service
- Playbook Read Service

**Other agents**
Defense Coordinator

- The Coordinator is the brains of the Host or Community level control loop.

- Goals
  - Deconflict competing Defense actions
  - Choose “best” actions based on:
    - Collection of Defense diagnoses
    - Belief about threat environment
    - High-level policy (as specified by MAU curves)
  - Local decisions when possible for efficiency

- Key design points
  - POMDP for belief calculations
  - XML TechSpecs for threats, assets, and defenses
  - Cost/Benefit analysis to select actions
Coordinator Architecture

Node

- Coordinator
  - Diagnoses
  - ActionOffers
  - CurrentAction
  - AllowedActions
    - WhenConnected
    - WhenDisconnected

Local Defense

Agent

Management Agent

- Coordinator
  - Diagnoses
  - ActionOffers
  - CurrentAction
  - AllowedActions
    - WhenConnected
    - WhenDisconnected

Community Defense

AAMAS'04
Assessment of Prototype

- Extensive integration lab
- Annual testing cycle
  - Engineering testing
  - Security red team
  - Functional assessment
  - Survivability assessment
- Web accessible run results
  - Tools for running & stressing society
  - Automatic report generation
  - Display debug information & survivability scores
## 2003 Assessment Results

### Improved Stress Results

<table>
<thead>
<tr>
<th>Stress</th>
<th>2000</th>
<th>2001</th>
<th>2002</th>
<th>2003</th>
<th>2003 Stress</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scalability</td>
<td>FAIL</td>
<td>OK</td>
<td>OK</td>
<td>PASS</td>
<td>Wartime loads1</td>
</tr>
<tr>
<td></td>
<td>FAIL</td>
<td>FAIL</td>
<td>OK</td>
<td>OK</td>
<td>Wartime loads2</td>
</tr>
<tr>
<td></td>
<td>FAIL</td>
<td>FAIL</td>
<td>OK</td>
<td>PASS</td>
<td>Wartime loads3</td>
</tr>
<tr>
<td></td>
<td>FAIL</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>Thrashing</td>
</tr>
<tr>
<td></td>
<td>FAIL</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>Scaling of nodes and agents</td>
</tr>
<tr>
<td></td>
<td>FAIL</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>Scaling logistics problem</td>
</tr>
<tr>
<td>Security</td>
<td>OK</td>
<td>PASS</td>
<td>PASS</td>
<td>PASS</td>
<td>Fraudulent, untrusted code</td>
</tr>
<tr>
<td></td>
<td>FAIL</td>
<td>PASS</td>
<td>PASS</td>
<td>PASS</td>
<td>Untrusted communications</td>
</tr>
<tr>
<td></td>
<td>FAIL</td>
<td>PASS</td>
<td>PASS</td>
<td>PASS</td>
<td>Insecure / dangerous code</td>
</tr>
<tr>
<td></td>
<td>FAIL</td>
<td>FAIL</td>
<td>OK</td>
<td>PASS</td>
<td>Corruption of persisted state</td>
</tr>
<tr>
<td></td>
<td>FAIL</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>Unauthorized processing</td>
</tr>
<tr>
<td></td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>PASS</td>
<td>Unexpected plugin behavior</td>
</tr>
<tr>
<td></td>
<td>OK</td>
<td>PASS</td>
<td>PASS</td>
<td>PASS</td>
<td>Component masquerade</td>
</tr>
<tr>
<td></td>
<td>FAIL</td>
<td>OK</td>
<td>PASS</td>
<td>PASS</td>
<td>Compromised agents1</td>
</tr>
<tr>
<td></td>
<td>FAIL</td>
<td>OK</td>
<td>PASS</td>
<td>PASS</td>
<td>Compromised agents2</td>
</tr>
<tr>
<td></td>
<td>FAIL</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>Intrusion</td>
</tr>
<tr>
<td></td>
<td>FAIL</td>
<td>FAIL</td>
<td>OK</td>
<td>OK</td>
<td>Compromised communications</td>
</tr>
<tr>
<td></td>
<td>FAIL</td>
<td>FAIL</td>
<td>OK</td>
<td>OK</td>
<td>Snooping</td>
</tr>
<tr>
<td></td>
<td>OK</td>
<td>PASS</td>
<td>PASS</td>
<td>PASS</td>
<td>Message intercept</td>
</tr>
<tr>
<td>Robustness</td>
<td>FAIL</td>
<td>OK</td>
<td>PASS</td>
<td>PASS</td>
<td>Processing failure1</td>
</tr>
<tr>
<td></td>
<td>FAIL</td>
<td>OK</td>
<td>PASS</td>
<td>PASS</td>
<td>Processing failure2</td>
</tr>
<tr>
<td></td>
<td>FAIL</td>
<td>OK</td>
<td>PASS</td>
<td>PASS</td>
<td>Network failure1</td>
</tr>
<tr>
<td></td>
<td>FAIL</td>
<td>OK</td>
<td>PASS</td>
<td>PASS</td>
<td>Network failure2</td>
</tr>
<tr>
<td></td>
<td>FAIL</td>
<td>OK</td>
<td>PASS</td>
<td>PASS</td>
<td>Processing contention</td>
</tr>
<tr>
<td></td>
<td>FAIL</td>
<td>OK</td>
<td>OK</td>
<td>PASS</td>
<td>DOS attack</td>
</tr>
</tbody>
</table>

### Passed 45% Loss Test

- **20% Capability Loss**
- **Passed 45% Loss Test**
- More Stresses PASS in 03

BBN TECHNOLOGIES

AAMAS'04  45% Infrastructure Loss
Future Work

➢ There is much we’d like to do in our UltraLog implementation
  ✓ Coordinators that learn
  ✓ Show how performance degrades with increased stress – where is the breaking point?

➢ There is work to do in proving our approach
  ✓ Apply to different application domains
    • What if anything is specific to logistics?
  ✓ Experiment with alternate control frameworks
    • Performance
    • Survivability

Related work available at cougaar.org: Many posted papers
Summary

- DMAS must be proven survivable to be adopted
- Survivability can emerge from DMAS operations
- To do so:
  - Measure survivability at runtime as part of system function
  - Create nested control loops that
    - measure at macro levels and
    - act at micro levels to
    - produce overall application function survivability
- UltraLog has done so using the Cougaar architecture
  in a 1092 agent logistics application

Use & add to Cougaar at http://www.cougaar.org