AUTONOMIA: An Autonomic Control and Management

- Provide dynamically programmable control and management services to support the development and deployment of smart (intelligent) applications
- Provide automated performance and fault tolerant support for smart applications
- Provide automated deployment, registration, discovery of components
- Provide automated configuration of applications and system resources
- Provide secure, open computing environment

Autonomia Architecture

Self-Configuring Architecture

Salim Hariri/University of Arizona
Self-Configuring Profiling scripts (XML)

```xml
<Engine name="Self-Configuration"
	task name="MMService version=1.0"
	author="Salim Hariri/University of Arizona"
>
	<configuration>
		<checkmachines>
			<host = "M1"/>
		</checkmachines>
	<!--- Checking Point configuration, set as 120 seconds default --->
	<component attributes>
		<Checking Point Interval>120</Checking Point Interval>
	</component attributes>
	<!--- Rules for tuning the checking point interval --->
	<component attributes>
		<interval tuning rules>
			<!-- If the CPU utilization is over 90%, Change the interval to 300 seconds.--->
			cpu_util > 90%, Checking Point Interval = 300
		</interval tuning rules>
	</component attributes>

</implementation>
</task>
</Engine>
```

Self-Configuring Algorithm

```java
Do {
	Read the table component;
	Copy the data into attributes repository from DB;
	Total_Configuration_policy = maximum policy_ID of table Selfconfig_Policies table;
	Subscribe the attributes data on event server according to the configuration attributes
	for (i = 0; i < Total_Configuration_policy; i++)
		If (javaspace.take())
			Read the real-time attributes from event server;
		else{
			break;
		}
		if (attribute condition matched)
			Choose available host from Resource Repository;
			Choose available component from Component Repository;
			Get the respective agent system info by component;
			Send the change notification;
		Write the data into DB Log table.

While the periodical time is arrived
```

Self-Protecting Architecture

[Diagram showing the architecture with various components and their interactions]
Self-Protecting Algorithm

Application deployment by the developer or infrastructure.
Agent sends request from application.
The agent is required to provide authentication using one of the following:
- Password-based
- Public Key Infrastructure
- X509 certificates

```matlab
if (agent is successfully authenticated)
    agent establishes intention list for the host.
    semantic check intention list for valid tasks.
    if (task is valid)
        task is included in the intention list.
        if (agent is authenticated)
            execute the task.
        else
            agent exits.
    endif;
else
    agent not authenticated.
    agent exits.
endif;
if (intention list is published)
    if (task that the agent is trying to execute is on the intention list)
        give access to the task based on the level of access from the ACL.
    else
        the agent is denied permission to execute on the host.
    endif;
else
    agent obtains permission from ACL.
    execute the task.
endif;
```

Network Vulnerability Analysis Toolset: Self-Protecting and Self-Healing

- Architecture
- Vulnerability Analysis
- Attack Scenario & Analysis
- Future work

NVAT - Objective

- Online modeling, monitoring, and analyzing network vulnerability.
- Adaptive learning and automatically identifying critical vulnerable Internet infrastructure resources.
- Proactive self-healing and preventing the network from a wide range of Internet infrastructure faults and/or attacks.
Methodology – Vulnerability Index

Network Services

Vulnerability Index

Under normal state

Vulnerable
Normal
Uncertain

Under attacks/Faults

Architecture

Online Monitoring Engine
System Vulnerability State
Adaptive Learning Engine
Self-healing Engine
Network Topology

Client
Server
Router

Topology for Impact Analysis

Total Clients, Routers and Servers in the Topology = 243

Protocols Used for simulation - OSPF, TCP/IP, UDP

Traffic on the backbone link is filled with UDP Client and Servers

Parallel Simulation on Beowulf Cluster with 4 timelines and 2 processors
**Vulnerability Impact Analysis**

**Component Vulnerability Index** – The CVIs quantify the network components’ vulnerability states under fault/attack scenario $FS_k$.

$$CVI_{Client, FS_k} = \frac{R_{\text{Client}} - R_{\text{min}}}{R_{\text{max}} - R_{\text{min}}} \quad CVI_{Router, FS_k} = \frac{R_{\text{Router}} - R_{\text{min}}}{R_{\text{max}} - R_{\text{min}}}$$

**System Impact Factor** – The SIFs quantify the impact of the components’ vulnerability on the overall network system operation.

$$SIF_{Client, FS_k} = \frac{\sum CVI_j}{\text{Max. no. Clients}} \quad SIF_{Router, FS_k} = \frac{\sum CVI_j}{\text{Max. no. Routers}}$$

Where, $COS_j = 1$ if $CVI_j > \delta$ otherwise $COS_j = 0$
### System Impact Factor

<table>
<thead>
<tr>
<th>Routers Failed</th>
<th>Core Routers SIF %</th>
<th>Total Clients SIF %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Router 4</td>
<td>54.28571</td>
<td></td>
</tr>
<tr>
<td>Router 5</td>
<td>41.428</td>
<td></td>
</tr>
<tr>
<td>Router 7</td>
<td>28.2142</td>
<td></td>
</tr>
<tr>
<td>Router 2</td>
<td>27.1428</td>
<td></td>
</tr>
<tr>
<td>Router 3</td>
<td>23.2142</td>
<td></td>
</tr>
<tr>
<td>Router 1</td>
<td>17.8517</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Routers Failed</th>
<th>Core Routers SIF %</th>
<th>Total Clients SIF %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Router 4 &amp; 5</td>
<td>53.4285</td>
<td></td>
</tr>
<tr>
<td>Router 2 &amp; 4</td>
<td>27.24</td>
<td></td>
</tr>
<tr>
<td>Router 3 &amp; 5</td>
<td>42.87</td>
<td></td>
</tr>
</tbody>
</table>

Critical Routers in Multiple Router Failures

<table>
<thead>
<tr>
<th>Routers Failed</th>
<th>Core Routers SIF %</th>
<th>Total Clients SIF %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Router 4 &amp; 5</td>
<td>53.4285</td>
<td></td>
</tr>
<tr>
<td>Router 2 &amp; 4</td>
<td>27.24</td>
<td></td>
</tr>
<tr>
<td>Router 3 &amp; 5</td>
<td>42.87</td>
<td></td>
</tr>
</tbody>
</table>

### QoP - VI Based Quality of Protection

1. **Task 1: Online Monitoring - Flow VI**
   - **Classifier:** incoming packets
   - **Attack Traffic:** Class $V_{IC}$
   - **Suspicious Traffic:** Class $V_{IS}$
   - **Normal Traffic:** Class $V_{IN}$

2. **Update Global Flow VI and Vulnerable Router List**

3. **Collect Flow Traffic Analysis**

4. **Vulnerability Analysis**

5. **Event Correlation**

6. **Flow VI > Normal?**
   - **Yes:**
     - **Scheduler:**
     - **Output Queues:**
   - **No:**
     - **Suspicious Traffic:**
     - **Flow:**
     - **VI $> V_{IS}$**
     - **Suspicious Traffic:**
     - **Flow:**
     - **VI $> V_{IC}$**

### Topology for QoP Analysis

- **Traffic Configuration**
  - Legitimate client traffic through same interface to other servers
  - Legitimate client traffic through different interface to attacked server
  - Legitimate client traffic through same interface to attacked server and towards attack targets
  - Legitimate server traffic (heavy) through different interface and towards other clients
  - Attack traffic
QoP Simulation Analysis

Salim Hariri/University of Arizona

Win32/SQL Slammer Worm
Demonstration (2)

- Utilizes the vulnerability in SQL server
- 5 hours from 12:30am EST Jan 25, 2003 packet loss across the Internet approaches 30%
- 200,000-300,000 servers were infected

Metrics for SQL Slammer Analysis

Salim Hariri/University of Arizona
**SQL Slammer Demonstration (2)**

**CVI of 2:8 and 3:9**

**Number of successful sessions**
CAI in the form of outgoing packets

Network Attack/Propagation Analysis

Network Attack Identification Analysis
Large computation requirements

- e.g. simulation of the core-collapse of supernovae in 3D with reasonable resolution (500^3) would require ~10-20 teraflops for 1.5 months (i.e. ~100 Million CPUs!) and about 200 terabytes of storage.
- e.g. turbulent flow simulations using active flow control in aerospace and biomedical engineering requires 5000x1000x500=2.5·10^9 points and approximately 10^7 time steps, i.e. with 1GFlop processors requires a runtime of ~7-106 CPU hours, or about one month on 10,000 CPUs! (with perfect speedup). Also with 700B/pt the memory requirement is ~1.75TB of run time memory and ~800TB of storage.

Computational Modeling and the Grid

- The Computational Grid
- Potential for aggregating resources
- Potential for seamless interactions
- New applications formulations

Developing application to utilize and exploit the Grid remains a significant challenge:
- The problem: a level of complexity, heterogeneity, and dynamism for which our programming environments and infrastructure are becoming unmanageable, brittle and insecure
- System size, heterogeneity, dynamics, reliability, availability, usability
- Currently (typically) proof-of-concept demos by “hero programmers”
- Requires fundamental changes in how applications are formulated, composed and managed
- Breaks current paradigms based on passive components and static compositions
- autonomic components and their dynamic composition, opportunistic interactions, virtual runtime, ...

Autonomic Computational Science and Engineering
Concluding Remarks

- Our system design methods and management tools seem to be inadequate for handling the complexity, size, and heterogeneity of today and future Information systems
- We need Information System Engineering so we build our systems based on science & engineering principles rather than ad hoc, trial/errors
- We need New Ways to Design Information Systems and Services – Autonomic Computing
  - Biological systems have evolved strategies to cope with dynamic, complex, highly uncertain constraints