Multiple Views on Multiplicity Computing: Opportunities Viewed through a Cyber-Security Lens

CREST Workshop

Rick Schantz, Partha Pal, Aaron Paulos, Joe Loyall, Kurt Rohloff

Distributed Systems Technology Group

March 23, 2012
1982: R&D Computing Landscape

Multiplicity emerging ...
1982: Heterogeneity, Specialization Among Plenty (or so it seemed at the time)
1990s Integrated Adaptive System Concept

- System-wide QoS
- Application or Domain-specific QoS
- Common Middleware Services QoS
- Distribution Middleware QoS
- Operating System QoS
- Network QoS

QoS Adaptive Control

ACE/TAO RT ORB
Dynamic Quality of Service is a Key Aspect of Mission Critical Distributed Systems

QoS management for distributed systems strives to provide a predictable high level of mission effectiveness and user satisfaction within available resources.

- Capture QoS aspects of mission requirements
- Effectively utilize available resources for mission effectiveness
- Manage the resources that could become bottlenecks
- Mediate conflicting demands for resources
- Dynamically reallocate as conditions change
Allocating Resources According to Utility

- How to determine mission utility?
- Each mission has multiple sets of tasks called application strings.
  - Take weighted sum of string utilities
    \[ UA_i^m = \sum_{i=1}^{N_i} w_j^s UA_j^s \]
  - Weighting for relative importance of strings.
- String utility
- Quality of Service Factors:
  - Timeliness
  - Availability
  - Quality
  - Throughput

\[ UA_j^s = F(T, a, q, Th) \]
End-to-end QoS management must
- Manage all the resources that can affect QoS, i.e., anything that could be a bottleneck at any time during the operation of the system (e.g., CPU, bandwidth, memory, power, sensors, …)
- Shape the data and processing to fit the available resources and the mission needs
  - What can be delivered/processed
  - What is important to deliver/process
- Includes capturing mission requirements, monitoring resource usage, controlling resource knobs, and runtime reallocation/adaptation

Control and Monitor Network Bandwidth
- Set DiffServ CodePoints (per ORB, component server, thread, stream, or message)
- Work with DSCP directly or with higher level bandwidth brokers
- Priority-based (Diffserv) or reservation-based (RSVP)

Control and Monitor CPU Processing
- CPU Reservation or CPU priority and scheduling
- Have versions that work with CPU broker, RT CORBA, RTARM

Shape and Monitor Data and Application Behavior
- Shape the data to fit the resources and the requirements
- Insert using components, objects, wrappers, aspect weaving, or intercepters
- Library that includes scaling, compression, fragmentation, tiling, pacing, cropping, format change

Coordinated QoS Management
System resource managers allocate available resources based on mission requirements, participants, roles, and priorities
Local resource managers decide how best to utilize the resource allocation to meet mission requirements
Dynamic QoS realized by
- Assembly of QoS components
- Paths through QoS components
- Parameterization of QoS components
- Adaptive algorithms in QoS components
2000s Multi-Layered QoS Management for Service-Oriented Distributed Information Systems

QoS management across multiple users
- Fairness, resource allocations, importance

QoS enforcement mechanisms
- Differentiated service
- Thread and queue control
- Rate control, compression, filtering, replacement

Mission-level QoS policies
- Roles, importance, deadlines, user prefs.

Enforce QoS policies at local decision points
- Priorities of operations and information
- Resource access and process/info shaping
From Protection to Auto-Adaptive to Survivable and Self-Regenerative Systems

No system is perfectly secure—only adequately secured with respect to the perceived threat.

Prevent Intrusions (Access Controls, Cryptography, Trusted Computing Base)

But intrusions will occur

Detect Intrusions, Limit Damage (Firewalls, Intrusion Detection Systems, Virtual Private Networks, PKI)

But some attacks will succeed

Tolerate Attacks (Redundancy, Diversity, Deception, Wrappers, Proof-Carrying Code, Proactive Secret Sharing)

1st Generation: Protection

Trusted Computing Base

Access Control & Physical Security

Cryptography

Firewalls

Boundary Controllers

Intrusion Detection Systems

VPNs

PKI

2nd Generation: Detection

Intrusion Tolerance

Big Board View of Attacks

Real-Time Situation Awareness & Response

Graceful Degradation

Hardened Operating System

3rd Generation: Intrusion Tolerance and Survivability
Survivability and Intrusion Tolerance

Premise
• The number & sophistication of cyber attacks is increasing – some of these attacks will succeed

Philosophy
• *Operate through attacks* by using a layered defense-in-depth concept
  • Accept some degradation
  • Protect (C,I, A) of most valuable assets (information, services, …)
  • Move faster than the intruder

Approach
• “Defense Enabling” Distributed Applications
• Survivability architecture

• Exploring beyond degradation-- regain, recoup, regroup and even improve
• Semi-automated: Survivability architecture captures a lot of low level (and sometimes uncertain and incomplete) information – utilizes advanced reasoning and machine learning
Slowly Advancing from Defending to Tolerance to Survivability toward Regeneration

- AQuA
- OIT
- QuOIN
- Defense Enabling
- Autonomic Defense
- Unpredictability
- Byzantine FT
- Survivability Architectures and IMSes
- Cognitive Survivability Management

Adaptive Distributed Object Middleware

Red Team Assessments

Survivable and Secure Systems

APOD: Applications that Participate In Their Own Defense

ITUA: Intrusion Tolerance Through Unpredictable Adaptation

DPASA: Designing Adaptation And Protection into a Survivability Architecture

Self-Regenerative Survivable systems

CSISM*

Focus Area

DARPA
AFRL
DHS/HSARPA

Survivable SOA-based Systems

Cognitive Support for Intelligent Survivability Mgmt

* Cognitive Support for Intelligent Survivability Mgmt
Military (USAF) Joint Battlespace Infosphere (JBI) information management system exemplar made survivable and subjected to sustained attacks over several weeks by multiple independent red teams.

**Results**
- The system survived 75% of attacks
- Of those that succeeded,
  - Average time to failure was 45 minutes
  - Vs. immediately in the unprotected system
  - Minimum of 10 minutes to failure
  - Required combinations of attacks
- Adaptive defenses added 5-20% overhead to call latency

**Challenge:** Develop automated mechanism that would interpret the reports and decide the effective course of action

**CSISM Approach:** 3 level decision making - reactive, deliberate and learned; use theorem proving and coherence to reason about accusatory and evidentiary information contained in reported events

**Results**
- Possible to minimize expert involvement
- Reasoning about accusatory and evidentiary information wrt encoded knowledge
  - Made correct decision in ~75% cases in red team exercises
  - Compute intensive
- Integrating learned responses online needs additional research
Elements of Cyber-Defensive Ideas

- Common threads that runs through our intrusion tolerance and survivability work:
  - Adaptation for security
    - Like in nature, services migrate; change behavior, structure and configuration in order to survive
  - Unpredictability
    - Changing and taking unexpected actions yield advantages
  - Intelligent behavior
    - Like high order life forms, cognitive capabilities are introduced to survivable systems for interpreting reported events and making decisions
  - Evolution
    - Learning to improve defenses over time
Objectives

The objective is to sustain mission effectiveness. Different mission components have different security needs and will make different trade-offs at run-time between these, quality of service, and even correctness.

- Provide 100% critical functions at all times in spite of attacks.
- Design out the root causes of all current technical vulnerabilities.
- Adapt around corruptions and recover to initial capabilities after penetrations.
- Learn how to defend against new vulnerabilities to improve robustness over time.
Humans have Two Immune Systems: Innate and Adaptive

- Innate:
  - Neutrophils
  - Macrophages
  - Dendritic cells
  - Natural killer (NK) cells

- Adaptive:
  - CD4 T cell
  - CD8 T cell
  - CD3/CD4 T cell
  - CD19

At least 20 – 30% of the body’s resources are involved in constant surveillance and containment.

DISTRIBUTION STATEMENT A. Approved for public release; distribution is unlimited.

Slide courtesy Dr. Howard Shrobe, DARPA
Key Objectives

Demonstrate application centric adaptation for survival – make the “application” survivable and resilient against novel attacks

• An execution environment supporting innately and adaptively resilient applications
  – The protected application is harder to attack, harder to make unavailable, and harder to repeat past successful attacks
  – Isolation from other computation, dedicated to the survival of the protected application
  – Reusable, cost-effective defense near the application and part of defense in depth strategy
The A3 Vision: Integration of 3 Concepts

Containerization to isolate application execution

Mediated channels enables the defense to observe and control the application’s interaction with devices on its own terms

1. Crumple Zone enforces application specific preventive adaptation on container’s interaction through mediated channels

2. Replay with Modification on top of mediated containers to facilitate immunity-focused adaptation

3. Advanced State Management for containerized applications to enable various forms of restarts (recovery-focused adaptation)
What is a hard problem: Novel Attacks

- Behavior invariants (e.g., deployer provided constraint such as this web service should never make an outbound connection) or something more drastic (e.g., a segfault) indicates something went wrong
  - But the real attack likely happened in the past
  - Attacker has been successfully executing his tasks
  - And until now, we had no clue

- How deal with the aftermath of such attacks?

Observed by the CZ policies

Observed by the CZ policies

![Diagram](image)

- e.g., A is corrupt when \( f(x,y,z) = \text{true} \)
- e.g., rollback and restart, but to which past state?

Work toward immunity

RwM Experimentation
Crumple Zone: VM-based Realization

- Each container is essentially a DomU VM
- Channels are pathways from the application to devices (Disk, UI, Network)

Only the Xen hypervisor and Dom0 is treated as TCB

Crumple Zone(CZ) are VMs interposed on basic channels

A3 Conglomerate: the collection of VMs dedicated to the defense of a protected application
Replay With Modification: Motivation

• In a clean slate resilient and survivable host system context, it should be possible to
  – Reproduce application’s past execution
    • With different levels of fidelity and control in a repeatable manner
  – Explore alternate execution history
    • Alternate line leading to an immune conglomerate
    • Exploration of multiple lines unveiling details of novel attack faster

• RwM is A3’s contribution to address novel attacks
  – If an immune conglomerate is found, then that attack is ineffective
  – Provides an infrastructure as well as the collection of recorded information and supporting tools for analysts and cyber defenders to analyze a zero day attack and develop a countermeasure

• 2 levels of replay: Deterministic VM replay and Application Level
• Claim: synergistic combination is helpful in experiment-based failure diagnosis and patch identification
Multi-Compiler Variants: Utilizing A Diversity Generator

This is what is happening inside the diversity generator
Cyber security becomes an obvious context
Multiplicity?

Record and replay, experiment-based diagnosis, patching and recovery!
Use diversity generator to create polymorphic components that exhibit different vulnerability profile

Suddenly resources may not be that bountiful!
But wait—clouds are gathering steam!

Recorded information, Replay experiments, Diversity generation, Experiment-based diagnosis and patching all can potentially be done in the cloud!

But have we come full circle? Do we really trust the cloud with our critical data and computation?
Noise in Ciphertexts

- Ciphertexts are a combination of noise, the public key and a message.
- The public key is a combination of noise and the secret key.
- EvalMult operations “multiply” the noise in the ciphertext.
- Decryption operations strip away the noise.

Huge Amounts of Data and Computation Beget Special Purpose Solutions
Computation Flow On Untrusted Host

Untrusted host supports running of program on encrypted data.

FHE Operations filter down to appropriate FPGA, CPU or GPU implementation based on available resources.

Source Program

Translation of source program to circuit representation

Data Encrypted with FHE Scheme

Circuit Rep. of Program

Calls to FHE operations

FHE Operations
Encrypt, EvalAdd, EvalMult, Recrypt

Selection of calls to FPGA, CPU or GPU implementation for FHE Evaluations

- FPGA-based Lattice Crypto Primitives
  - Selection of FPGA circuits for lattice-based primitives
    - SIPHER FPGA Circuits
- CPU-Based Primitives
  - Selection of CPU libraries for lattice-based primitives
    - SIPHER CPU libraries
- GPU-Based Primitives
  - Selection of GPU libraries for lattice-based primitives
    - SIPHER GPU libraries
Asymmetric Operation Location Considerations

Operations Run on Trusted Host

Key Infrastructure
(Possibly On Client or Trusted Third Party)

KeyGen
Run on FPGA (*maybe* CPU)

Pk, Sk
Key Pair

Enc
Run on CPU, GPU or FPGA

Sk
Secret Key

Client

Plaintext
Final Output

Dec.
Run on CPU, GPU or FPGA

Final CryptoText

Program
Single Message
Circuit, program, formula, etc.
that may need translation.

Program Source

Enc(Pk,Sk)
Encrypted Secret Key

Untrusted Host

Evaluations and Recryptions
Directed by CPU
Run on FPGA

Program Translator
Run on CPU

Circuit Version of Program

Sensitive Plaintext Data

Enc
Run on CPU, GPU or FPGA

Programs Run on Untrusted Host

External Data Source(s)

CryptoText
Whew!

- The Big Bang (of Higher Performance Networked Diversity) Continues to Inflate
- Lot’s of Bottom Up Momentum Building across a number of planes to use that advancing Multiplicity
- Needs coupling with more Top Down concept-of-operation/theory weaving
- And Plenty More to Do to Keep Us Busy for a Long Time