An Agent-Supported Simulation Framework for Metric-Aware Dynamic Fidelity Modeling

Amy Castner
Ilya Chukhman, Ed Colbert, Markus Dale, Bessie Lewis, David Zaret
amy.castner@jhuapl.edu

Agent-Directed Simulation
Spring Simulation MultiConference 2007

APL
The Johns Hopkins University
APPLIED PHYSICS LABORATORY

Motivation

- **Information assurance (IA) seeks to protect the confidentiality, integrity and availability of data**
- **Quantitative analysis allows us to:**
  - **Compare systems before they are deployed**
  - **Gauge a technology's effectiveness before it is built**
  - **Understand trade-offs when modifying a system**
- **Analyzing IA in large complex computer networks is difficult**
  - **Large networks require scalability**
  - **Intricacies of attacks and defenses require high fidelity**

U.S. Department of Defense Information Assurance emblem

What is Fidelity?

"The accuracy of the representation when compared to the real world"*
Behavioral – Spatial – Temporal

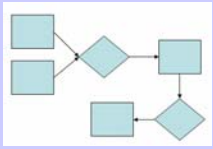
Behavioral Fidelity Example:
A Rate Limiter Running on a Router

Lower Fidelity → **Higher Fidelity**

Closed-Form Equations

Use a probability function to determine whether each packet is passed or dropped

Algorithms



Execute the rate limiting algorithm to monitor traffic and decide whether packets are passed or dropped

Models

Simulate passage of each packet through a network stack that includes the rate limiter

Testbed Systems

Run the rate limiting software on a router in the lab

*Defense Modeling and Simulation Office (DMSO) Fidelity Special Topic, 9/30/00.

APL

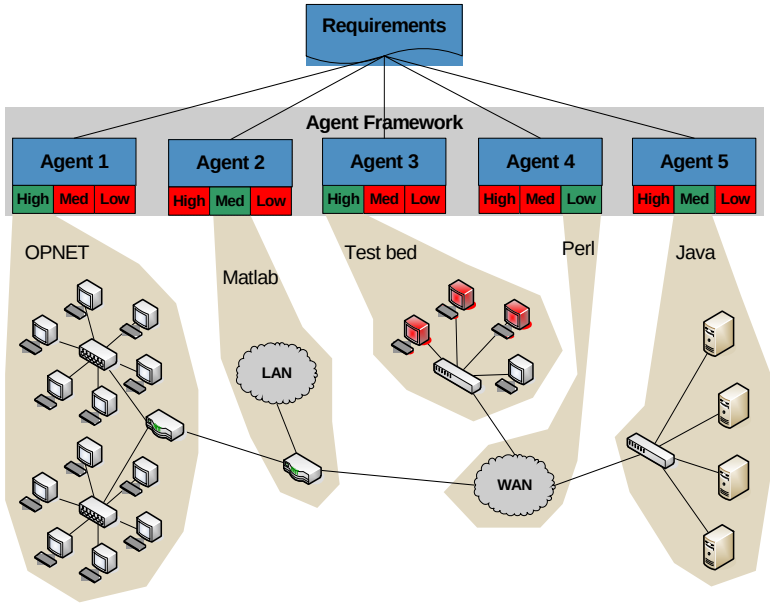
A New Approach

What fidelity do we need to answer a specific analysis question?

Build a M&S framework that *selects fidelity consistent with analysis goals*.
Use autonomous software agents to manage distributed variable-fidelity components of a network simulation.

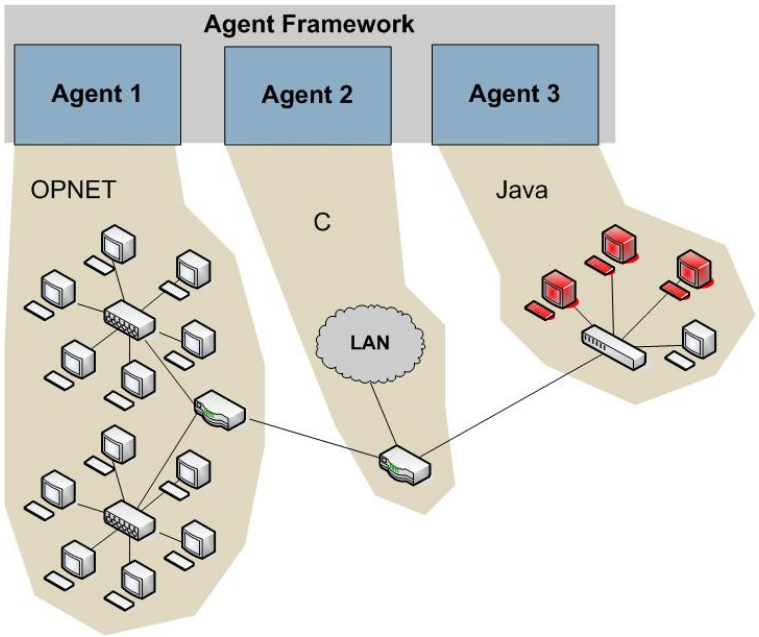
MADFM Concept

- The analyst:
  - Identifies analysis requirements
  - Divides conceptual scenario into logical segments
- Agents within the MADFM framework:
  - Oversee simulation of each segment
  - Possess multiple representations of each segment
  - Interpret requirements and select representations with the appropriate fidelity
  - As requirements change, dynamically vary segment fidelity by replacing one representation with another



Current Scope

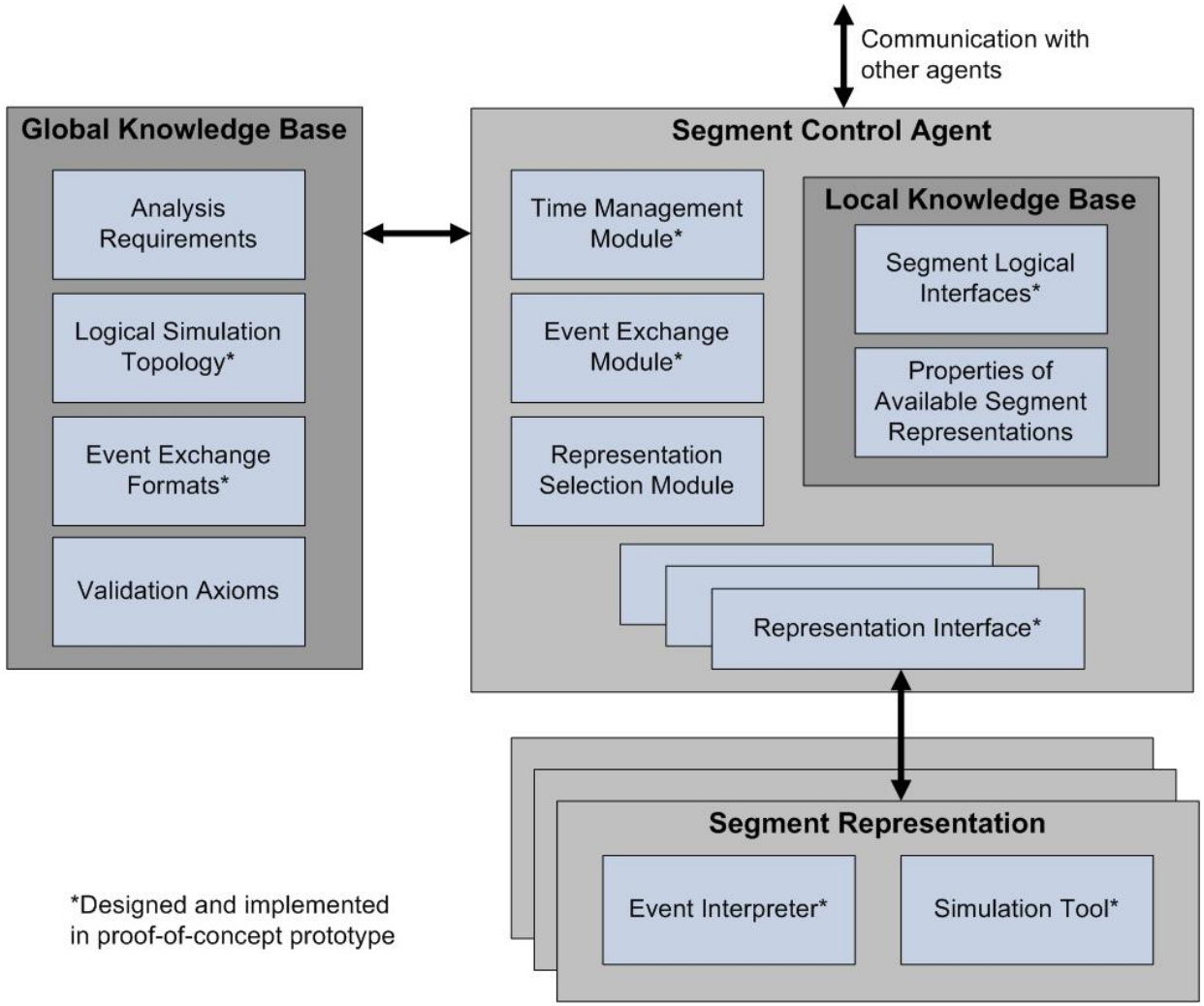
- **Build a multiple fidelity simulation framework**
  - **Use an agent-supported framework to interface diverse simulation environments**
- **Later, augment this framework to add advanced capabilities**
  - **Metric Awareness: Allow the agents to select which representation to run**
  - **Dynamic Fidelity: Enable agents to swap representations during the simulation**



Multiple Fidelity Simulation Framework Requirements

- **Interface diverse simulations, including discrete-event, continuous (e.g., testbeds), and static (e.g., mathematical equations) representations**
- **Enable distribution of component simulations across multiple operating systems in a networked environment**
- **Provide time management**
- **Provide a mechanism for event exchange among simulations**
- **Facilitate the future addition of metric awareness and dynamic fidelity capabilities**

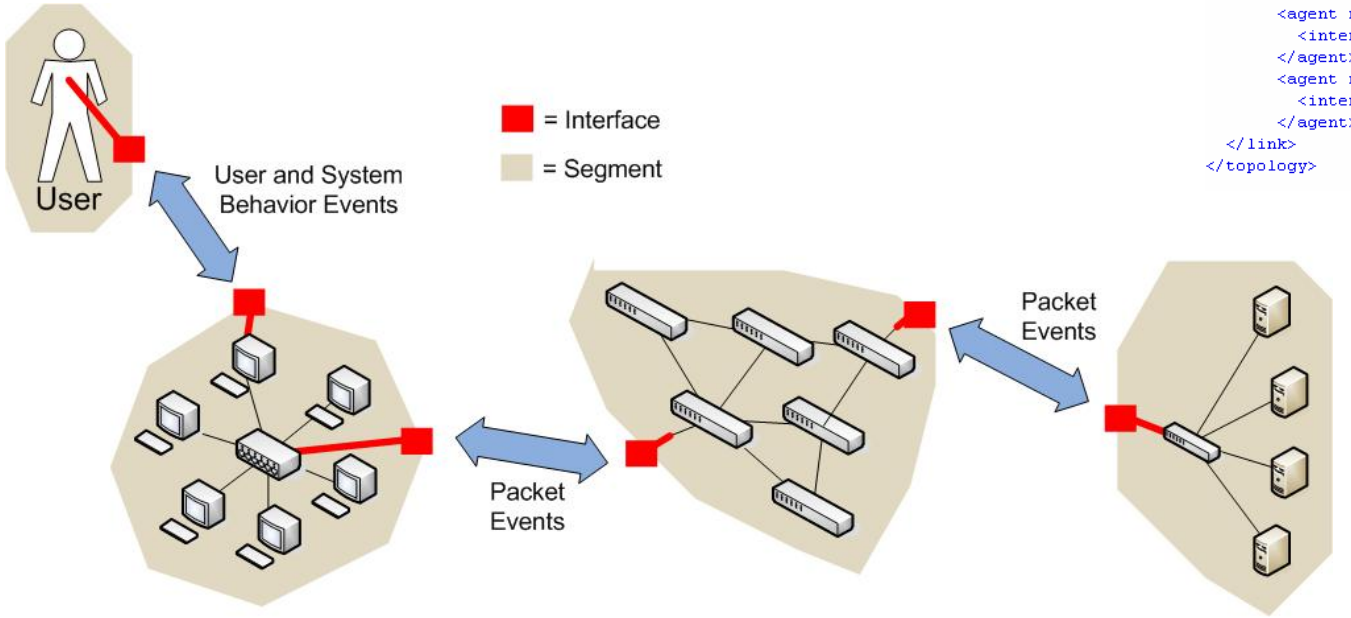
System Architecture



Event Exchange

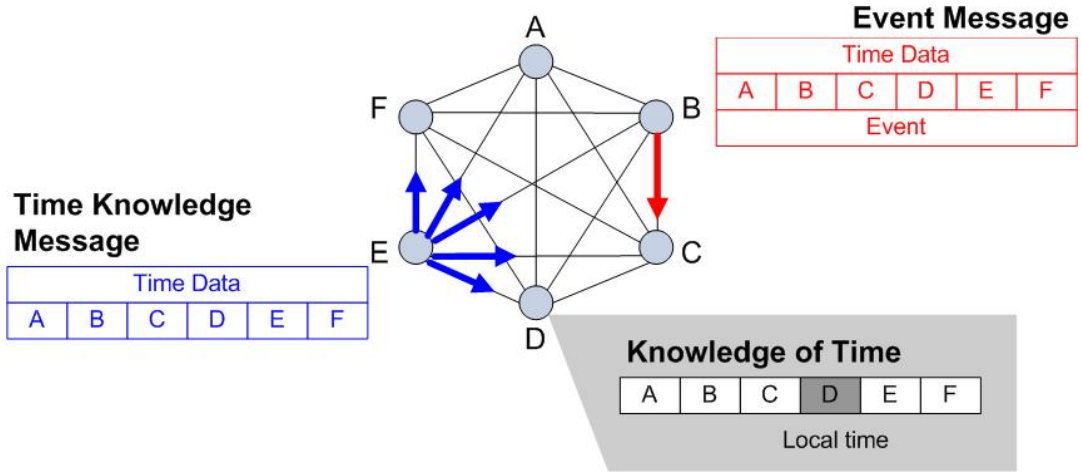
- **Agents exchange events across interfaces**
- **Use XML to map incoming and outgoing interfaces at simulation boundaries**
- **Framework supports event definition and exchange**

```
<topology>
  <link>
     <agent name="agent1">
       <interface name="1"/>
     </agent>
     <agent name="agent2">
       <interface name="3"/>
     </agent>
  </link>
  <link>
       <agent name="agent1">
         <interface name="2"/>
       </agent>
       <agent name="agent3">
         <interface name="3"/>
       </agent>
  </link>
  <link>
       <agent name="agent2">
         <interface name="1"/>
       </agent>
       <agent name="agent3">
         <interface name="1"/>
       </agent>
  </link>
</topology>
```

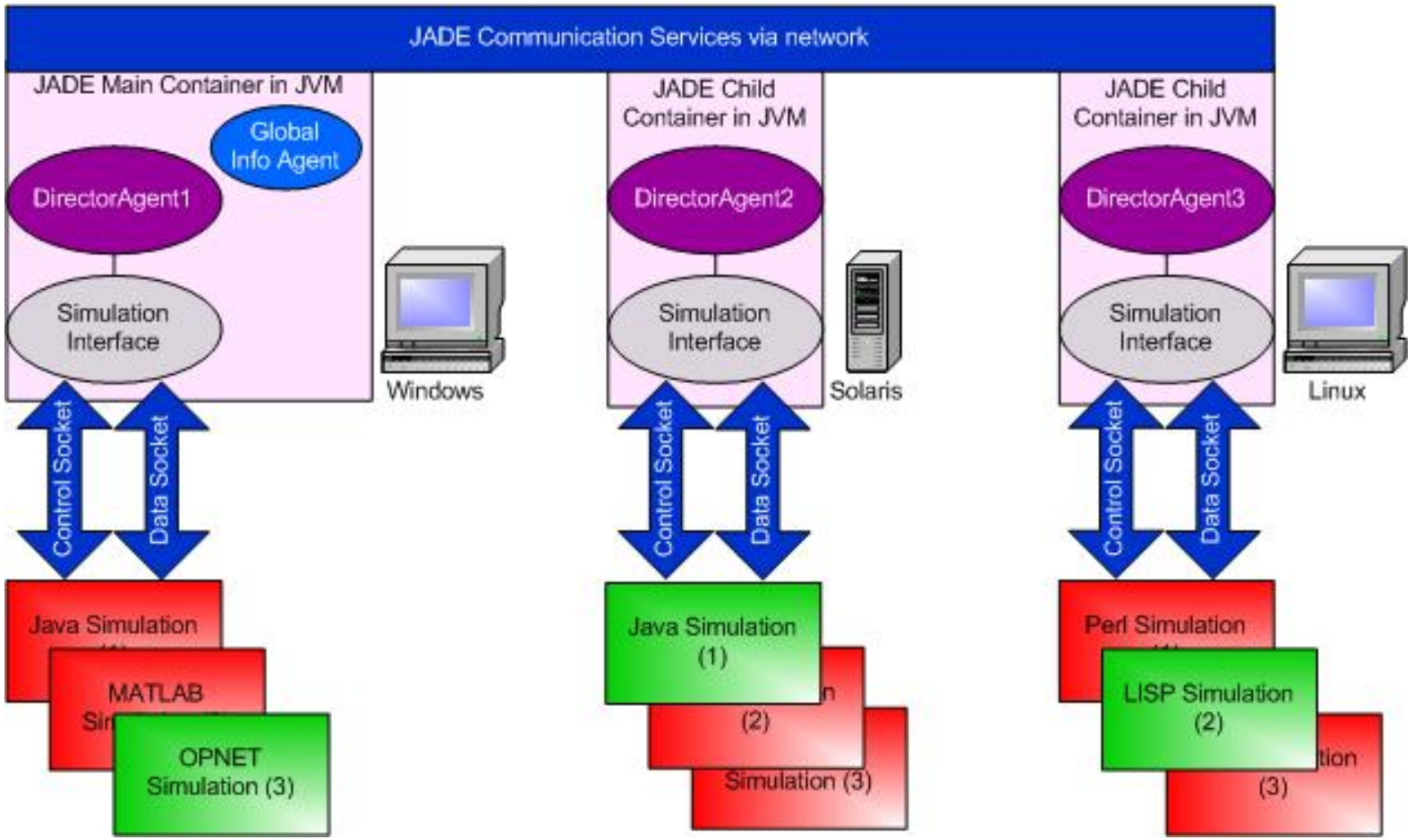


Time Management

- Framework contains time management module
  - Selected based on simulation requirements and representation capabilities
- Present implementation uses a best effort algorithm
  - Avoid rollback operations and look-ahead analysis of event queues
  - Each agent shares its knowledge of time in all segments at intervals of its own simulation time and when exchanging events
  - Agents cannot allow their simulation time to exceed the global minimum time by more than the configured threshold

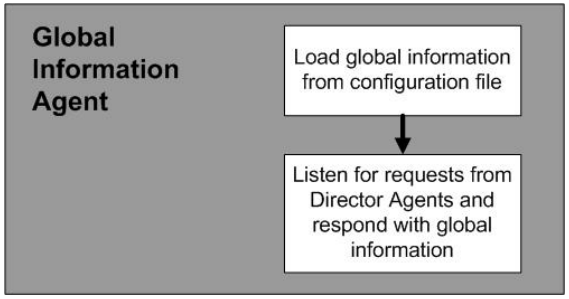


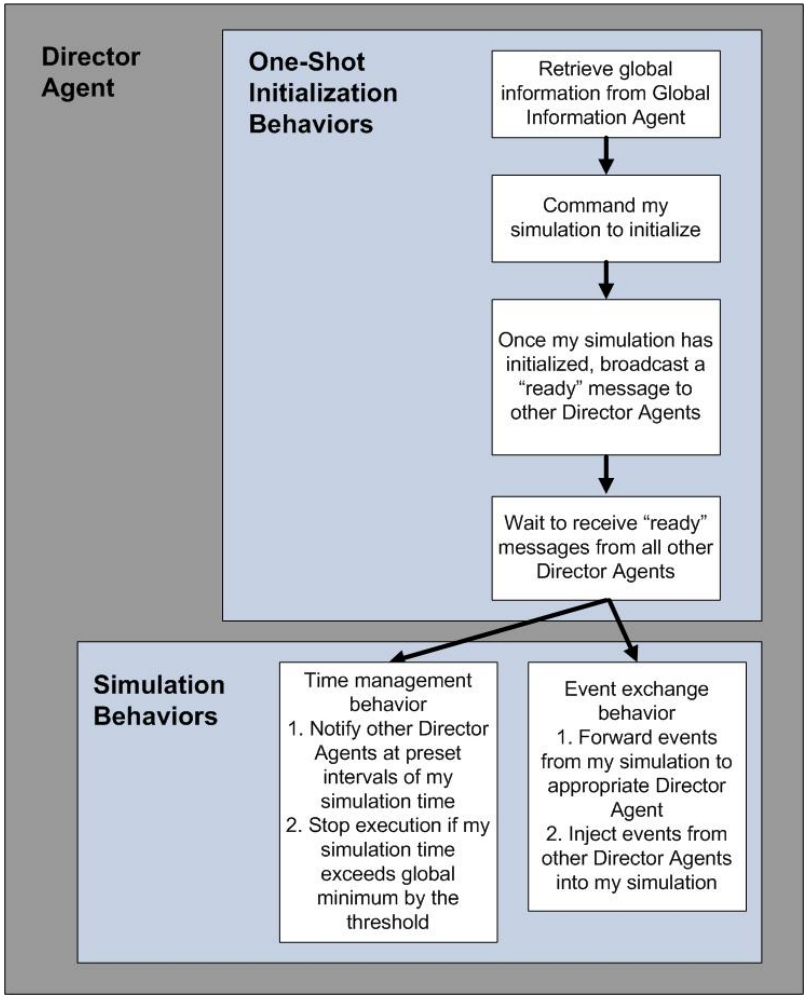
Implementation



Agents and Behaviors

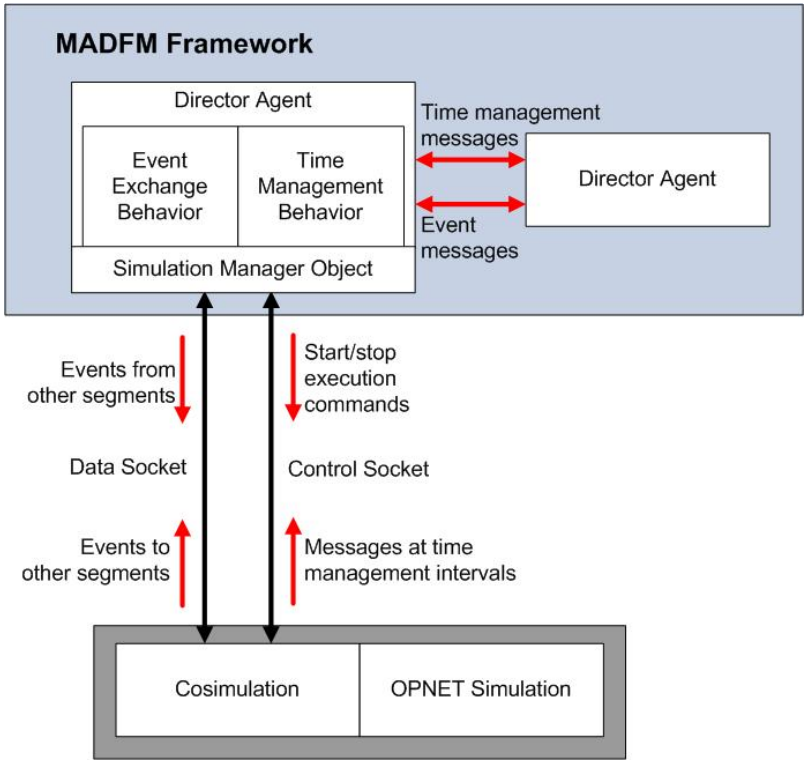
- An executing instance of MADFM contains:
  - One Global Information Agent
  - One Director Agent for each simulation segment
- A JADE agent performs tasks specified by its behaviors
  - Behaviors can be added to an agent during execution





Interfacing with Simulations

- The Director Agent uses a Simulation Manager object to interface with a simulation tool
- Implemented Simulation Manager for OPNET Modeler
  - Uses OPNET Cosimulation API
    - Controls OPNET execution
    - Shares a memory buffer with OPNET
  - Monitors local OPNET simulation time and notifies Director Agent at intervals
  - Works with custom-built OPNET process model to handle packet event exchange

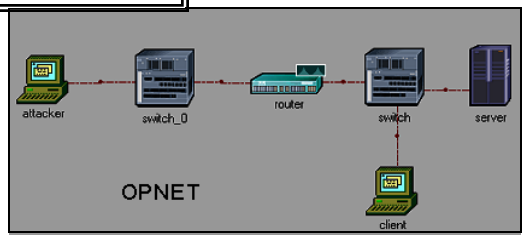


Validation

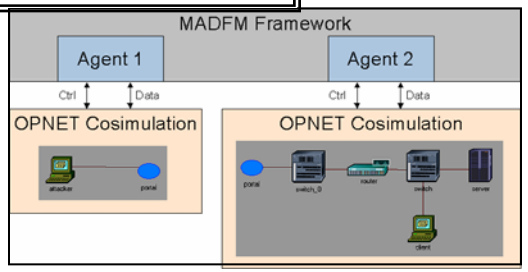
- **Test scenario**
  - **Attacker SYN floods target server for 400 seconds**
  - **Client attempts to connect to server every 5 seconds**
  - **Analysis goal: Calculate the client's Probability of Denied Service (PDS) during the attack**
- **Collected three sets of data**
  - **Control: Previously validated OPNET model**
  - **Case 1: OPNET attacker simulation with an OPNET target network simulation**
  - **Case 2: Java SYN flood generator with an OPNET target network simulation**

$$PDS = \frac{\text{\# TCP Handshakes}}{\text{\# TCP Handshake Attempts}}$$

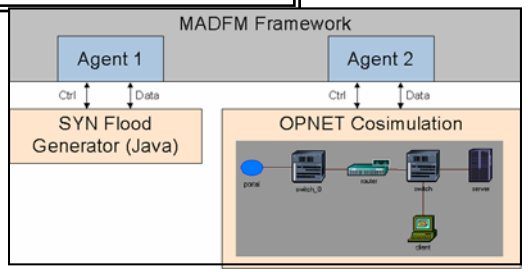
Control



MADFM Case 1



MADFM Case 2



Validation Results

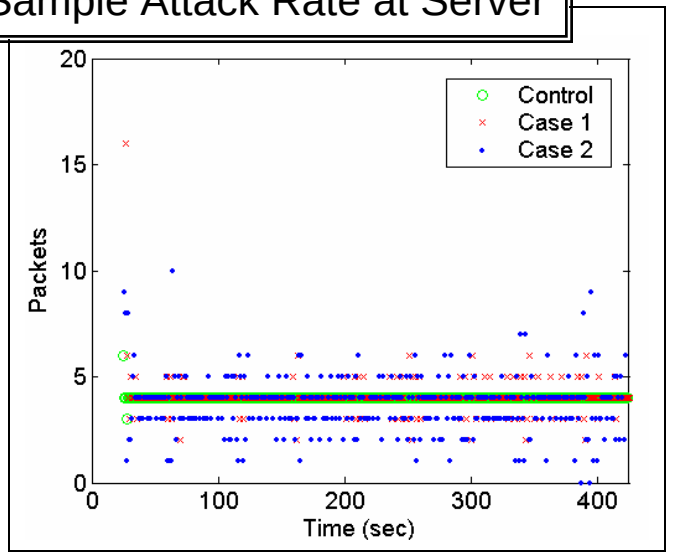
- Data collected from all three cases is very similar
  - Analysis yields similar average PDS values
  - Some jitter observed in attack traffic pattern in MADFM cases
    - This can be removed by tuning the time management algorithm

➢ **Increasing the fidelity of the attack traffic pattern is not necessary to collect valid results**

PDS Metric Calculations

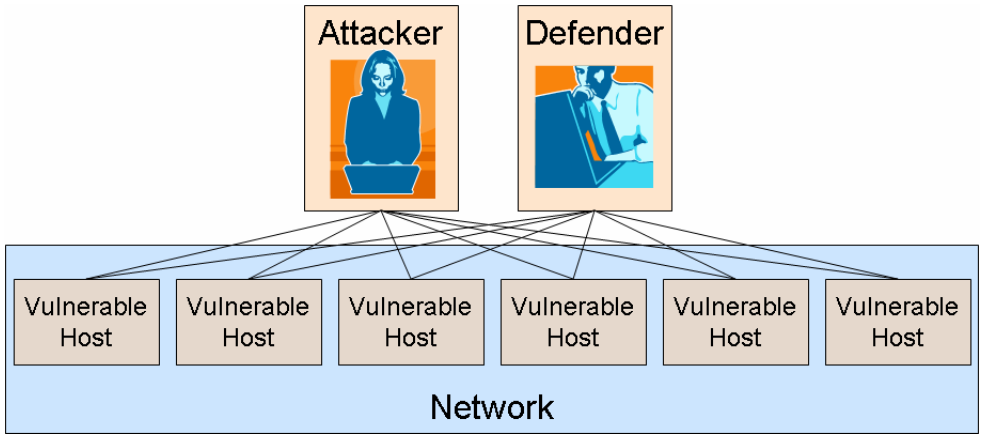
	Runs	Average PDS	Standard Deviation
Control	10	0.44	0
MADFM Case 1	10	0.45	0.01
MADFM Case 2	10	0.41	0.02

Sample Attack Rate at Server



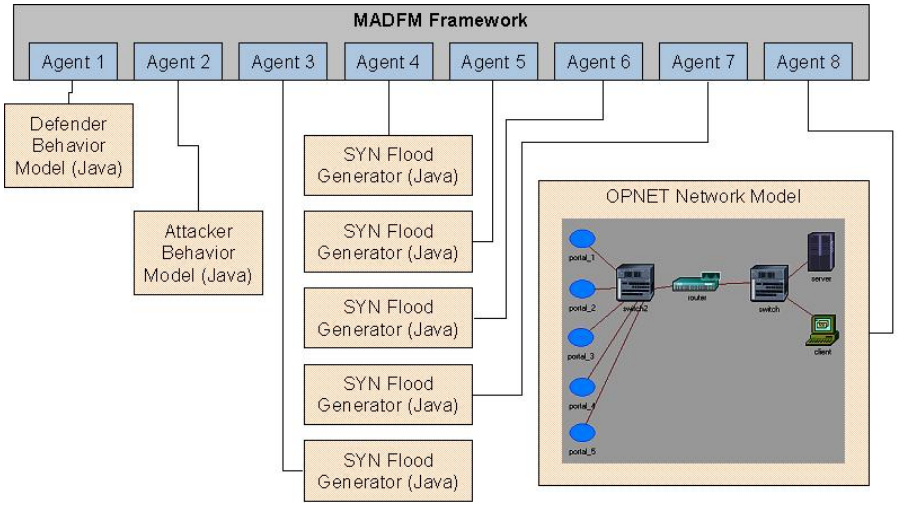
Sample Results: Conceptual Analysis Scenario

- **Goal: Measure the Probability of Denied Service (PDS) and link utilization as an Attacker enables drones in a distributed denial of service attack and a Defender works to disable them**
- **Setup:**
  - **Periodically, the Attacker selects a host and attempts to exploit it. If successful, the Attacker enables a SYN flood from that host to the target server.**
  - **Periodically, the Defender selects a target and sweeps it for malware. Any SYN floods running on the host are disabled.**



Sample Results: Network Segments

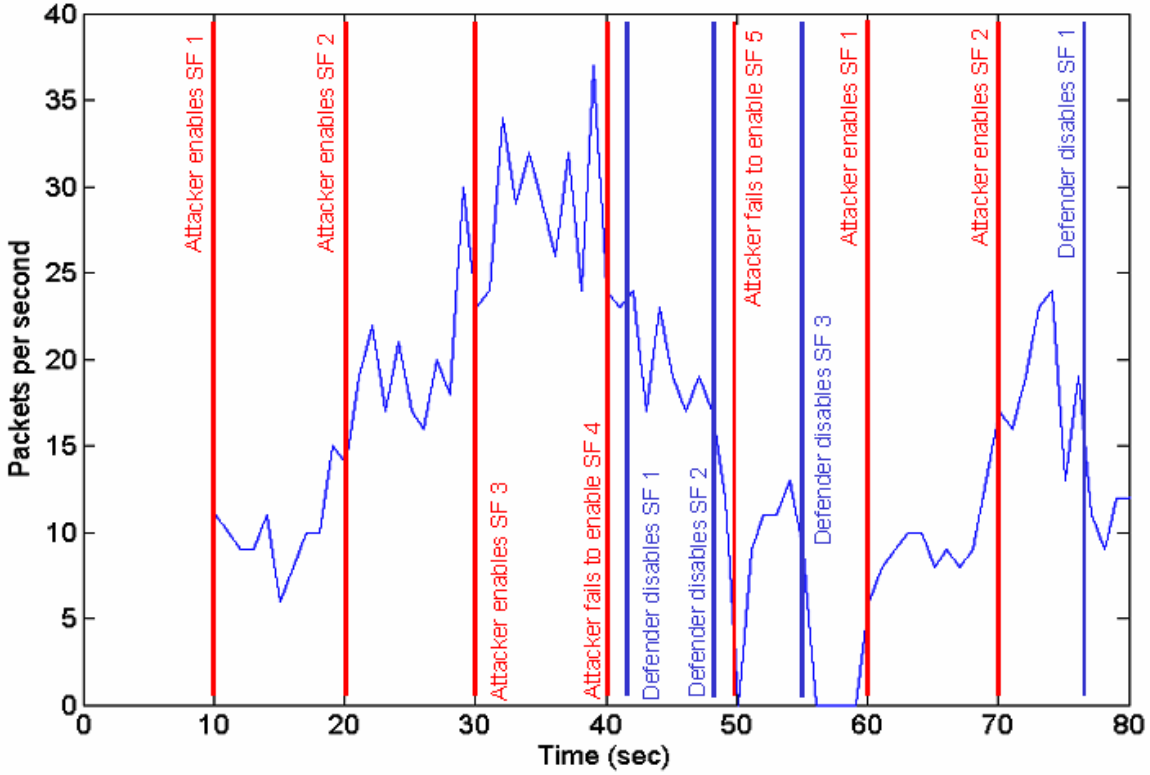
- OPNET network model
  - Represents the target network, including server and routing infrastructure
- Java SYN flood models
  - Generates attack traffic when enabled



- Attacker behavior model
  - Cycles through list of hosts, performs trial that represents attempt to gain access, and starts SYN flood if successful
  - May augment with more complex behavior models or attempting exploit in a testbed
- Defender behavior model
  - Cycles through list of hosts, stopping SYN floods

Sample Results

- Implemented in MADFM multiple fidelity simulation framework
- Behavior models accurately affect the traffic observed in OPNET



Ongoing Work

- Developing metric awareness capability
  - Mechanisms to describe simulation requirements, components, and specify knowledge base
  - Algorithms for agent decision-making (e.g., rule-based systems)
  - Tools for use with implemented MADFM agents (e.g., Jess Rule Engine, Protégé Knowledge Editor)
- Integrating additional simulation tools (e.g., ns-2)
- Scalability and performance testing

Questions?

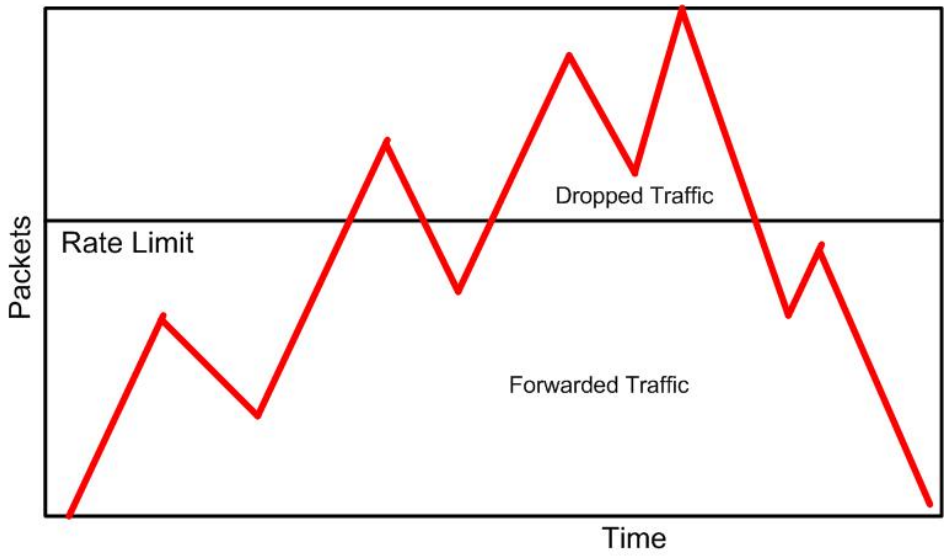
APL

The Johns Hopkins University
APPLIED PHYSICS LABORATORY

Backups

Rate Limiting

- A rate limiter controls the rate of traffic sent or received on a network interface
- This rate is typically measured in packets per unit time
- Traffic that exceeds the specified rate is dropped
- Rate limiters can run on routers



Benefits of Agent Architecture

- Distributed processing
  - Event processing may be done autonomously
  - Bridging gaps between low and high fidelity models will likely be intensive
  - Metric awareness and fidelity decisions
- Control architecture flexibility
  - Easily swap out time management algorithms
  - Make agents as cooperative or independent as required
  - Facilitates addition of new control components
- Facilitates addition of new functionality
  - Observer agent: May aggregate data as the simulation runs by receiving messages from other agents
  - Model checking agent: Verify model during execution