



JOCosim: Using Java, OPNET, & C to Enable Mobile Co-Simulation

Chris Augeri¹, Kevin Morris², & Barry Mullins¹

Air Force Inst. of Tech. (AFIT)¹ & Air Force Comm. Agency (AFCA)²



Copyright © 2006 OPNET Technologies, Inc.

CONFIDENTIAL - RESTRICTED ACCESS: This information may not be disclosed, copied, or transmitted in any format without the prior written consent of OPNET Technologies, Inc.
Used with permission of the Author.

§ HARVEST

- § Conceptual UAV swarm

- § Policy management vs. cross-layer design

§ JOCosim I (JOC-I)

- § Java data exchange

- § Single UAV

§ JOCosim II (JOC-II)

- § Java control

- § Multiple UAVs

HARVEST

Host of Armed Reconnaisance Vehicles

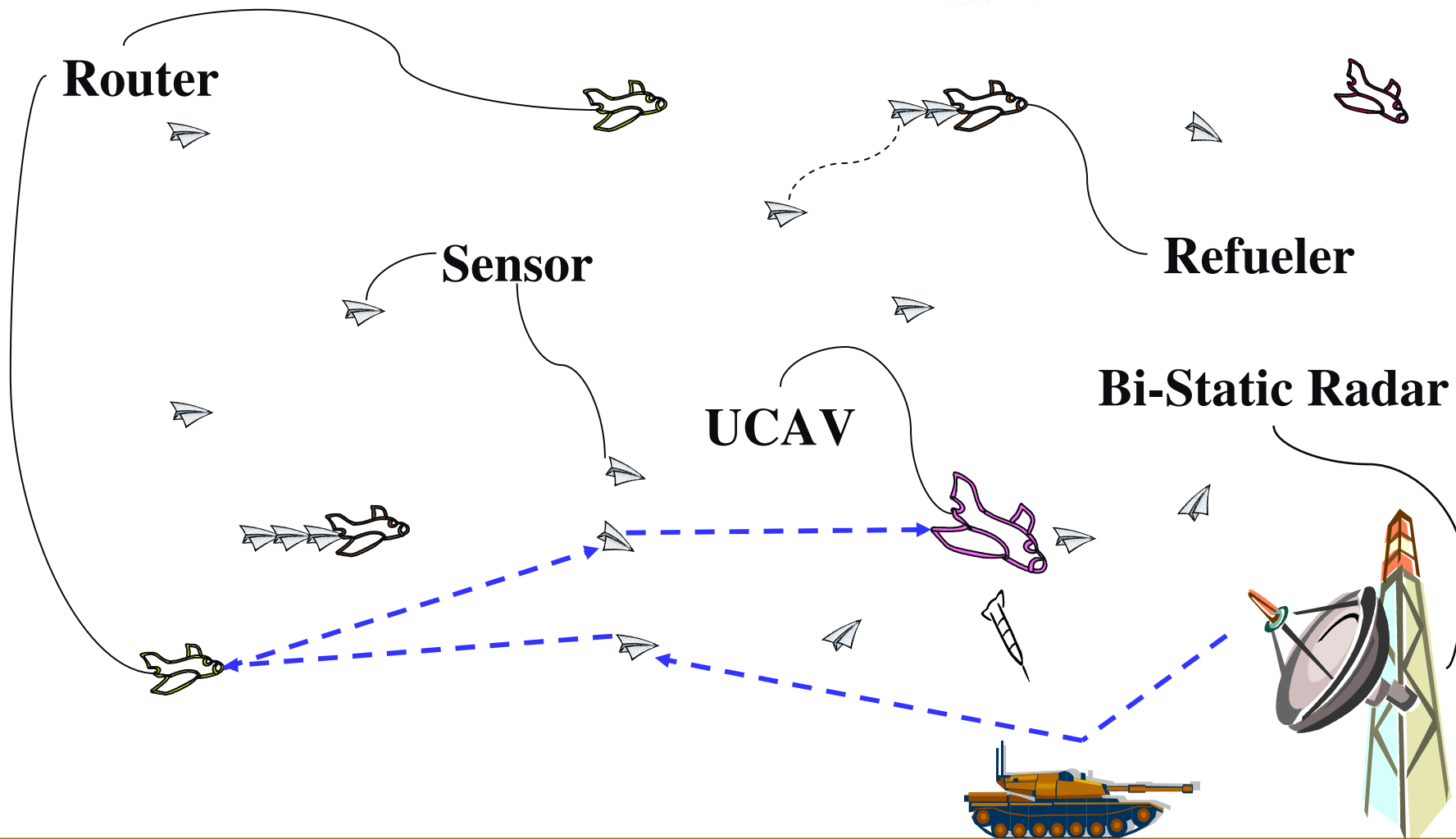
Enabling Surveillance & Targeting

HARVEST: A Prototypical UAV Swarm



- § **What it is:** cooperative UAV swarm concept
- § **What it is *not*:** architecture for UAV design, e.g., JAUS
- § **Configurations**
 - § A micro-UAV assigned to each member of a platoon
 - § A small UAV assigned each security forces element
 - § A deployable attack/recon asset
- § **Applications**
 - § Troop reconnaissance or anti-sniper support
 - § High-threat environments
 - § Active target tracking

HARVEST: A Conceptual Deployment



HARVEST: A Representative Deployment



§ Configuration

- § (4) Mobility Domains
- § (4) GCSs
- § (4) UAVs

§ Operation

- § GCS communicates w/ vertically/horizontally adjacent GCSs, but not diagonally adjacent
- § UAV assigned to GCS controlling domain
- § UAV flies using random waypoint model



§ Cross-Layer Design

§ Integrate vertically...

§ Pros: optimization

§ Cons: testing, upgrading

§ Management Policies

§ Command vertically

§ Pros: maintainable, scalable

§ Cons: speed?

§ (2) policies (planes)

§ Preservation

§ Employment

Layers	Services	Policies
Application	User Interaction Swarm Services Vehicle Services	Preserve fuel defense
Transport	IPv6	Employ recon target
Network	IPv6	
Link	+Timing	

References

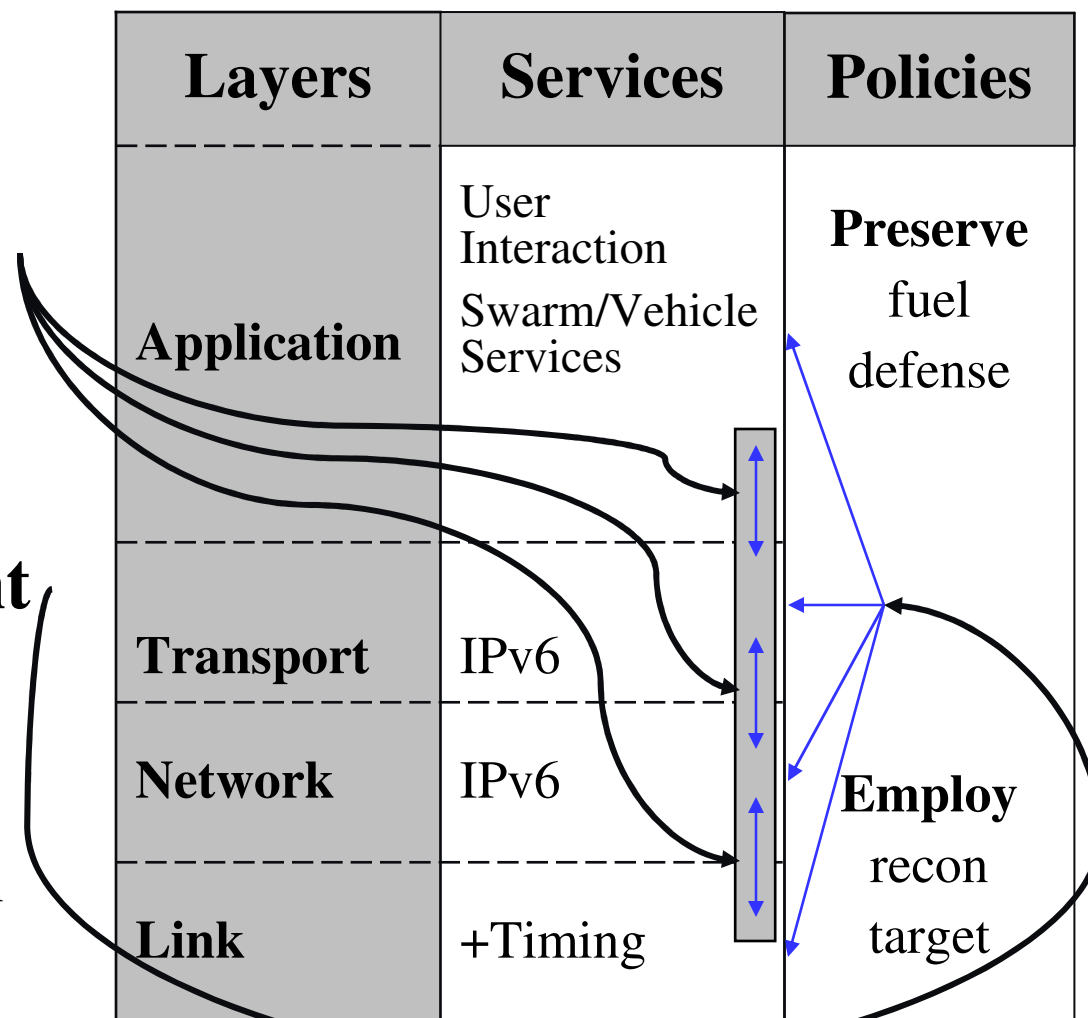
I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. "A survey on sensor networks," IEEE Comm., vol. 40 (8),2002, pp. 102-114.

§ Localization

- § Link-layer timing
- § Network & transport layer forward timing
- § Application-layer Localization algorithm consumes timing data

§ Power Management

- § Each layer/protocol provides API
- § Each protocol adjusts consumption based on UAV status, swarm intel, and user inputs



HARVEST: Application Layer



Swarm Services

Aerial
Docking

Data
Indexing

Target
Tracking

Cooperative
Search

Vehicle Services

Collision
Avoidance

Payload
Tasking

Telemetry
Reporting

Information
Sensing

User Interaction

Swarm
Management

Flight
Planning

Query
Generation

Munitions
Deployment

§ What is co-simulation?

- § Integration of multiple simulation engines
- § Leverages capabilities or investment in multiple simulators

§ Co-simulation goals

- § Wireless performance (OPNET)
- § Application-layer protocols (Java)
- § Aerial behavior (DTED, flight simulator)

§ Co-simulation interfaces?

- § OPNET: ESI (generic co-simulation interface)
- § OPNET: HLA (from DoD DMSO)

§ Java

- § Java Native Interface (JNI)
- § Threads & Networking

§ OPNET

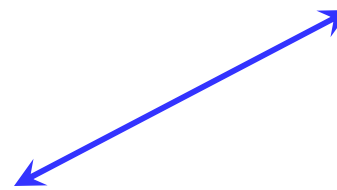
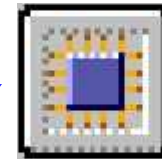
- § External Model Access (EMA)
- § External System Definition (ESD)

- § External System Interface (ESI)
 - § External System Access (ESA, from external)
 - § External System Calls (ESYS, to external)
- § System Definition (SD)

§ C

- § Common language
- § Enables both phases of JOCosim

External System Model
(ESM, aka CPU)

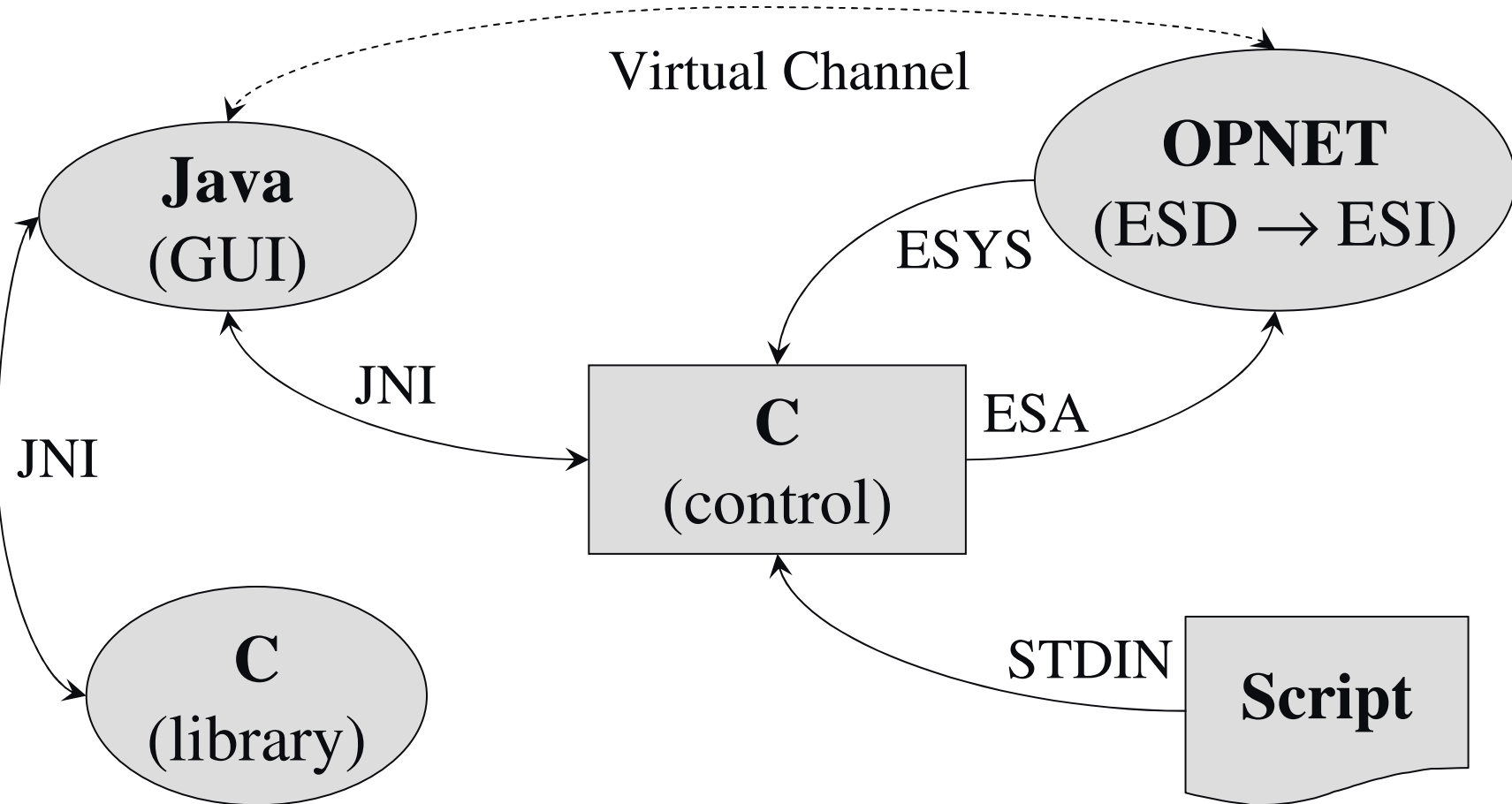


JOCosim I

(JOC-I)

- § **JOCosim:** Java, OPNET, and C Mobile Co-Simulator
- § **Concept:** Enable independent and parallel research about UAV swarms, e.g., HARVEST
- § **Motivation:** To see if it could be done...
- § **How:** Phased development
 - § JOC-I: verify Java can exchange data w/OPNET
 - § JOC-II: verify Java can control an OPNET simulation

JOC-I: Design Overview



JOC-I: Functional Decomposition



Node Model (*manet_station_adv_jocosim*)

External System (ESYS) Process Module (*op_esys_interface_**)

External System (ES)

ESD

ESI

esd_Duration
esd_xPsn...

SD

platform: windows
bind_objs: JOCosim.obj
use_esa_main: no

Controller (C)

Esa_Main/Init/Load
Esa_Interface_Group_Get

JNI_CreateJavaVM
GetStaticMethodID
CallStaticVoidMethod

GUI (Java)

xInit, xUpdateSpeed, xUpdateBearing

Library (C)

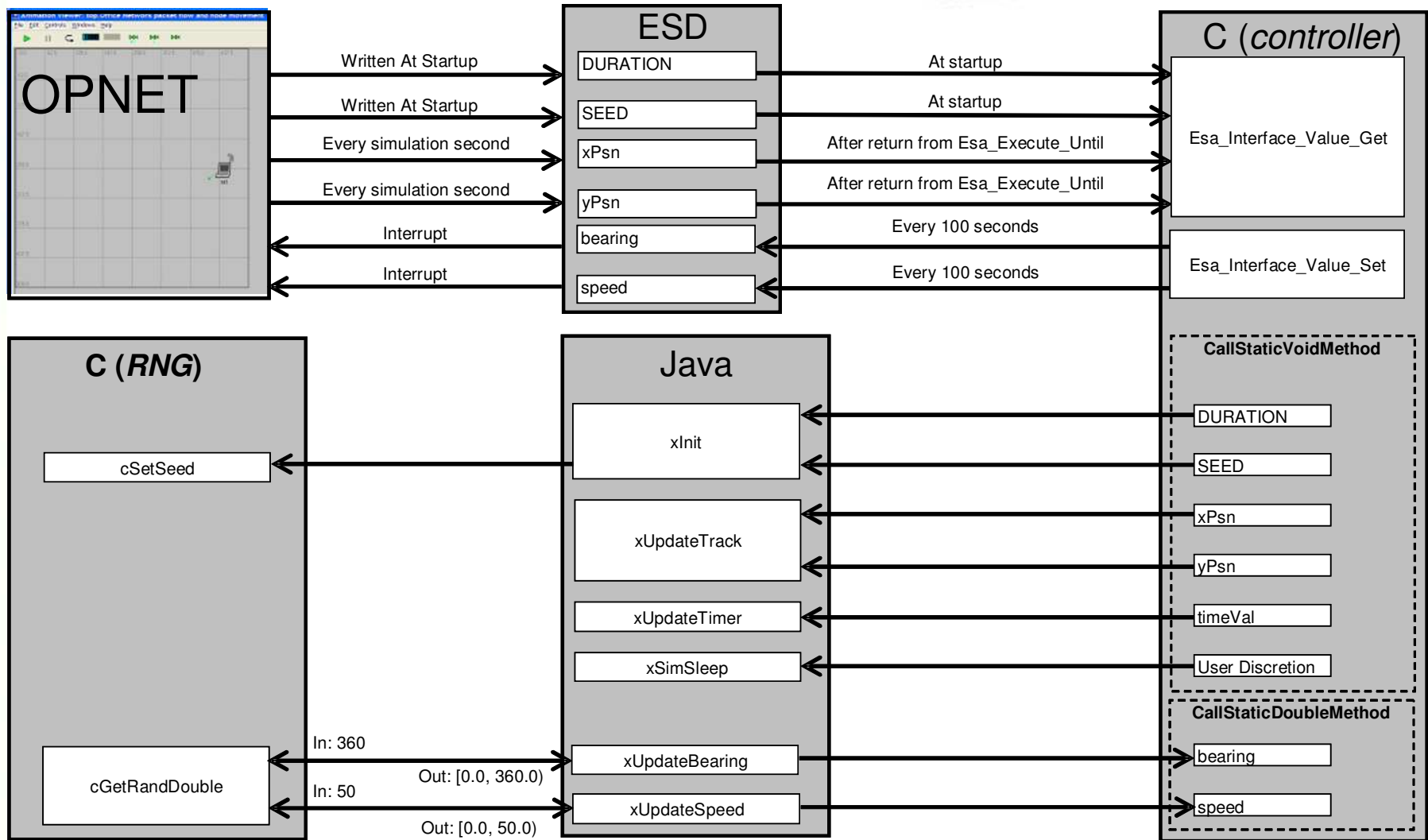
cGetRandDouble, cSetSeed

JOC-I: External System Definition (ESD)



Field	Type	Data Flow	When?
esd_DURATION	double	Java ← OPNET	@ initialization
esd_SEED	integer	Java ← OPNET	@ initialization
esd_xPsn	double	Java ← OPNET	1/sec, for GUI
esd_yPsn	double	Java ← OPNET	1/sec, for GUI
esd_speed	double	Java → OPNET	every 100 secs
esd_bearing	double	Java → OPNET	every 100 secs

JOC-I: Data Flow Diagram (DFD)



JOC-I: Calling Java from C



Java: Define Method

```
public static void xSimSleep(int ms)...
```

Java: Compile & Extract Signatures

```
javac callbacks
```

```
javap -s callbacks > callbacks.s
```

```
type callbacks.s
```

```
public static void xSimSleep(int); // "(I)V"
```

C: Find Method (use signature from callbacks.s)

```
jMIDs[JVM_SLEEP] = (*jEnv)-> GetStaticMethodID  
(jEnv, jCls, "xSimSleep", "(I)V");
```

C: Call Method

```
(*jEnv)-> CallStaticVoidMethod  
(jEnv, jCls, jMIDs[JVM_SLEEP], 5000);
```

JOC-I: Calling C from Java (1 of 2)



Java: Define Method Stub

```
private static native void cSetSeed(int seed);
```

Java: Call Method

```
int s = 97;  
cSetSeed(s);
```

Java: Compile & Extract C Header

```
javac callbacks  
javah callbacks  
type callbacks.h  
JNIEXPORT void JNICALL Java_Callbacks_cSetSeed  
    (JNIEnv *, jclass, jint);
```

JOC-I: Calling C from Java (2 of 2)



C: Define Method (from callbacks.h)

```
#include <callbacks.h>
JNIEXPORT void JNICALL
    Java_Callbacks_cSetSeed
    (JNIEnv *, jclass, jint);
```

C: Compile Dynamic Link Library (DLL) — all one line

```
CL /LD /MD /I.
    /I"C:\Progra~1\opnet\10.5.A\sys\include"
    /I"C:\Progra~1\Java\jdk1.5.0_06\include"
    /I"C:\Progra~1\Java\jdk1.5.0_06\include\win32
    " jocosim_CoSimOPNET.c opsim.lib /link
    /LIBPATH:C:\Progra~1\OPNET\10.5.A\sys\pc_inte
    l_win32\lib
```

JOC-I: Co-Simulation Results



§ Links Java/OPNET

§ Script control (C)

§ Uses ESI (C ↔ Java)

§ Single UAV

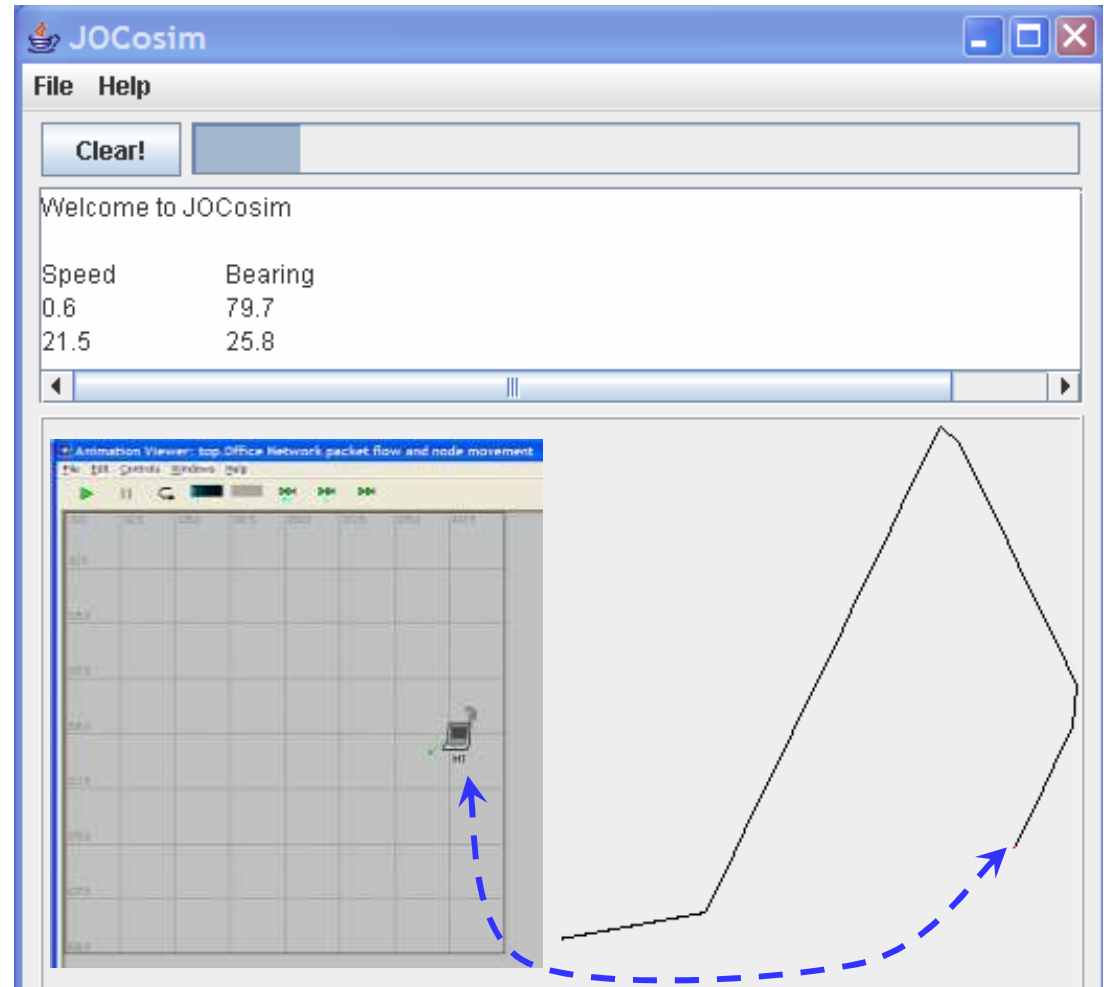
§ Position → Java

§ Vector → OPNET

§ GUI

§ Single UAV tracker

§ Verify using animation viewer



JOCosim II

(JOC-II)

JOC-II: Extensions to JOC-I



1. Move co-simulation control to Java
2. Develop interface cloud for ESYS access
3. Create models w/external model access (EMA)
4. Enable broadcast & routed transmissions (modes)
5. Support node failure and recovery (via power mgmt)
6. *Active statistic collections via probes (global)
7. *Port a node mobility algorithm to Java
8. *Integrate a data indexing app
9. *Migrate from OPNET 10.5 to 11.5

* partially completed extensions

JOC-II: Simulation Creation Dialog Box



- § User interface to create a simulation
- § GUI tailored for cooperative search
- § Network created w/EMA (JOCnet)
 - § Network size
 - § Search grid
 - § UAVs & cloud
 - § ESD

The screenshot shows a dialog box titled "Create simulation..." with a close button (X) in the top right corner. The dialog has two tabs: "Basic" (selected) and "Advanced".

Basic Tab:

- Model Name: JOC_5_100_100
- Scenario Name: JOC_5_100_100_10_10_600_3
- # Observers: 5
- Network Width (m): 100
- Network Height (m): 100
- Cell width (m): 10
- Cell Height (m): 10
- Time (sec): 600
- Random Seed: 3
- Save Animation
- View Animation
- Probe Model: Global
- Use Defaults
- Create Scenario!
- Run Scenario!

JOC-II: Animation Viewer



§ Interface cloud

§ UAVs

§ Active

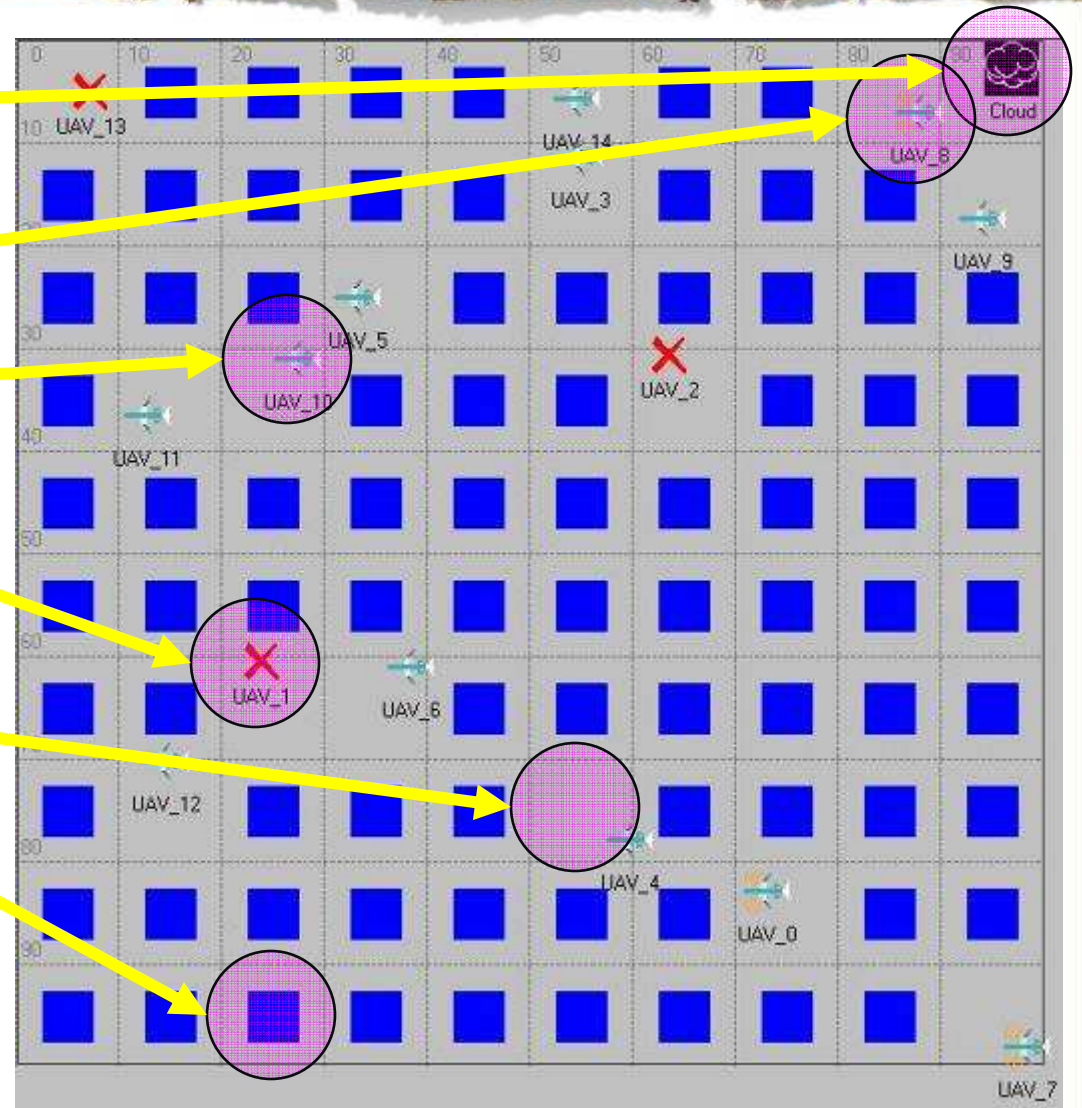
§ Reserve

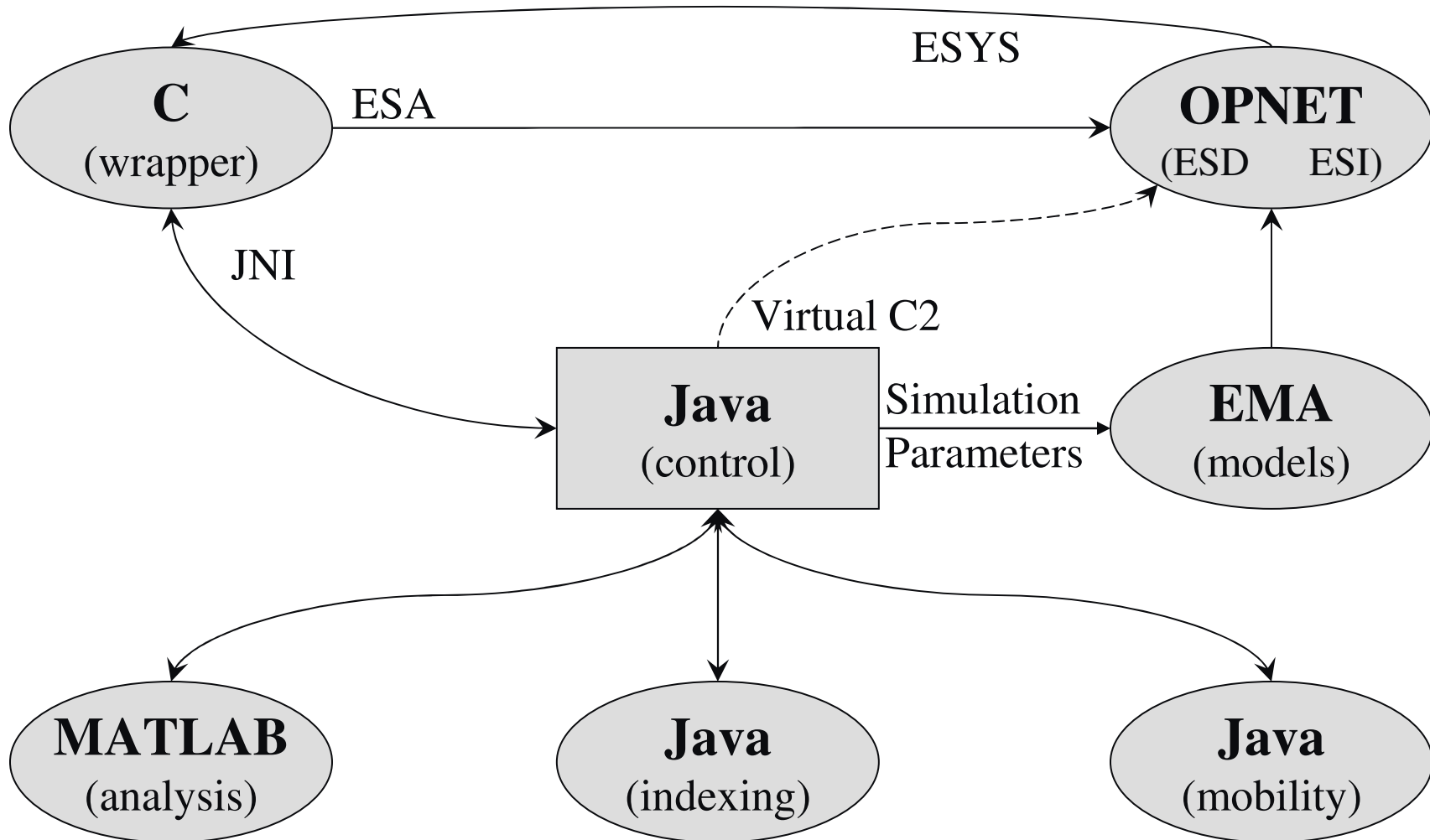
§ Destroyed

§ Search Areas

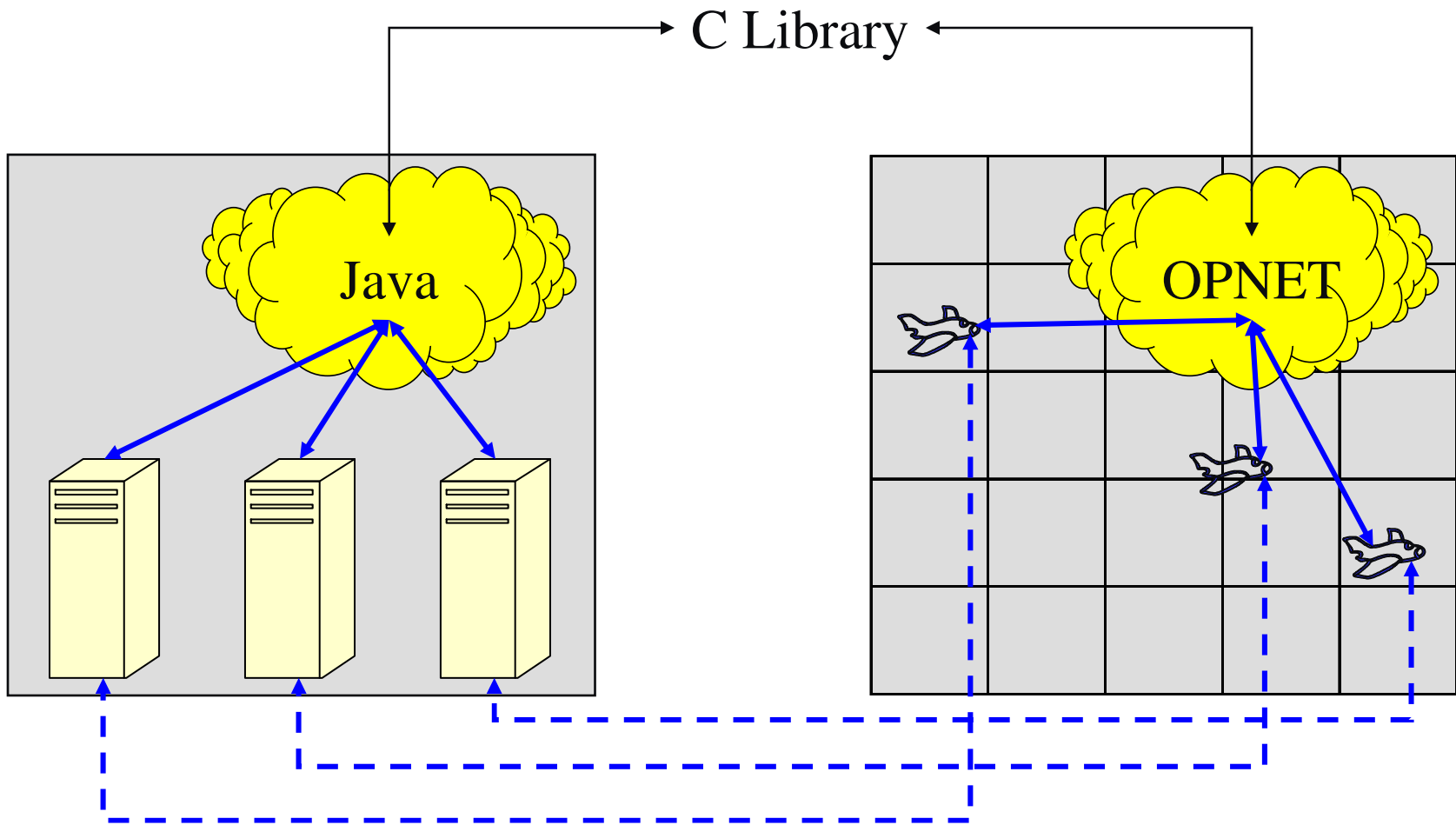
§ Searched

§ Not searched





JOC-II: Conceptual Data Flow



JOC-II: External System Definition (ESD)



- § Parallel arrays use synchronized timing control
- § Inbound/outbound packet channels are asynchronous
- § Used assess utility/issues of both control mechanisms

Usage	Field Name	Data Type	Data Flow	Size
Parallel arrays, 1 / UAV & cloud	esd_status	integer	Java → OPNET	N+1
	esd_psnX	double	Java → OPNET	N+1
	esd_psnY	double	Java → OPNET	N+1
Ctrl = -1: Free Ctrl = 0: Hold Ctrl = +1: Data	esd_inCtrl	integer	Java ↔ OPNET	1
	esd_inPkt	character	Java → OPNET	4000
	esd_outCtrl	integer	Java ↔ OPNET	1
	esd_outPkt	character	Java ← OPNET	4000

- § 4,000 bytes
- § Used in all communication modes
- § Multiple payload packets, currently only one defined

Network Source (4)	Network Sink (4)	
Application Source (4)	Application Sink (4)	
Payload Length (4)	Payload Type (1)	Reserved (3)
Payload (3976)		

JOC-II: Communication Modes (1 of 3) Overview

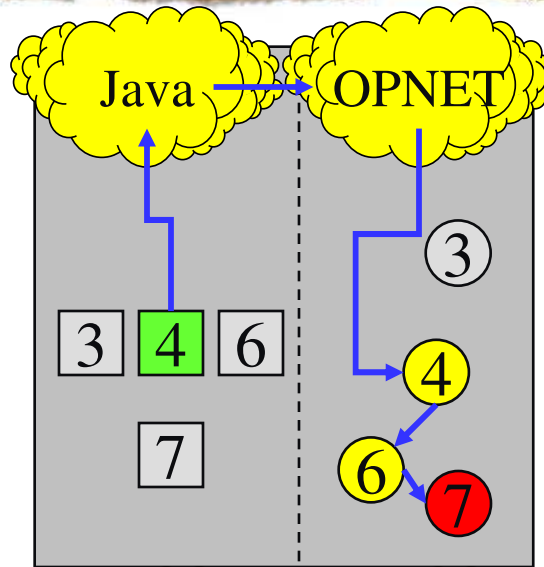


Communication Category	Net Source	Net Destination	Communication Mode
Networking	+ID1	+ID2	route (ID1 → ID2), e.g., AODV
	+ID1	0	broadcast (1x) ID1 → neighbors
	+ID1	-ID2	broadcast (1x) ID1 → ID2
	+ID1	$-\infty$	flood — reserved for future use
Inter-Process Communication (IPC)	0	0	Cloud → Cloud
	-ID1	0	ID1 → Cloud
	-ID1	-ID1	ID1 → ID1
	0	-ID1	Cloud → ID1

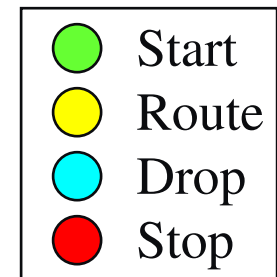
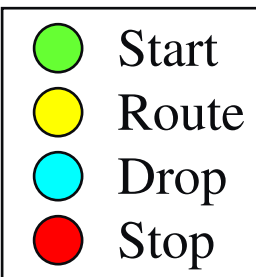
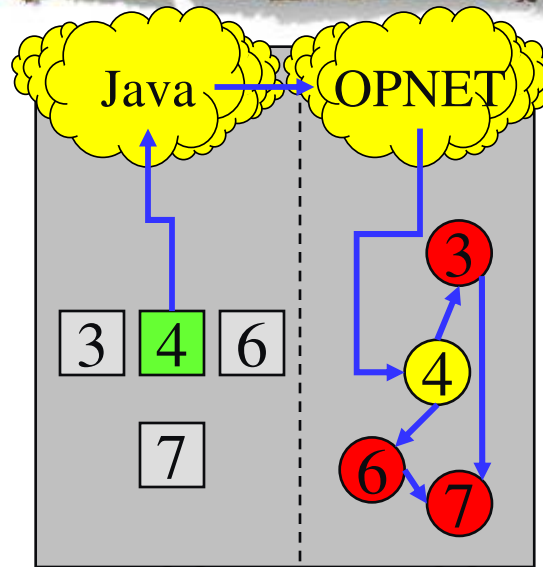
JOC-II: Communication Modes (2 of 3) Networking



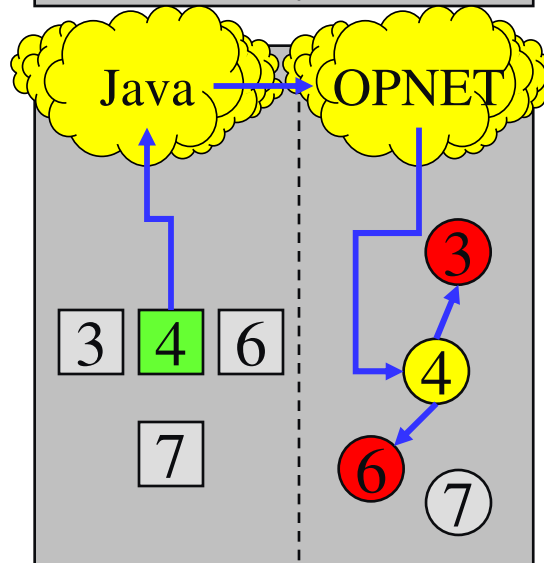
**Route, e.g.,
AODV, DSR
(+ID1, +ID2)**



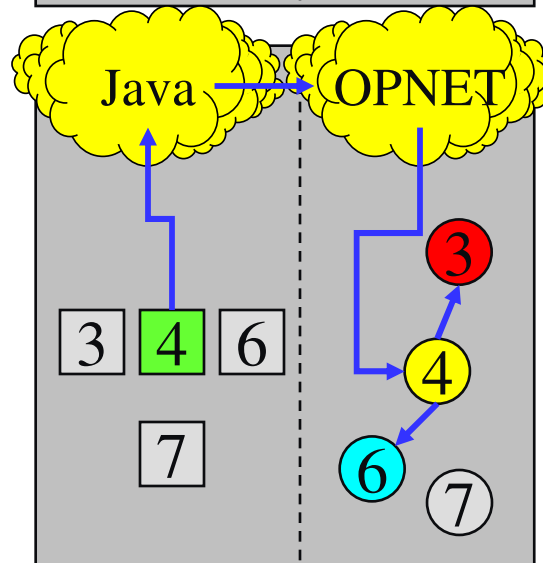
**Network
Flood
(ID1, +∞)**



**Neighbor
Broadcast
(+ID1, 0)**



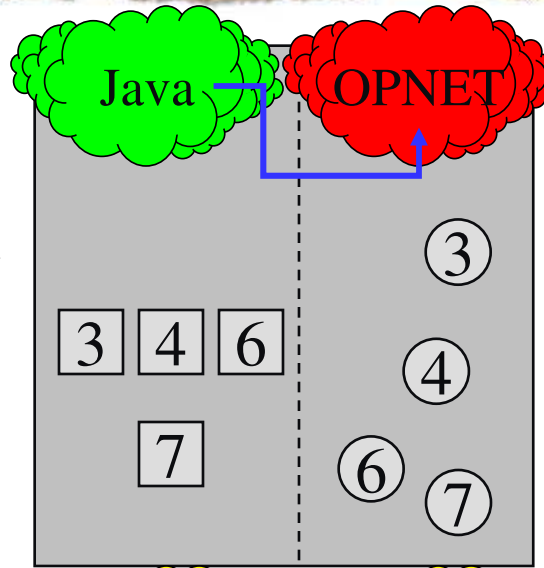
**Targeted
Broadcast
(+ID1, -ID2)**



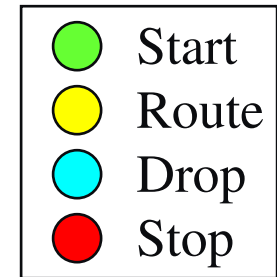
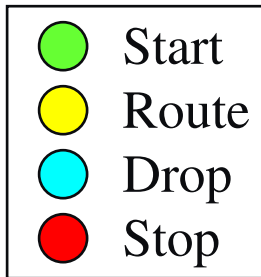
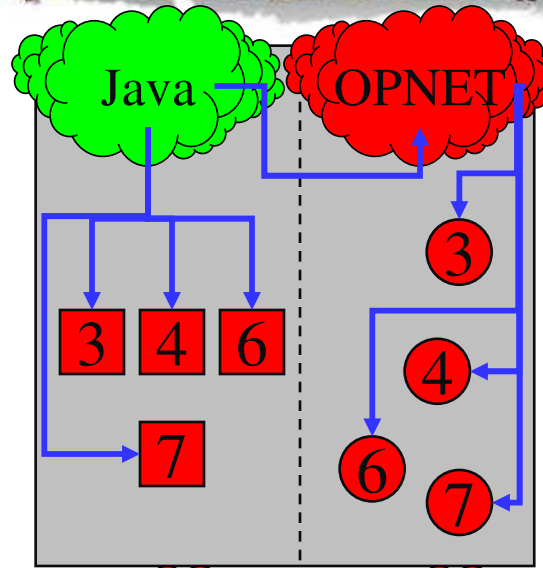
JOC-II: Communication Modes (3 of 3) Inter-Process Comm. (IPC)



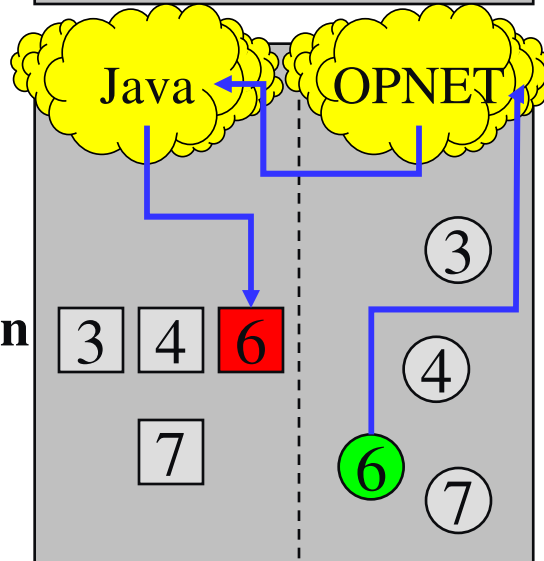
Configuration Management
(0, 0)



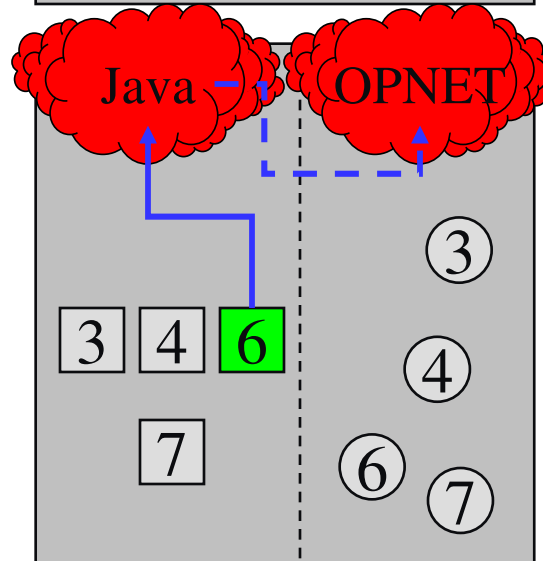
Policy Setting
(0, -ID1)



Inter-Process Communication
(-ID1, -ID1)



Status Updates
(-ID1, 0)



JOC-II: Starting a Simulation



Java Cloud: call C library wrapper

```
public static native int SimInit(...);
```

C Wrapper: call OPNET API

```
Esa_Main (...);
```

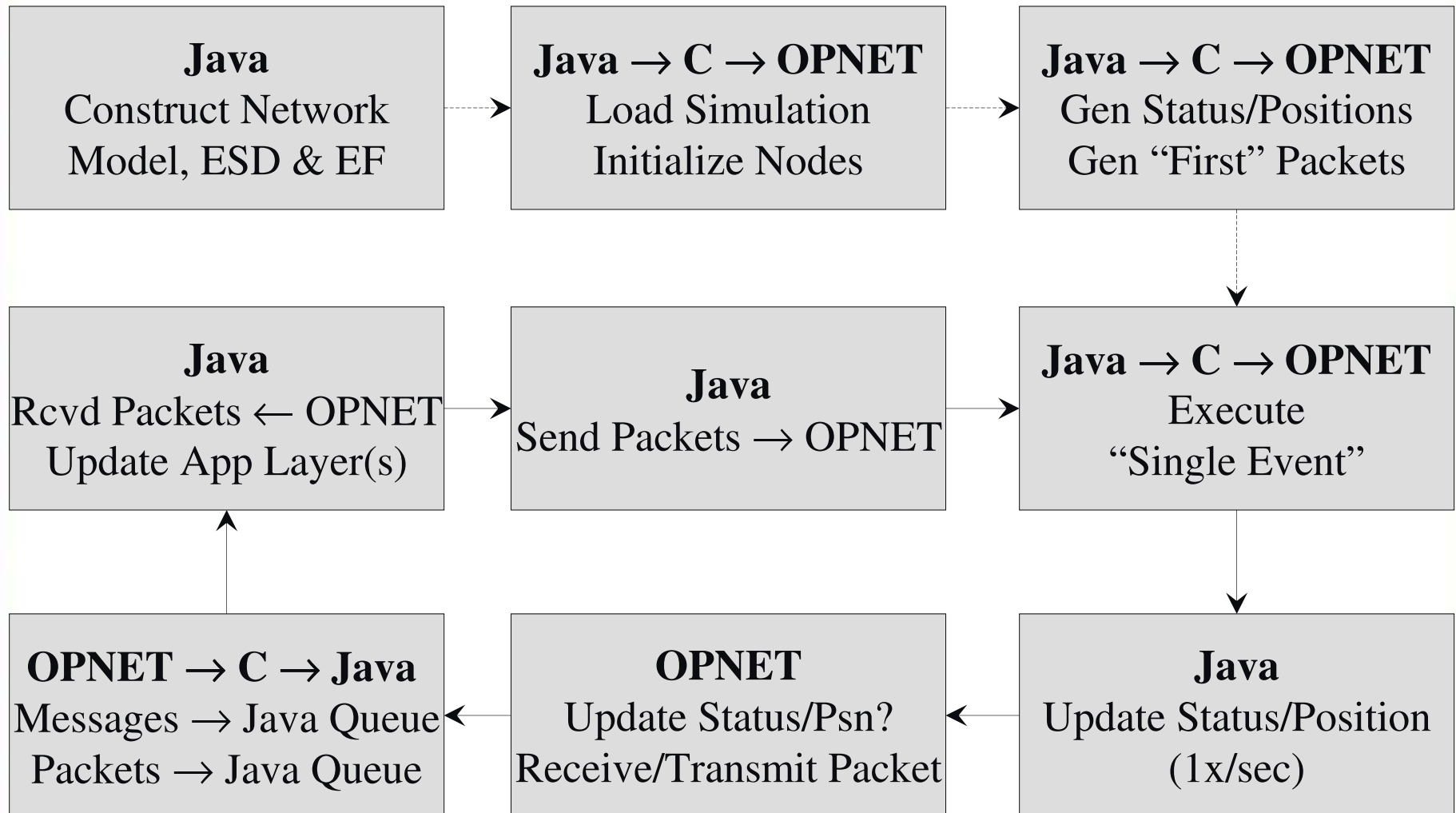
```
Esa_Text_Output_Callback_Register (...);
```

```
Esa_Init (...);
```

```
Esa_Load (...);
```

```
Esa_Interface_Callback_Register (...);
```

JOC-II: Control Flow



JOC-II: MANET Station Advanced Node Model



§ Why this model?

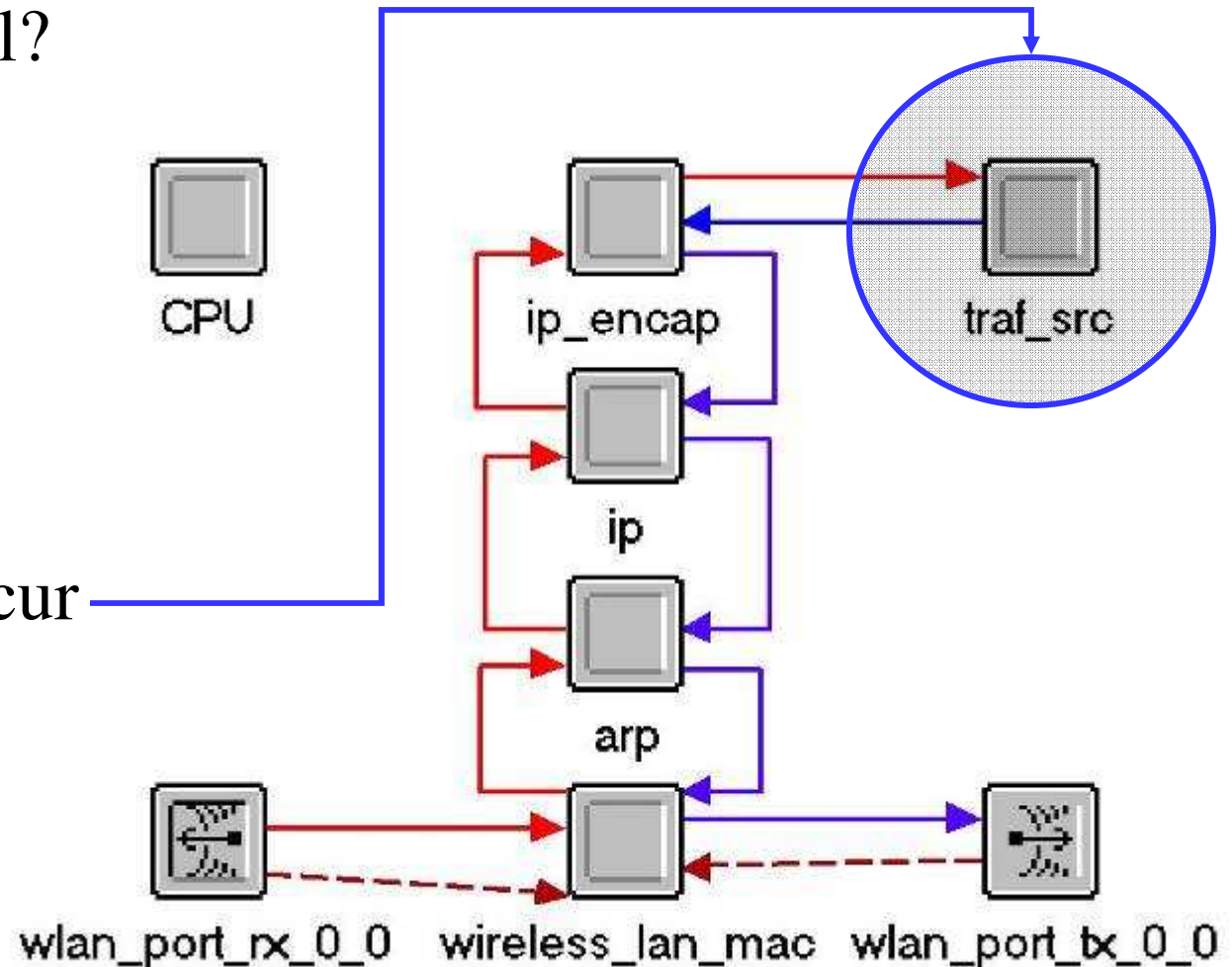
§ Mobility

§ Packet Generator

§ Wireless

§ IP

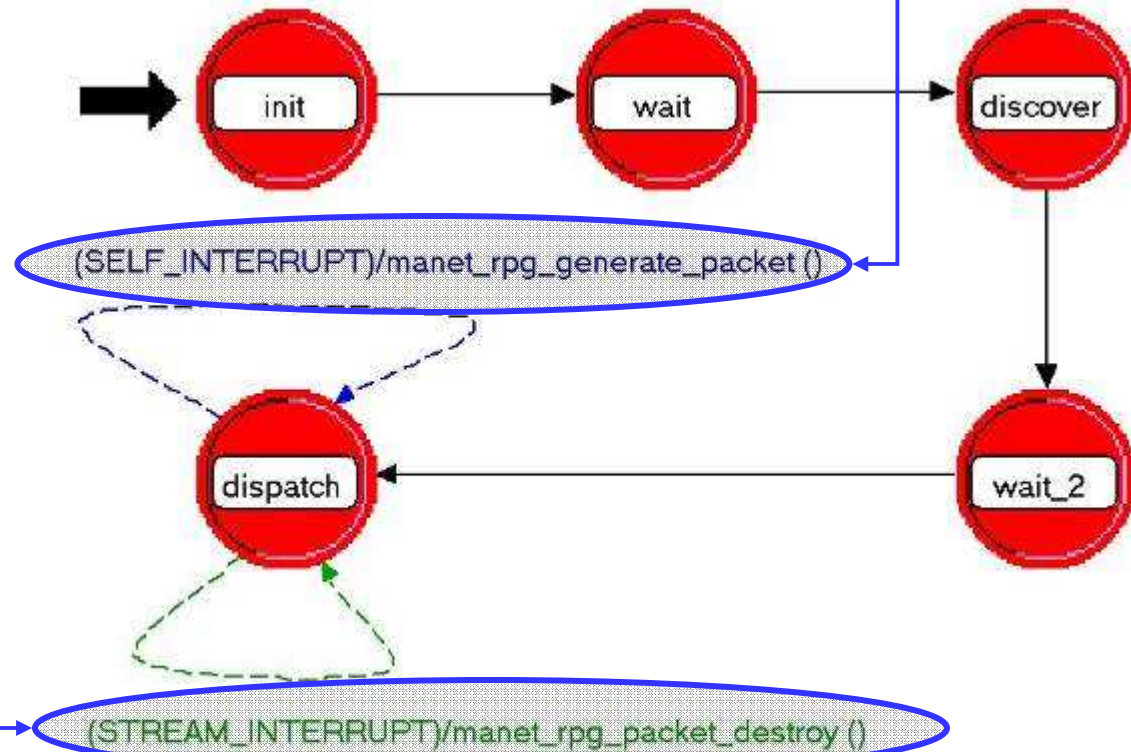
§ All changes occur in “traf_src” process model



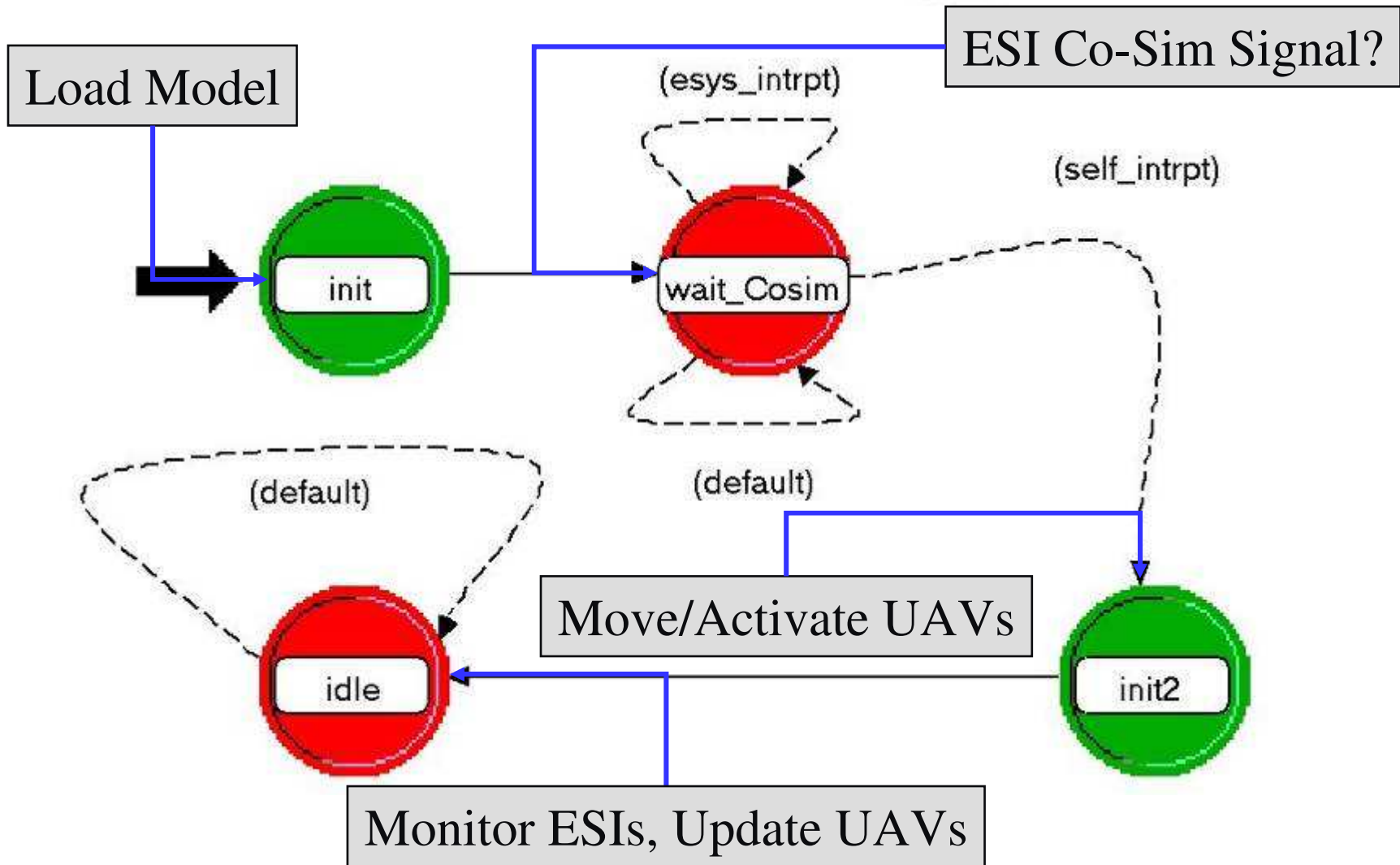
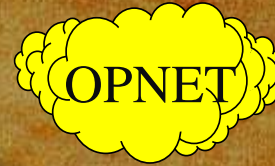
JOC-II: Traffic Source (traf_src) Process Model



- § Remove packet generator (1)
- § Disable packet destruction (1)
- § Add packet delivery to cloud (1)
- § Add packet format statistics handles (80)



JOC-II: OPNET Interface Cloud Process Model



JOC-II: Co-Simulation Results



**Simulation
clock**

The screenshot shows the JOCosim software interface. The title bar reads "JOCosim". The menu bar includes "File", "View", "Sim", and "Help". The "Simulation Control" panel is active, showing a "Simulation Time" of 311,500 secs. Below this is a "Step Size (secs)" field set to 1, with "Step", "Run", and "Stop" buttons. The "Output Messages" panel displays the OPNET logo and the text: "Simulation and Model Library Copyright 1987-2004 by OPNET Technologies, Inc. as a part of OPNET Release 10.5.A". Below the logo is the slogan "OPTIMUM NETWORK PERFORMANCE". The "Error Messages" panel shows the following text: "Loading OPNET... Registering OPNET callbacks... Initializing OPNET... Loading Simulation... Initiating Status... Sim Ready...".

**OPNET
messages**

**Co-simulation
messages**

- § Discuss an unmanned swarm design framework, HARVEST
- § Demonstrates Java & OPNET can successfully be used in a co-simulation
- § Shows that Java is able to maintain positive control of an OPNET simulation
- § Proposes that centralized cloud management provides a viable co-simulation mechanism

Questions?