P2P Interests at NRL

And our capabilities for modeling P2P middle-ware in MANETs.

PROTEAN
RESEARCH GROUP

Outline

- Unique Navy requirements
- NRL's approach to protocol engineering

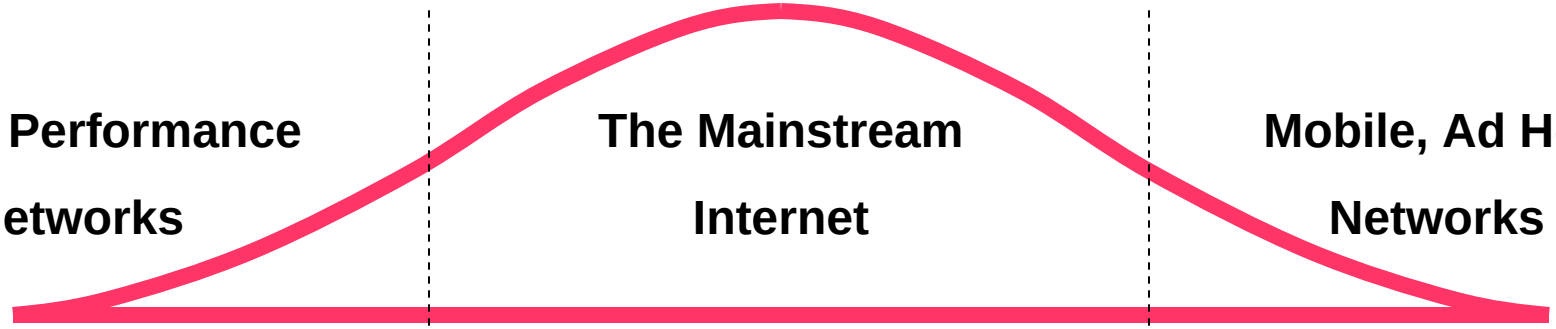
- Simulating Java apps with AgentJ + NS-2
- AgentJ implementation details
- AgentJ demo
- AgentJ development status

Challenging DoD Requirements

- Important to leverage open standards where sensible
  - Cost-savings, interoperability and maintainability.
  - Complex systems need “more eyes” on algorithms, code , etc.
- But DoD requirements often push beyond organic standard expectations:
  - Operations in highly stressed environments
  - Peak performance issues may override average performance
  - Rapid, flexible deployment with minimal pre-configuration and management
  - High volume of group communication as compared to current commercial systems
  - Heterogeneous mix of platforms, operating environments (air, land, sea)
  - Inter-service, national, and coalition interoperation
  - Security requirements
- Challenges:
  - Selection of appropriate standards
  - Adaptation of those standards to DoD environments
  - Understanding of performance and interactions across layers

A View of the Internet

(from 80,000 feet)



High Performance Networks

The Mainstream Internet

Mobile, Ad Hoc Networks

Some Common Characteristics

- **Stable infrastructure**
- **Fiber optic/High-speed RF/wireless optical**
- **Highest bandwidth**
- **Low latency**
- **Connection-oriented links**
- **Policy-based QoS**

Some Common Characteristics

- **Mixed range of assets**
- **Mixed media**
- **Tending to higher bandwidth**
- **Low to high latency**
- **Table-based routing**
- **Mixed policies in forwarding and QoS**

Some Common Characteristics

- **Ad hoc assets**
- **Generally wireless**
- **Design for degraded operation**
- **Large variability in latency and bandwidth**
- **Highly dynamic routing**
- **More distributed service models**
- **Change is the norm**

Service Oriented Architecture

Network Application Design and Issues

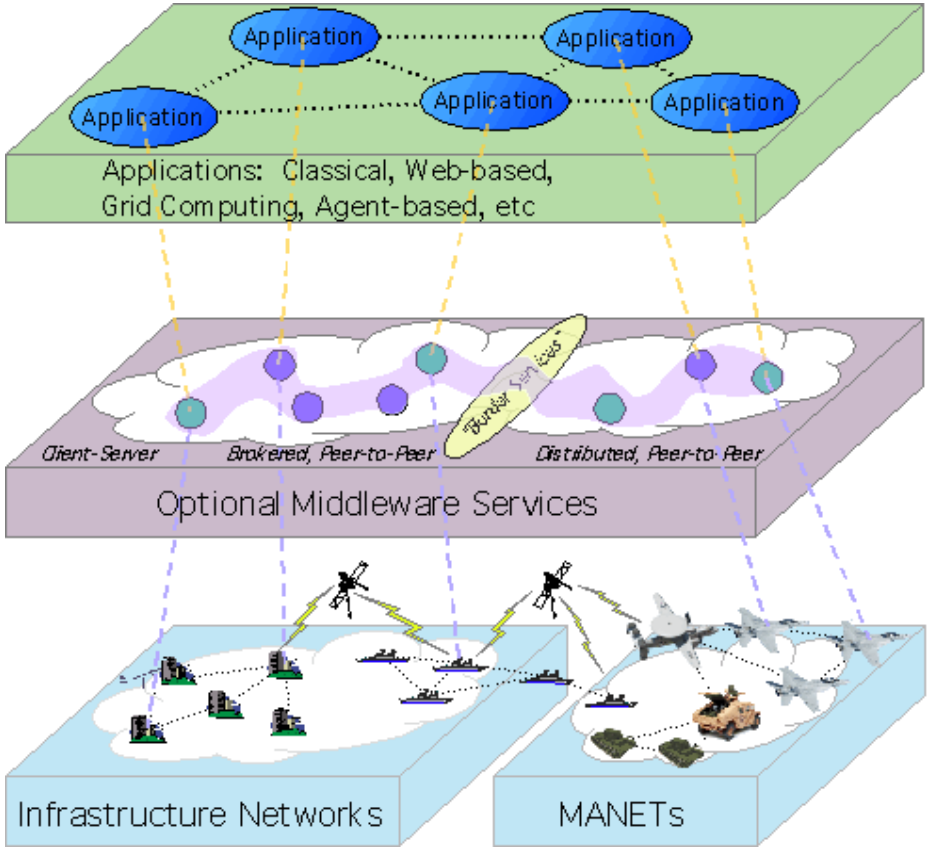
- Different distributed application paradigms (Web Services, Grid Computing, Agent-based)
- Different service types and reliability Requirements
- Highly Distributed and Dynamic Inter-app communication design issues
- Effectiveness of designs in MANET environments

Middleware Services Opportunities & Challenges

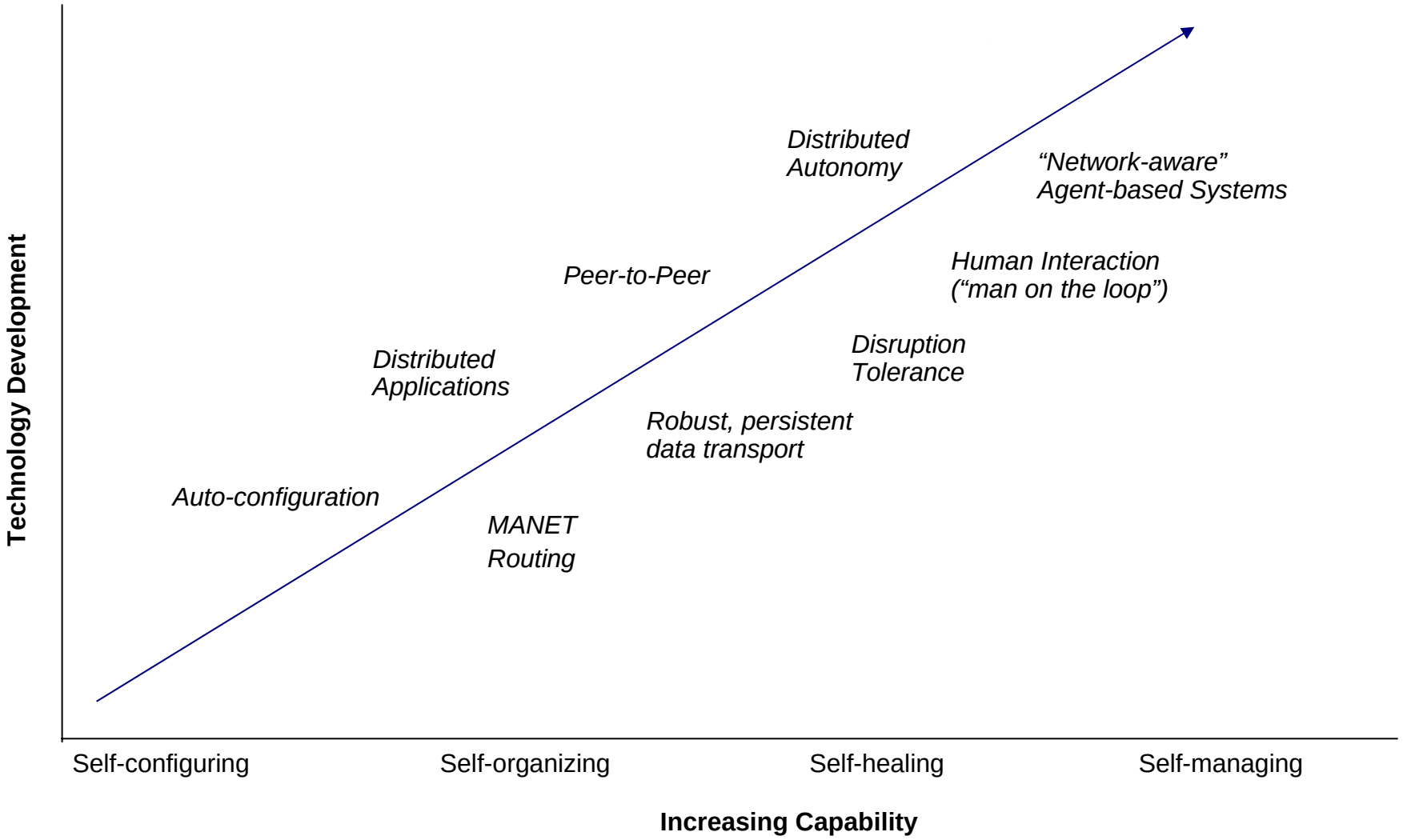
- Publish/Subscribe
- Peer discovery
- Service discovery
- Persistent data transport
- Application security services
- **Very immature in MANET environments**

Network Challenges

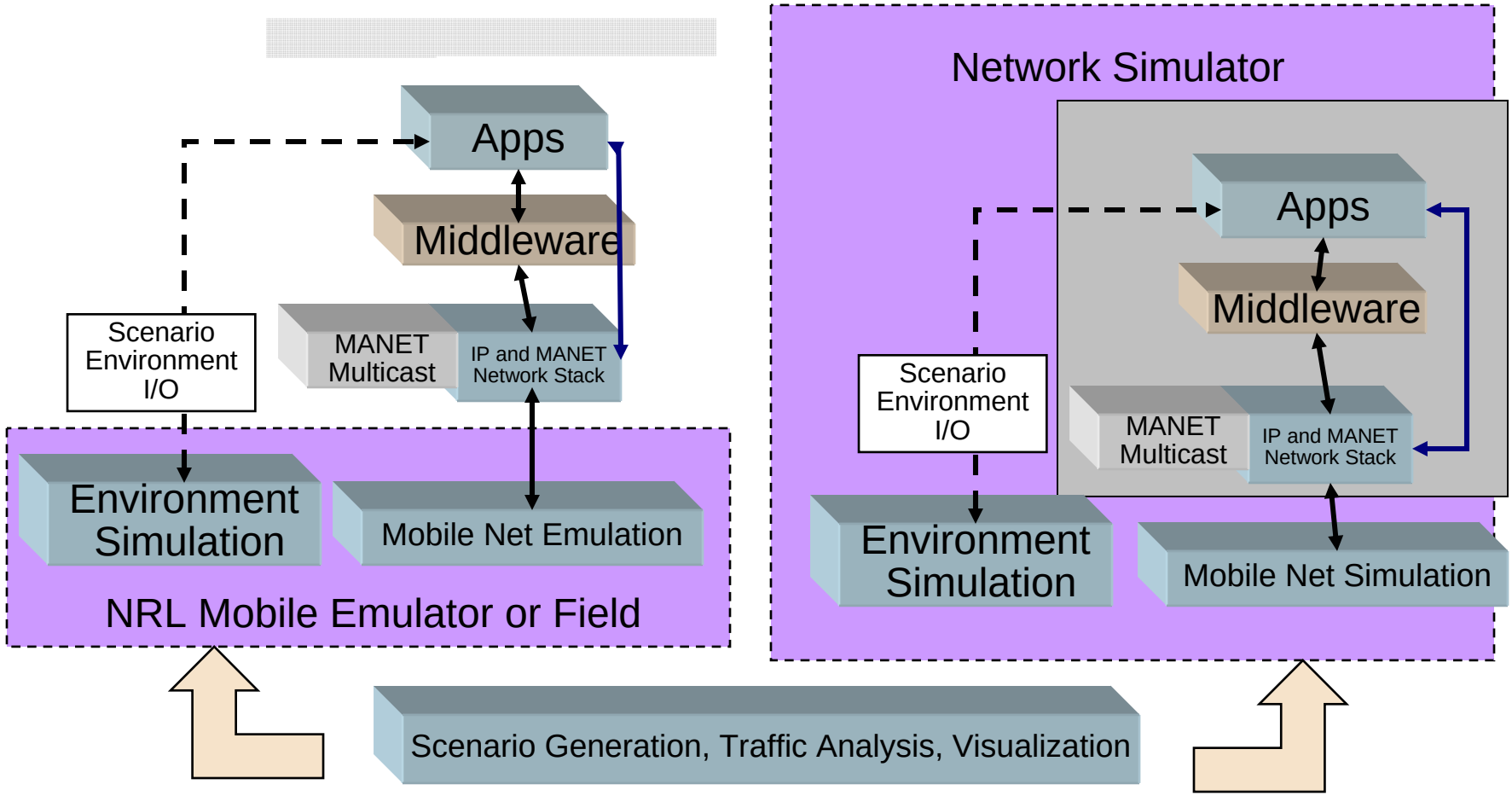
- Infrastructure vs. Ad-hoc environments:
  - Tactical Edge networks and platforms may have intermittent connectivity to infrastructure.
  - Different network services ("underware" incl. transport, name resolution, auto-configuration, discovery, etc) may be needed for extreme environments.
- Cross-layer integration for better performance



Vision: Potential Evolution of Network-based Systems



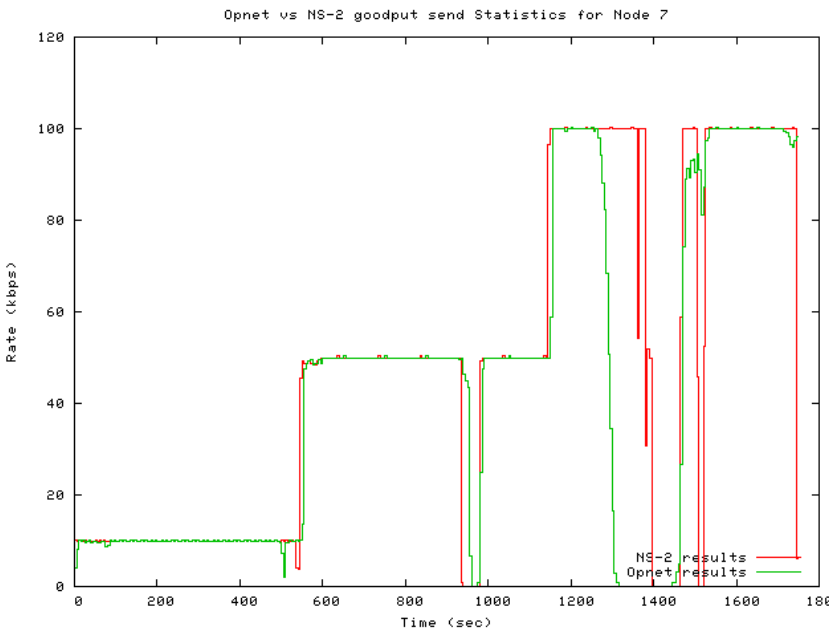
Our approach is multi-layered.

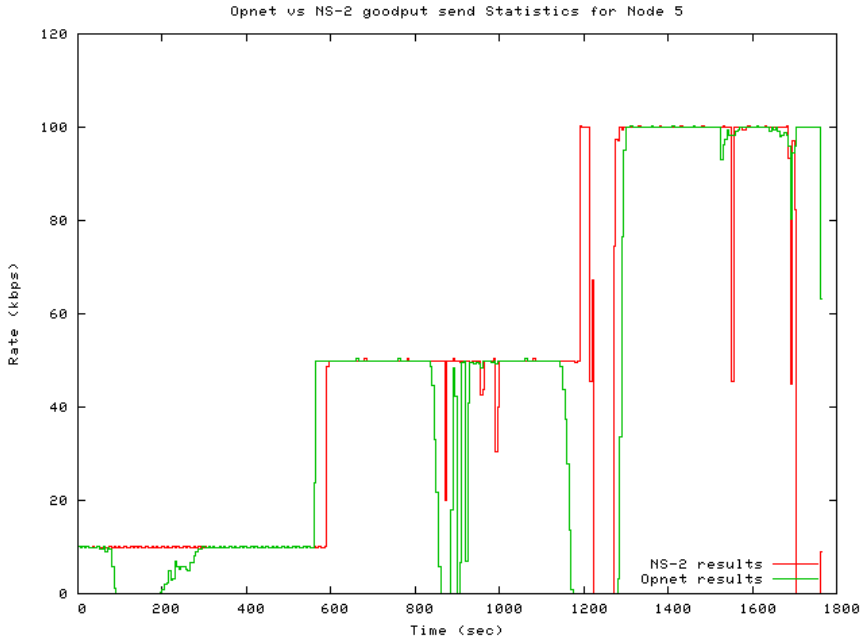


Approach

- Simulation
  - Allows for possible large scale evaluation.
  - Rich set of candidate protocols available
  - Virtually unlimited instrumentation, highly reproducible
  - NRL has used *ns-2* and *OPNET* modeling tools
- Emulation
  - Real systems with emulated connectivity, automated for possibly reproducible results
  - Low to moderate scalability
  - Encompasses real-world *system* phenomena
  - Often more detailed and comprehensive than simulation
- Field Testing
  - The “real deal” including subtleties of physical phenomena (propagation, etc)

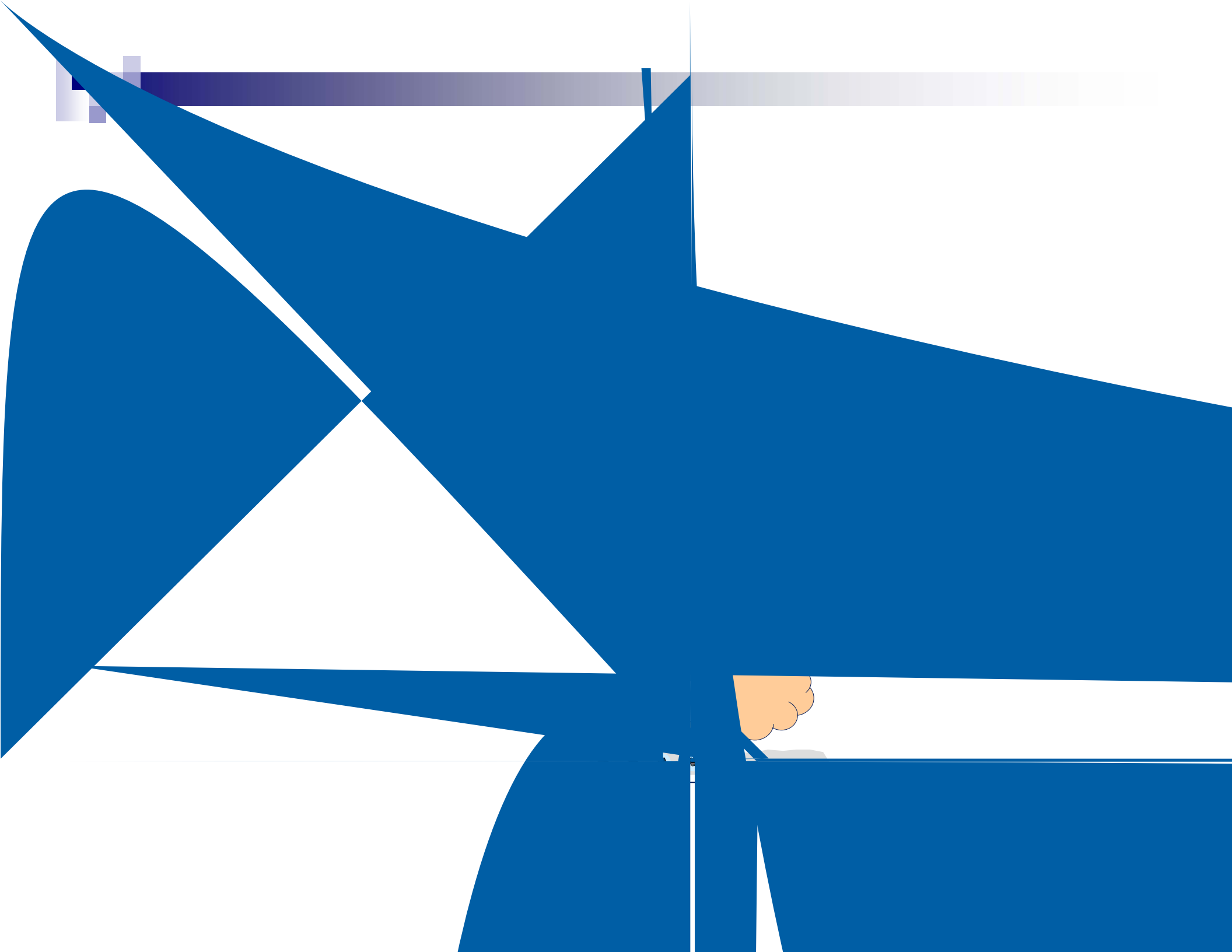
For example: Opnet vs NS-2 Multicast (SMF) Results



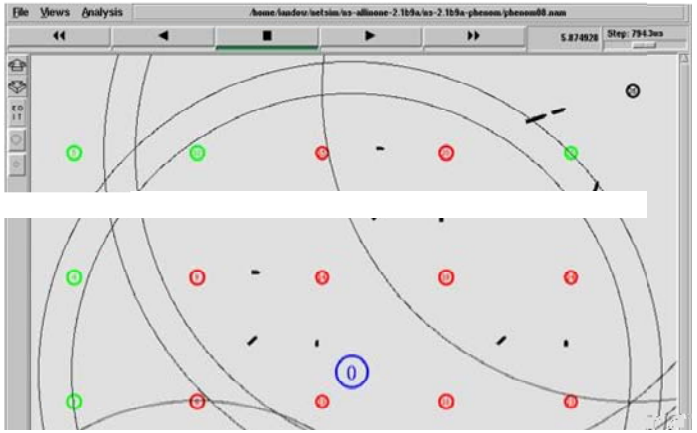


Simulate (NS-2, Opnet), Emulate (MAN, MANE), Live-network Tests…

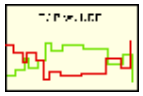
Ideally, these different approaches should cross-validate their results!



Our products



- Protocol development
  - OLSR
  - Multicast (simplified and reliable)
  - Anycast
  - Protolib
- Analysis Tools
  - Simulation models (operating on real-world code)
  - Mobile Network Emulator
  - Traffic generator
  - Visualization and network animation tools



- All **open sourced**, to move the community forward.
  - http://pf.itd.nrl.navy.mil

Recommendations for the P2Prg CORE research approach

1. Localized service discovery
2. Responsive and reliable discovery
3. Cross-layer design

Interpreting java.net for P2P Middleware Models in NS-2

Presented by Ian Downard.

Regarding:

- From: "John Buford" <buford@research.panasonic.com>
  To: <p2prg@ietf.org>
  Date: Sat, 28 Jan 2006 23:04:27 -0500
  Subject: [P2Prg] request for presentations at IETF65 on P2P Simulation Tools

“Given the wide set of tools for simulating P2P systems as described by Alan's and Mario's recent draft, we are proposing to have some presentations at the next P2PRG meeting at IETF 65 by researchers who are building such simulations. Goal would be to share mutual experiences with both existing and home-grown tools.”

Background

- NRL has been working with Cardiff University in Wales who have a p2p implementation entitled "P2PS" (Peer-to-Peer Simplified) for applications in Grid computing and web services

- Using AgentJ, we have ported a version of P2PS into ns-2, and we are conducting ongoing work with looking at p2p in MANET environments.

Motivation

- AgentJ is being used at NRL to test peer-to-peer protocols for service discovery in simulated wireless networks. This work is motivated by a need to discover resources in disadvantaged mobile ad-hoc networks. Presently, we are looking at P2P approaches to resource discovery and comparing them against various multicast techniques in a focused effort to produce middleware that conserves bandwidth and energy, reduces congestion and contention, and exhibits robustness to network fragmentation and node mobility.

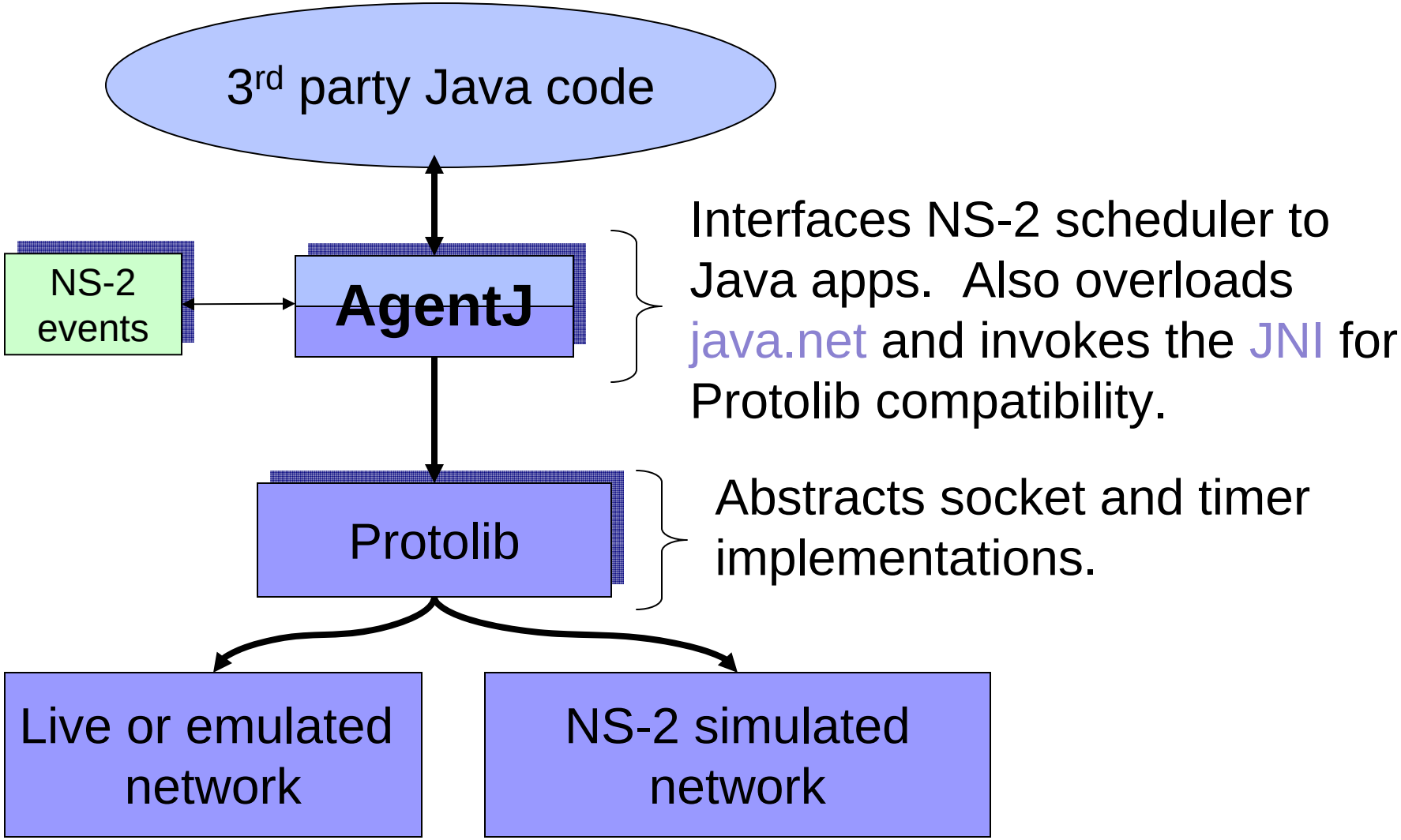
References

- IRTF P2PRG Internet Draft, “Tools for Peer-to-Peer Network Simulation”
  - draft-irtf-p2prg-core-simulators-00.txt

AgentJ Presentation Goals

1. What is AgentJ? Design summary.
2. Technical capabilities
3. Demo
4. Implementation caveats, and ongoing work.

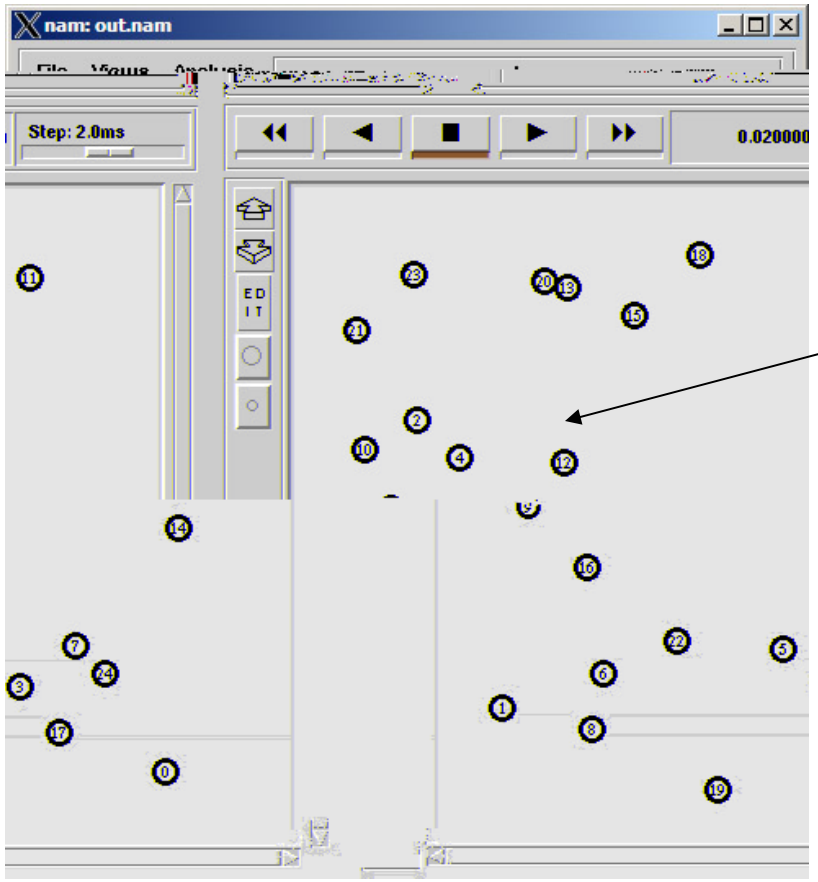
AgentJ Overview



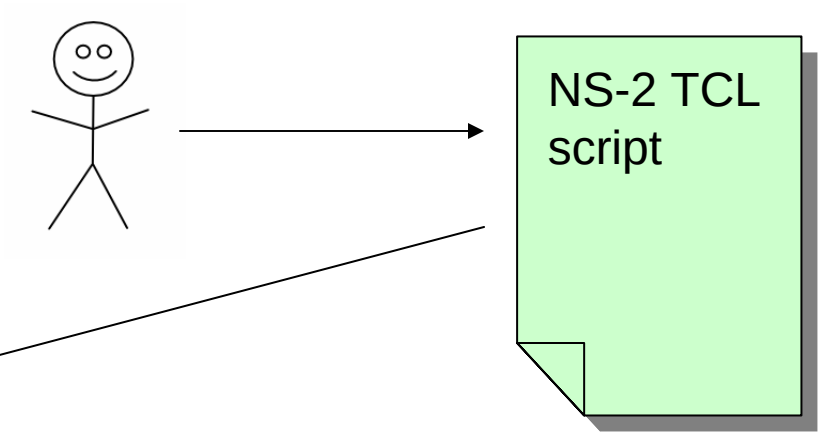
Important!

- AgentJ is an **NS-2 agent,**
  - not an agent as in “Mobile Agents”, or “Intelligent Agents”

How the simulation is configured.

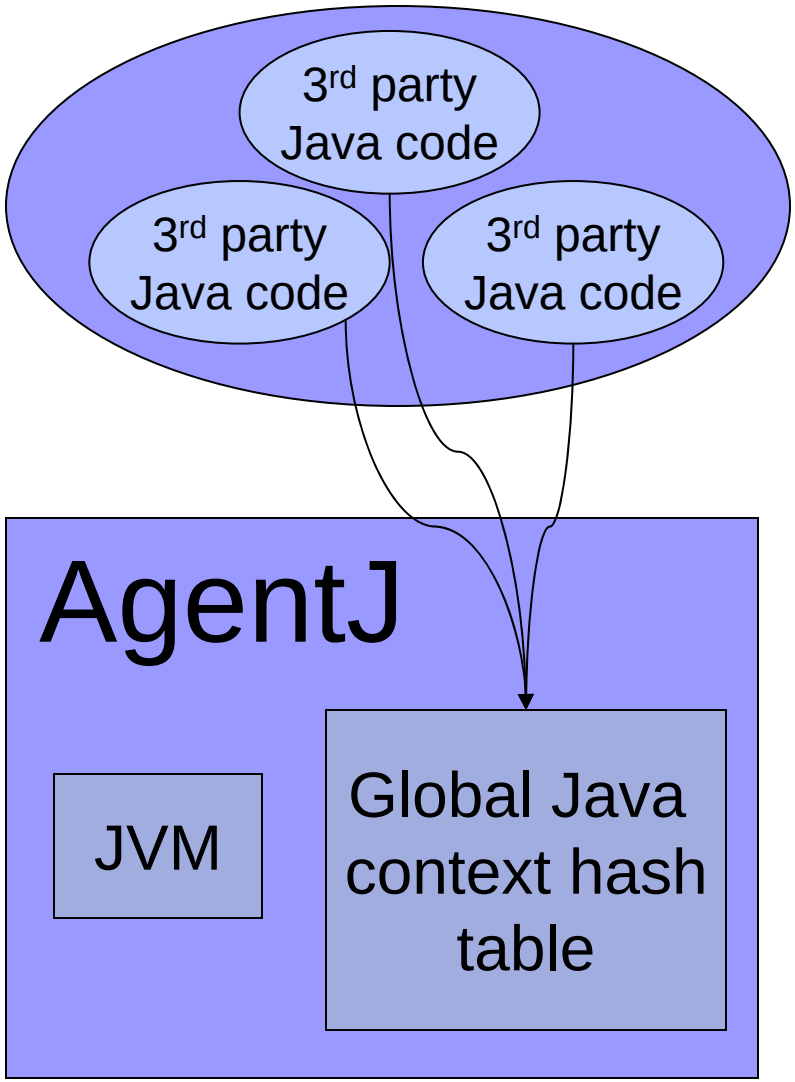


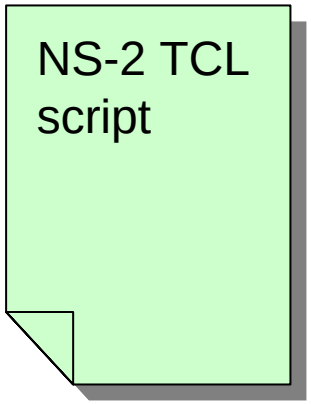
NS-2 network animator screenshot of a 25 node MANET.



1. User creates the network (nodes, MAC, motion, etc)
2. User instantiates 3rd party Java applications into NS-2
3. And binds them to AgentJ
4. Finally the simulation is RUN!

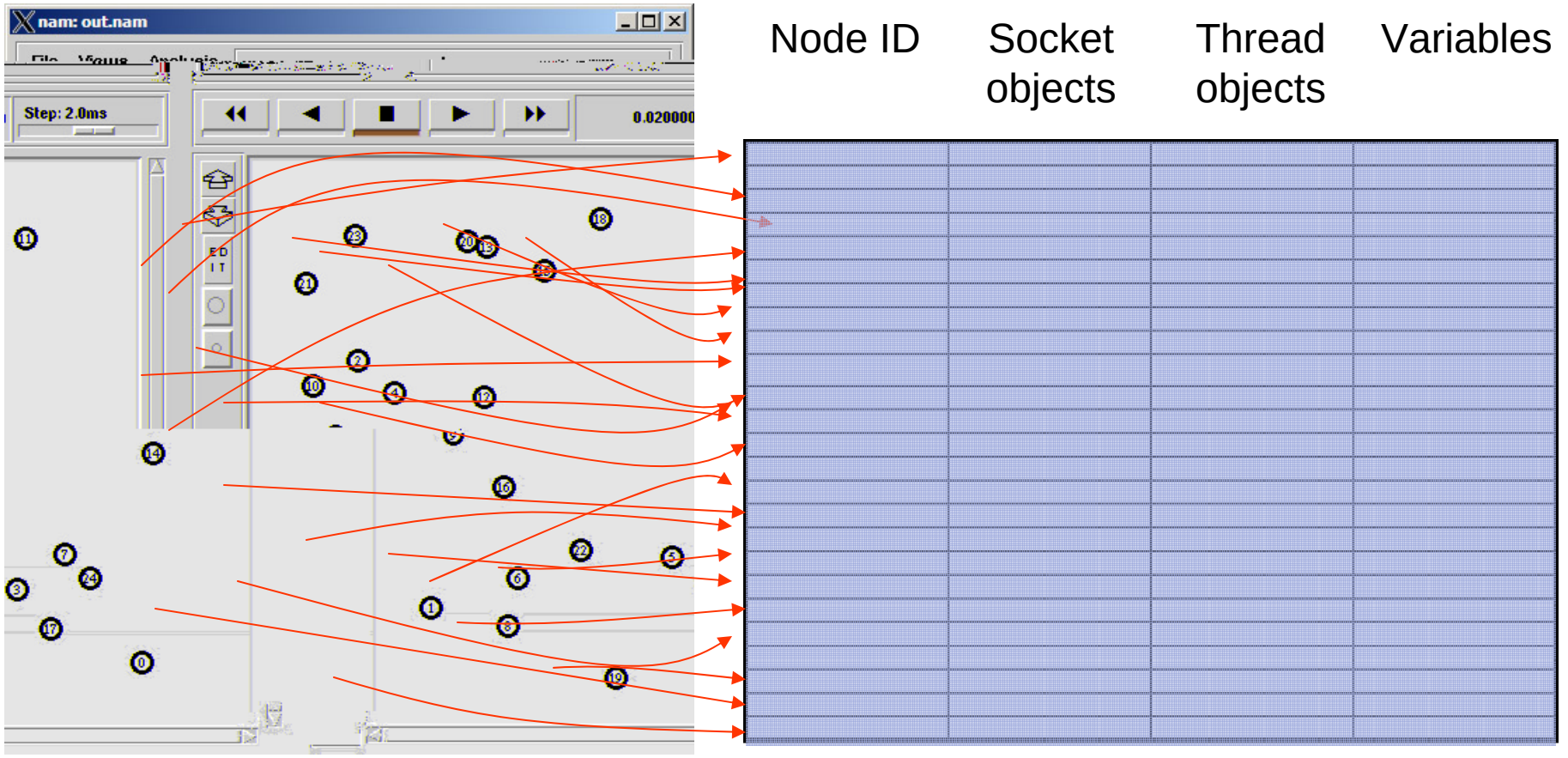
How the simulation is configured.





1. User creates the network (nodes, MAC, motion, etc)
2. User instantiates 3rd party Java applications into NS-2
3. And binds them to AgentJ.
4. Finally the simulation is RUN!

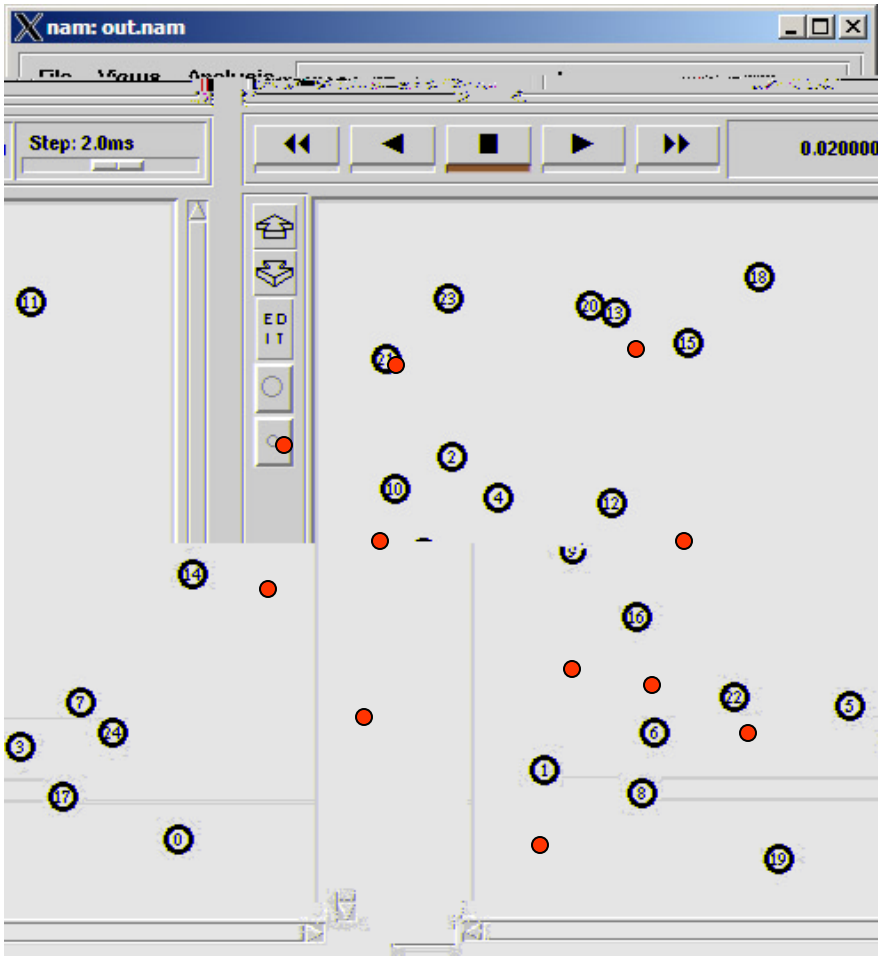
AgentJ maintains a global context for every Java object in the simulation.



NS-2 network animator screenshot of a 25 node MANET.

Global Java context hash table

The simulation runs!

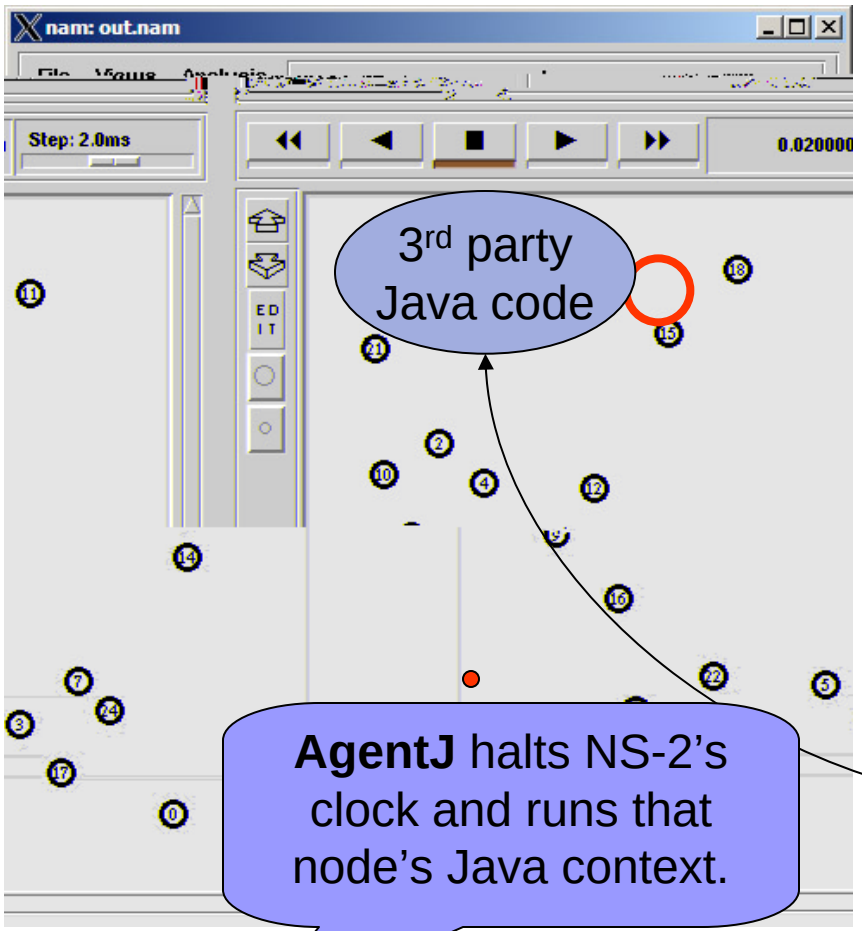


Simulated time

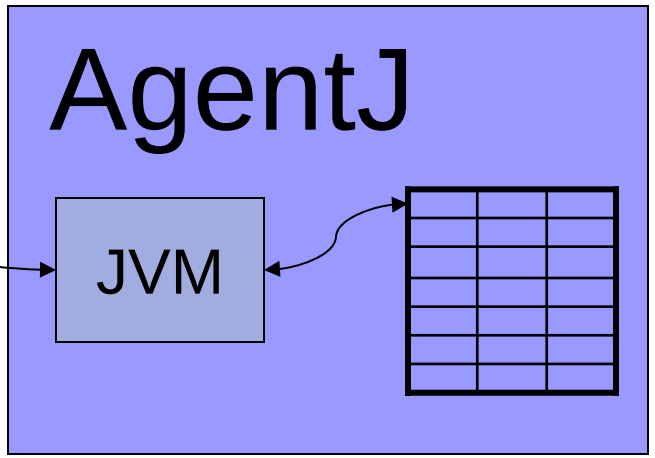
NS-2 TCL script

1. User creates the network (nodes, MAC, motion, etc)
2. User instantiates 3rd party Java applications into NS-2
3. And binds them to AgentJ.
4. Finally, the simulation is RUN!

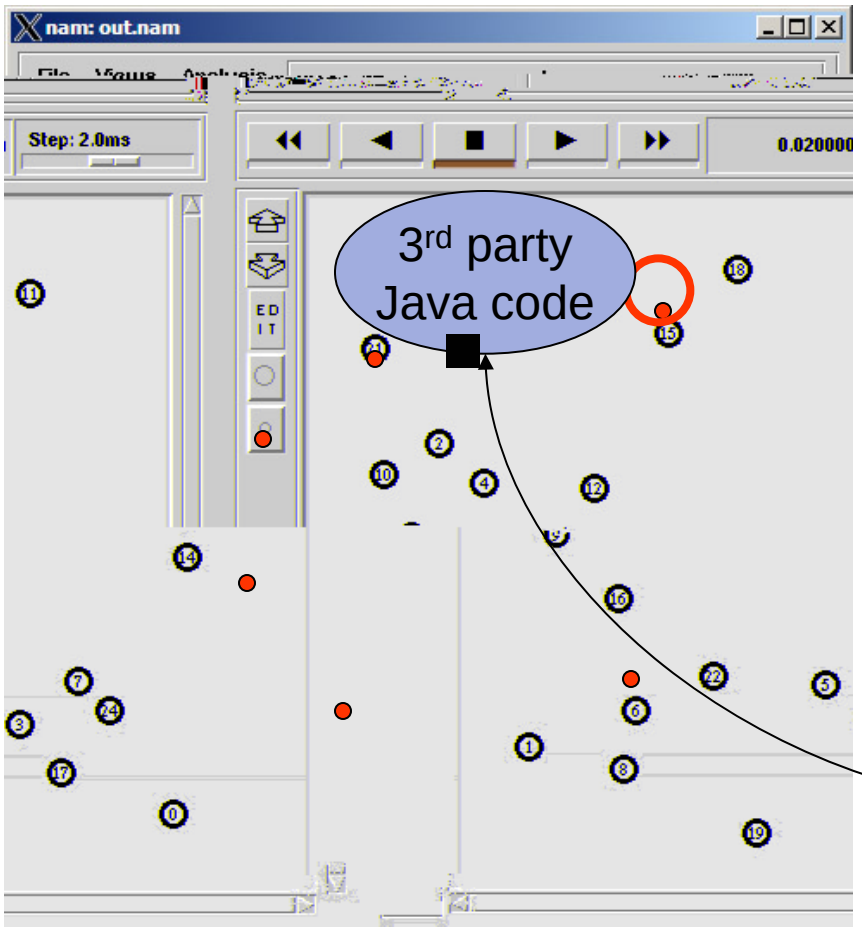
AgentJ runs when NS-2 fires off an event for a Java object.



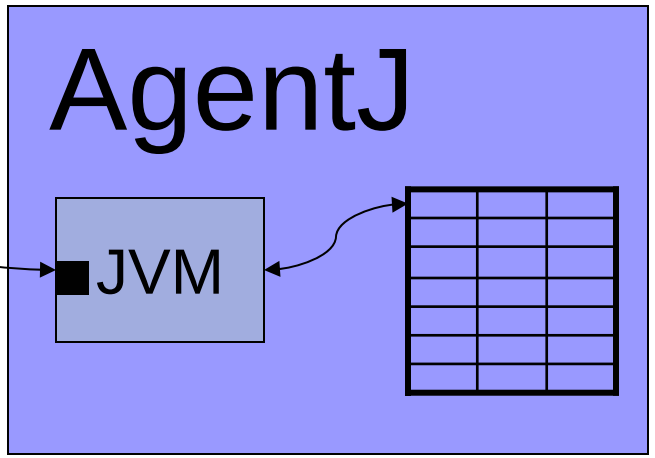
- NS-2 fires off an event for a Java object, such as *data received*, or *timer expired*.



AgentJ runs when NS-2 fires off an event for a Java object.



- When the Java code returns or blocks, AgentJ gives clock control back to NS-2



Simulated time

AgentJ Design Summary

- AgentJ instantiates a single JVM for the entire simulation
- AgentJ loads a node's Java context into the JVM whenever NS-2 fires off an event for that node.

AgentJ threading capabilities

- Includes limited support for multi-threaded Java network applications
- Synchronous socket I/O implemented via Java byte code rewriting
  - java.net ➡ agtj.net
- AgentJ also supports asynchronous socket I/O with Protolib sockets pai.net

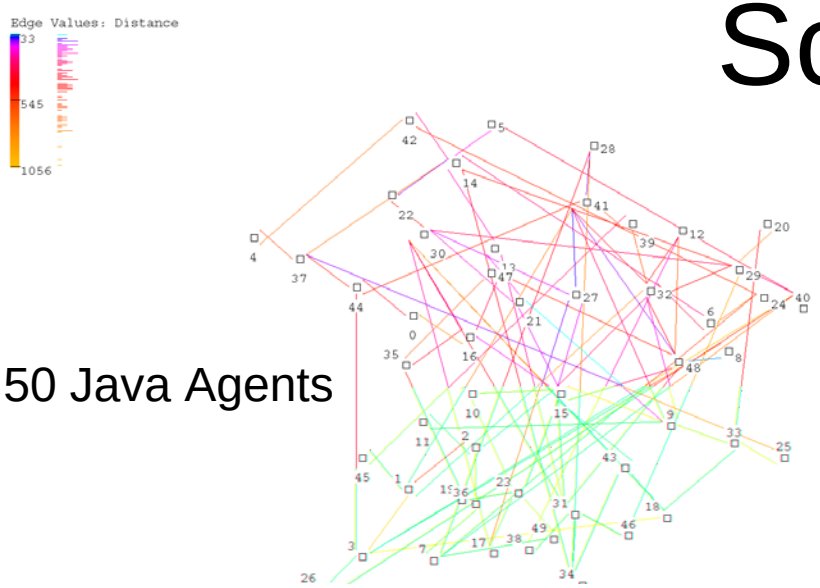
Java.net support

- Create sockets on a port
  - DatagramSocket (int port)
  - MulticastSocket (int port)
    - This is **untested** due to limitations in group based multicast routing in NS-2. Note, we can use SMF by addressing DatagramPackets to -1 (the NS-2 broadcast address).
- Send packets on a *socket* to an *address*
  - SocketAddress (java.lang.String addr, int port)
    - Accepts IP addresses or NS-2 addresses
  - DatagramPacket (byte[] buf, int length, SocketAddress address)

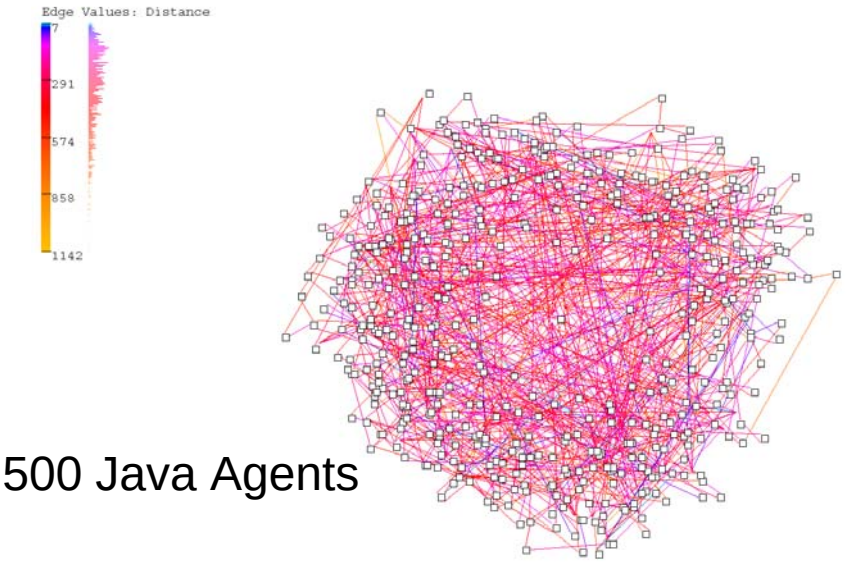
java.lang.Threads support

- Thread.start()
  - Spawns new java threads
  - Used by java.net.DatagramSocket to receive packets
- Thread.sleep() not yet implemented, but should be later.
  - Requires use of the Protolib Timer Interface, which has not been tested much.
  - Java – oTCL interface provides a workaround

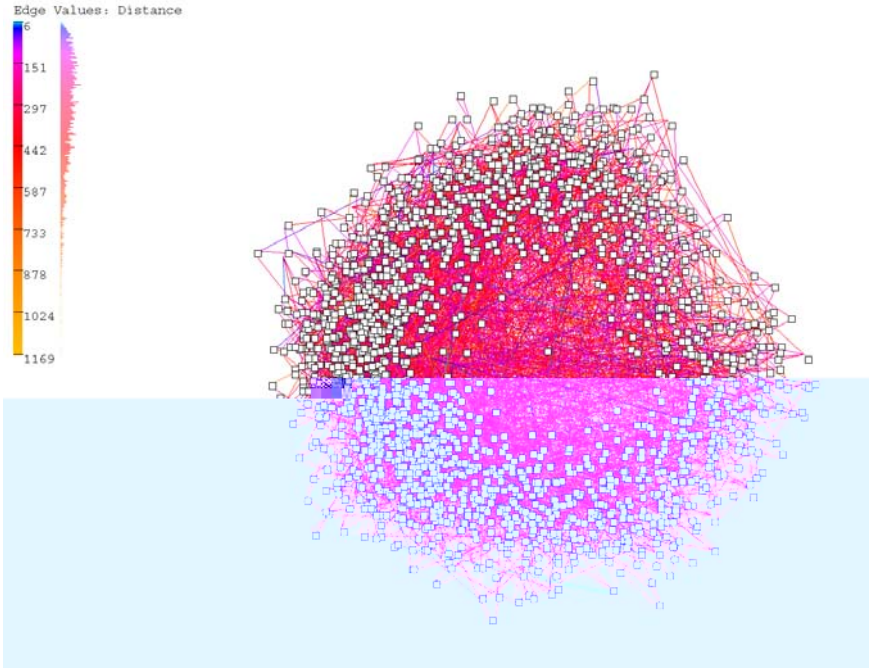
Scale?



50 Java Agents

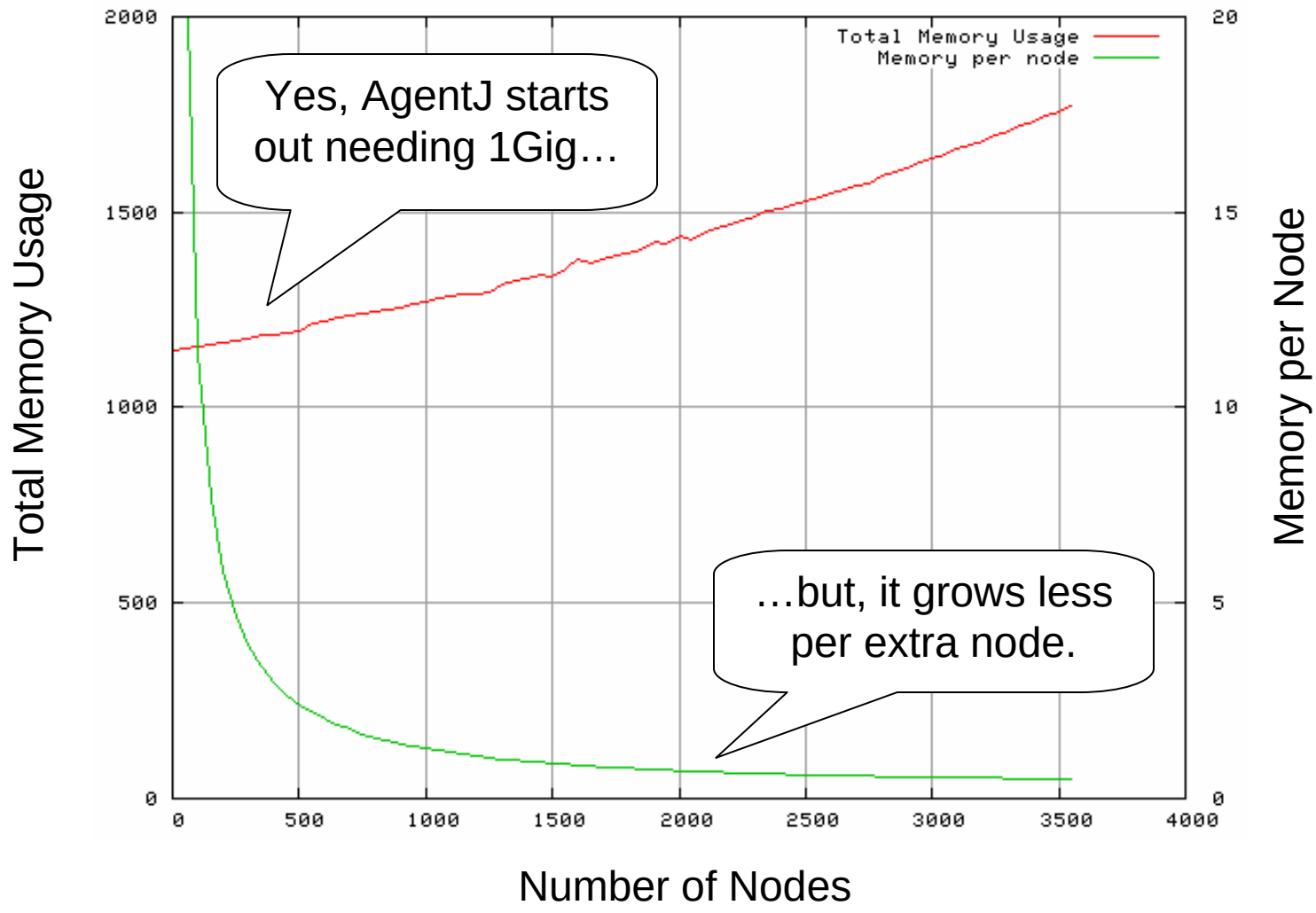


500 Java Agents

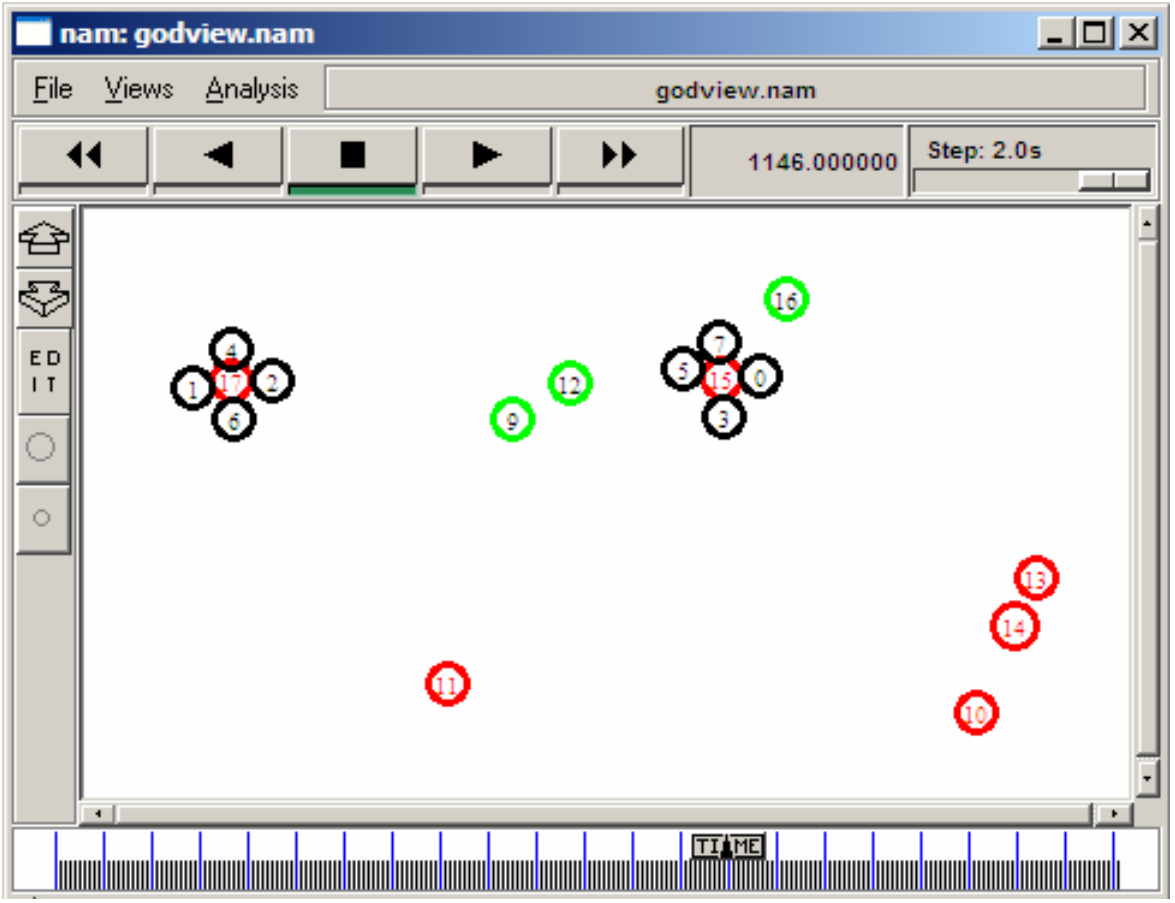


2000 Java Agents

Memory Complexity for AgentJ in NS-2



AgentJ Demo



Mine field scenario

Mobile Agent Systems

- A study in role allocation with “intelligent” agents.
- Measuring the agents’ resistance to unreliable comms.

Current AgentJ Products

- Documentation (conference publication, JavaDoc, web page)
  - http://pf.itd.nrl.navy.mil/srss
- Basic proof-of-concept applications (client-server)
- Total of **50+ classes**, including:
  - sample applications
  - Protolib socket and timer interfaces
  - multicast socket support
  - command interface to oTCL
  - java.net, java.lang.Threads reimplementations

Development Status

- “AgentJ: Enabling Java NS-2 Simulations for Large Scale Distributed Multimedia Applications”, accepted for publication at the Distributed Frame for Multimedia Applications (DFMA) conference, May 2006.

- *(Alpha code)*
- Limited java.lang.thread capabilities
- Limited java.net capabilities

Ongoing work

- Mobile intelligent agent systems
- Integration with P2PS

- Attempting to attract more open-source developers, who can help expand our java.net and java.lang.Threads implementations.


NAVAL RESEARCH LABORATORY
WASHINGTON, DC


PROTEAN
RESEARCH GROUP


Agent J

Questions?

Peer-to-Peer Simplified

1. An Overview of the *Peer-to-Peer Simplified (P2PS)* Architecture
2. Outline it's integration into NS-2
3. Tutorial for simulating Java Apps in NS-2.
4. Demonstration of P2PS Service Discovery over a multihop wired network in NS-2

PROTEAN
RESEARCH GROUP

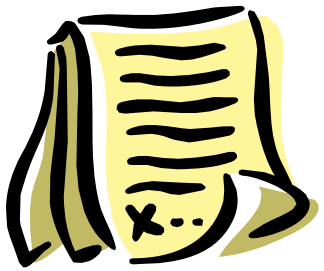
CARDIFF UNIVERSITY
PRIFYSGOL CAERDYDD

Terminology used in P2PS

- XML
- Advertisements
- Discovery
- Queries
- Pipes
- Services
- Endpoints
- Pipe/Endpoint Resolution
- Rendezvous peers

Advertisements

- Nodes announce network resources they have access to via advertisements written as XML documents.
  - Pipes
  - Services
  - Endpoint addresses (of peers)
  - Rendezvous nodes



- XML is used as a language independent format for exchanging messages.

Advertisements

- *Pipe advertisements* describe communication channels' attributes such as:
  - unidirectional, bidirectional
  - secure, insecure
  - reliable, unreliable
  - multicast, unicast
- *Discovery pipes* typically operate over UDP multicast, which allows peers in a subnet to hear each other's advertisements.

Advertisements

- *Endpoint advertisements* announce the address a peer is using for sending and receiving data.
  - Pipes are resolved to endpoint addresses by *resolver peers*.
- Peers which can resolve endpoint addresses are advertised with *Endpoint Resolver advertisements.*

Advertisements

- *Rendezvous Advertisements* announce specialized peers, called rendezvous peers, that can forward queries (not advertisements) across multiple subnets.
- *Rendezvous Peers* cache advertisements and queries. When a cached adverts are matched with new queries, or cached queries are matched with new adverts, then the rendezvous peer broadcasts the query within its local discovery subnet.

Queries

- Used to locate specific resources (pipes, services, endpoints, pipe/endpoint resolvers, etc).
- Queries request a response from peers which can match the query to a cached advertisement.

P2PS/NS2 Design Overview

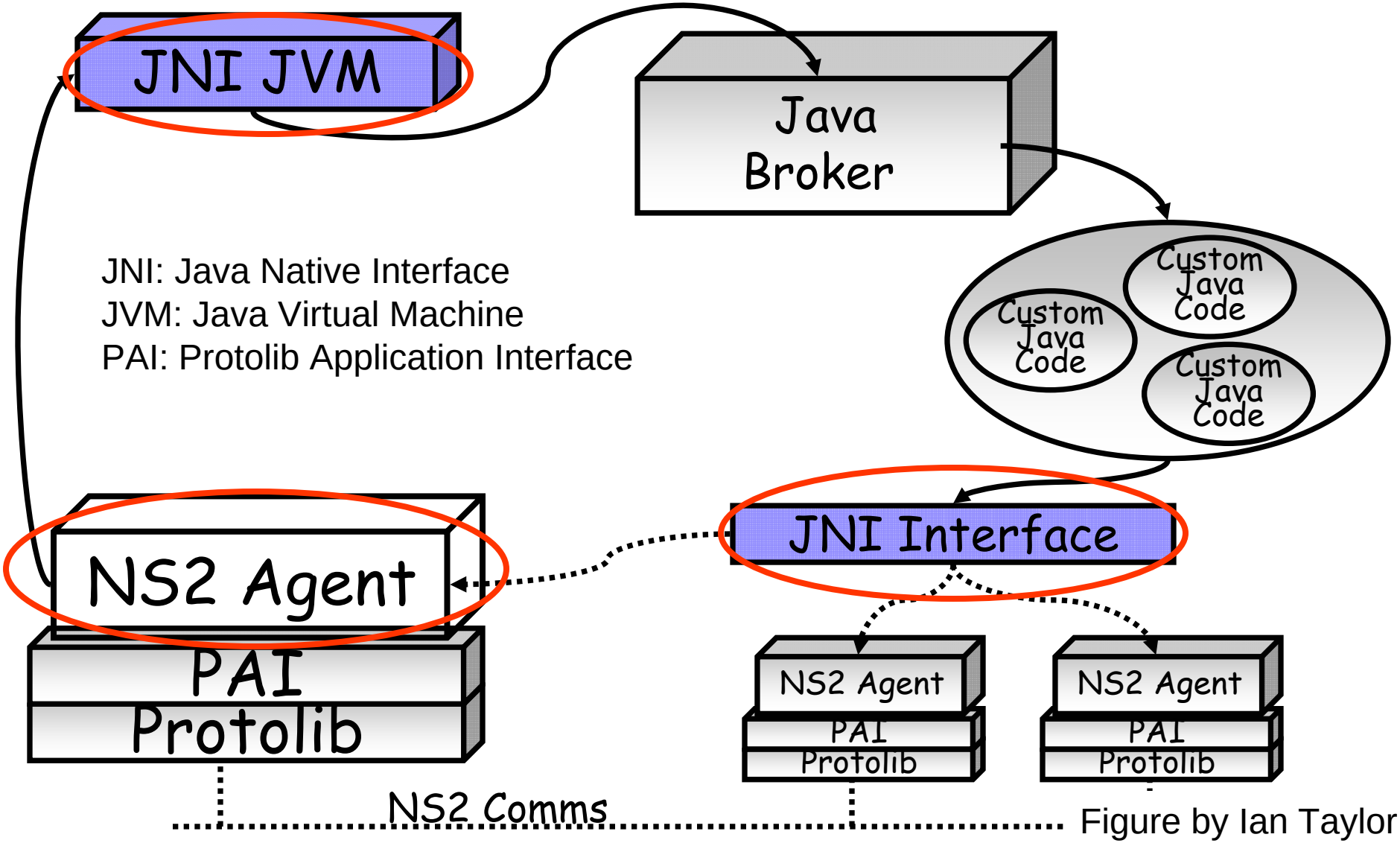
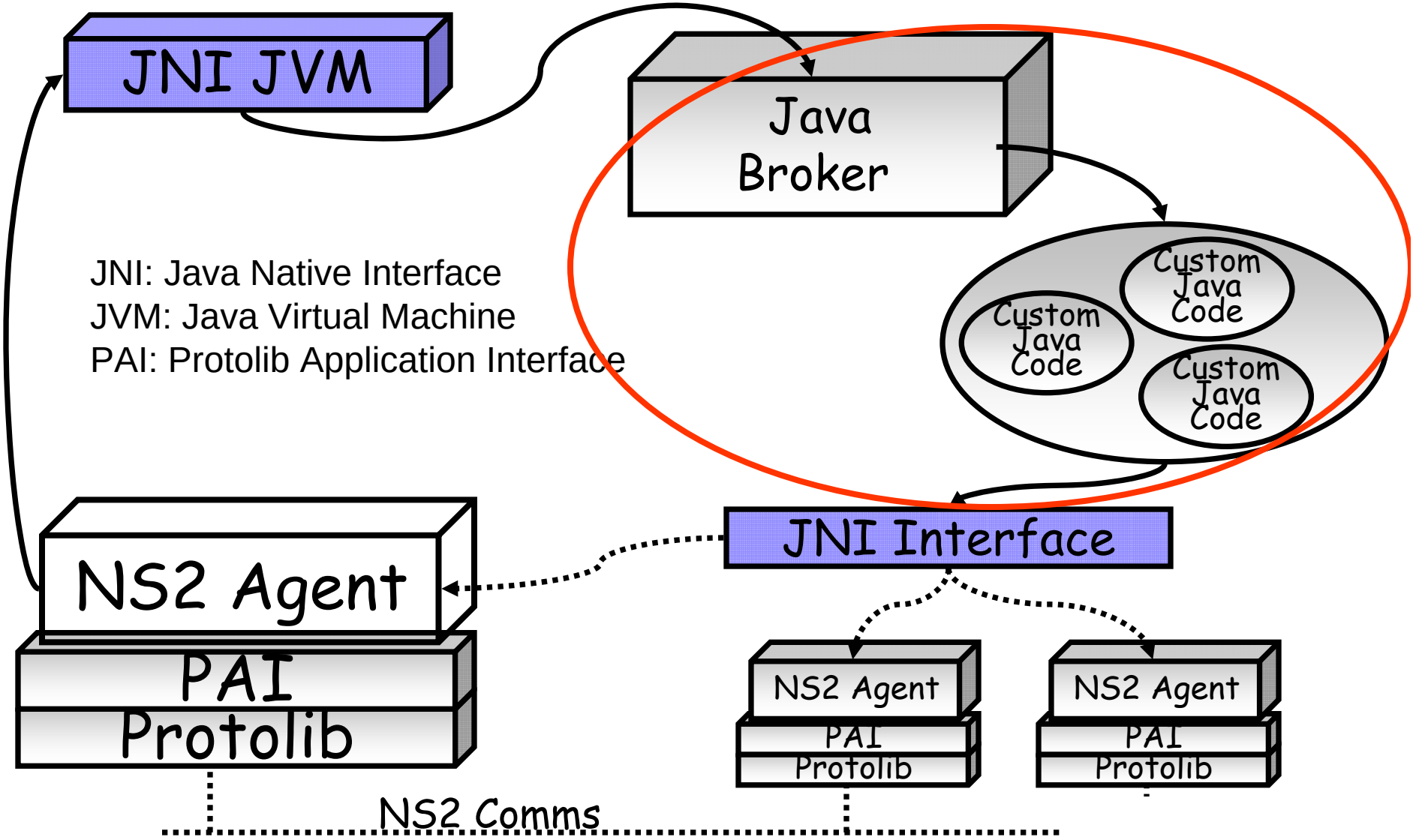


Figure by Ian Taylor

P2PS/NS2 Design Overview



How P2PS Interfaces with Protolib

PAI – Protolib Application Interface
PCI – Protolib Communication Interface
PTI – Protolib Timer Interface

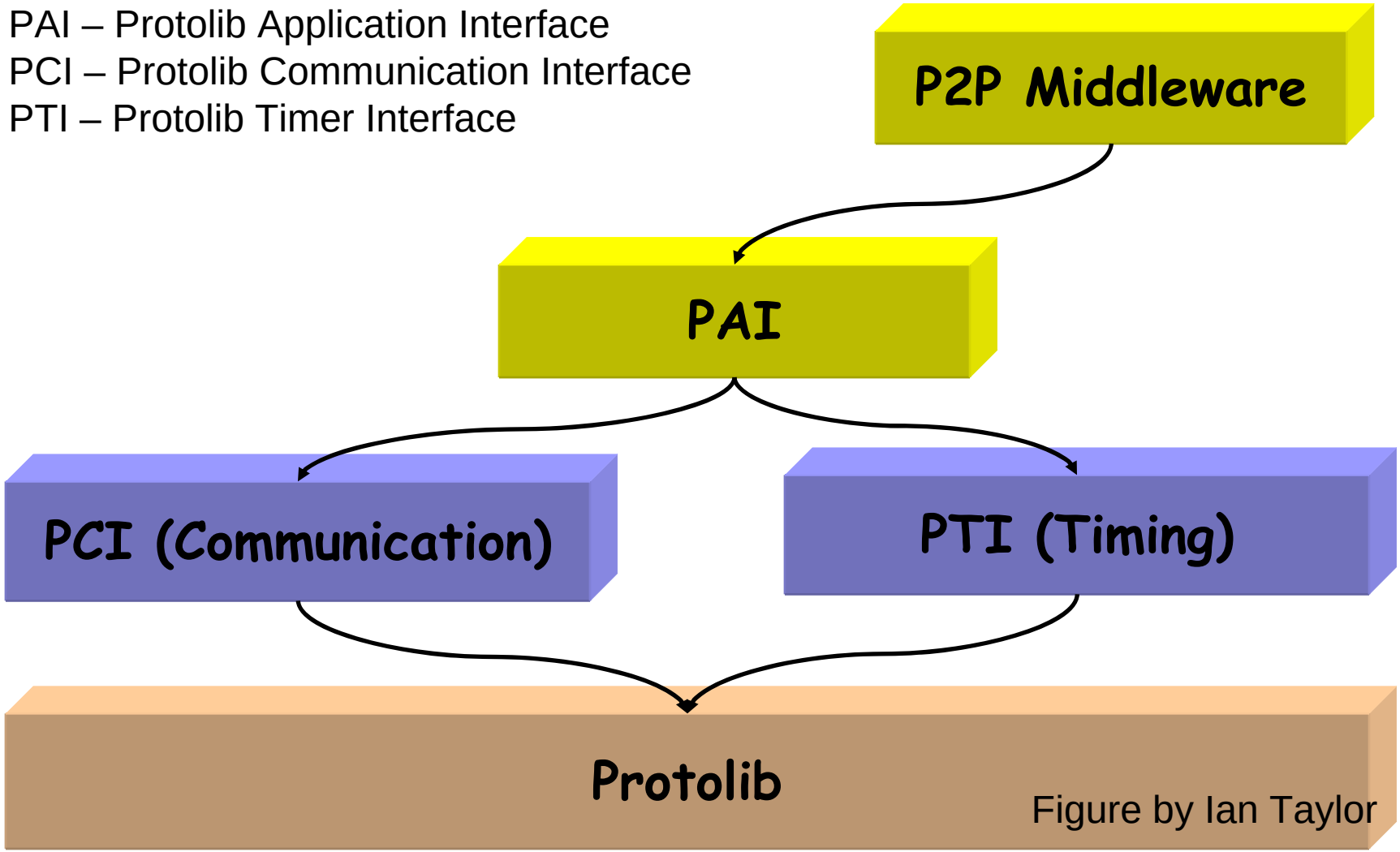
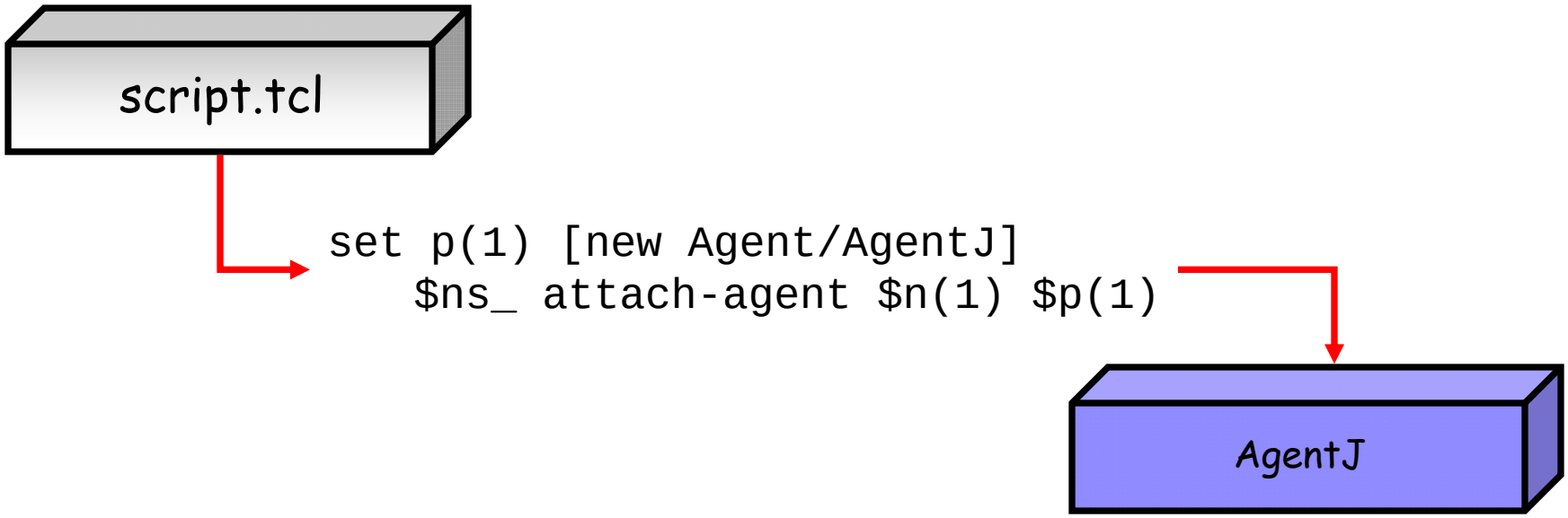
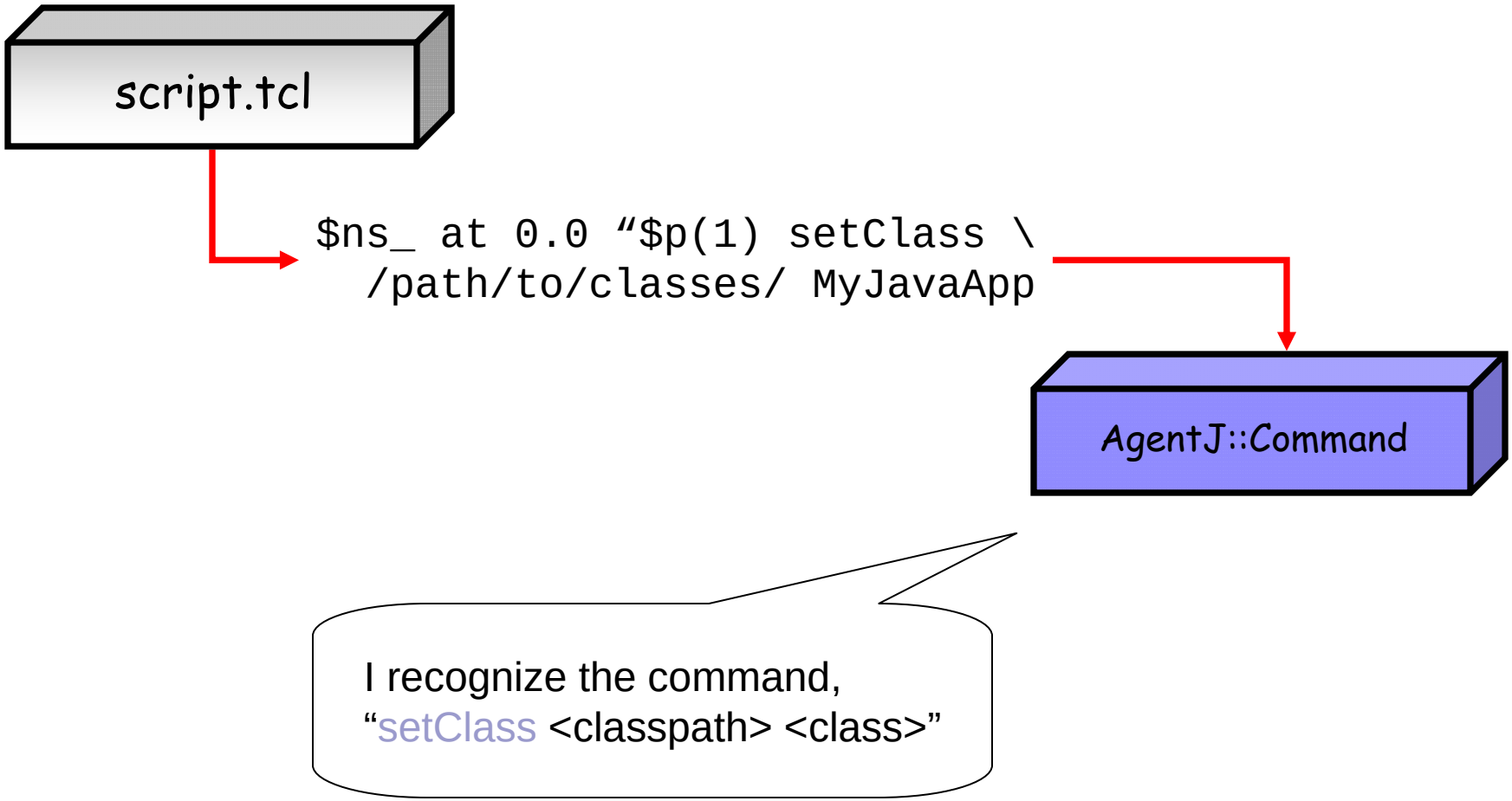


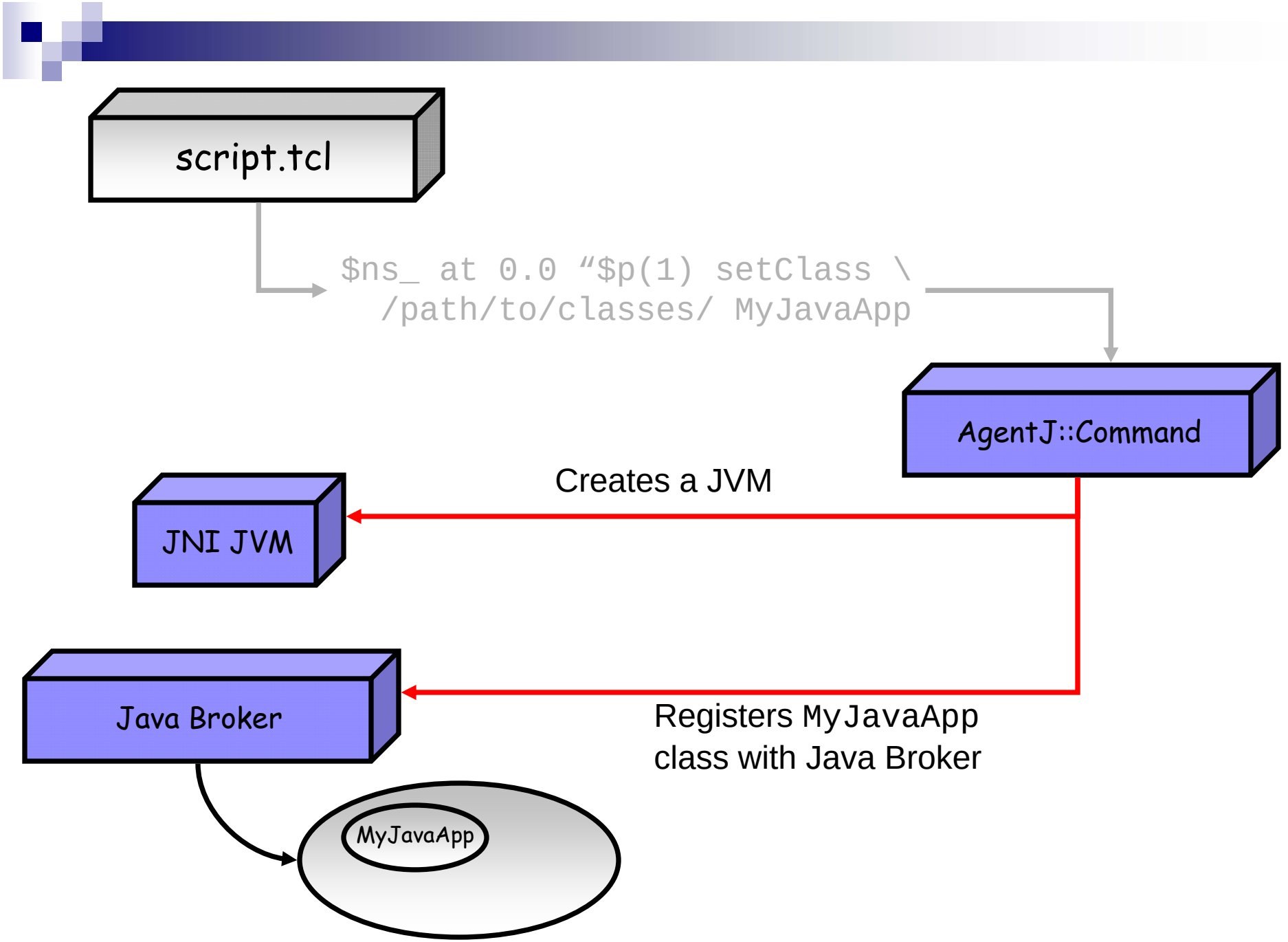
Figure by Ian Taylor

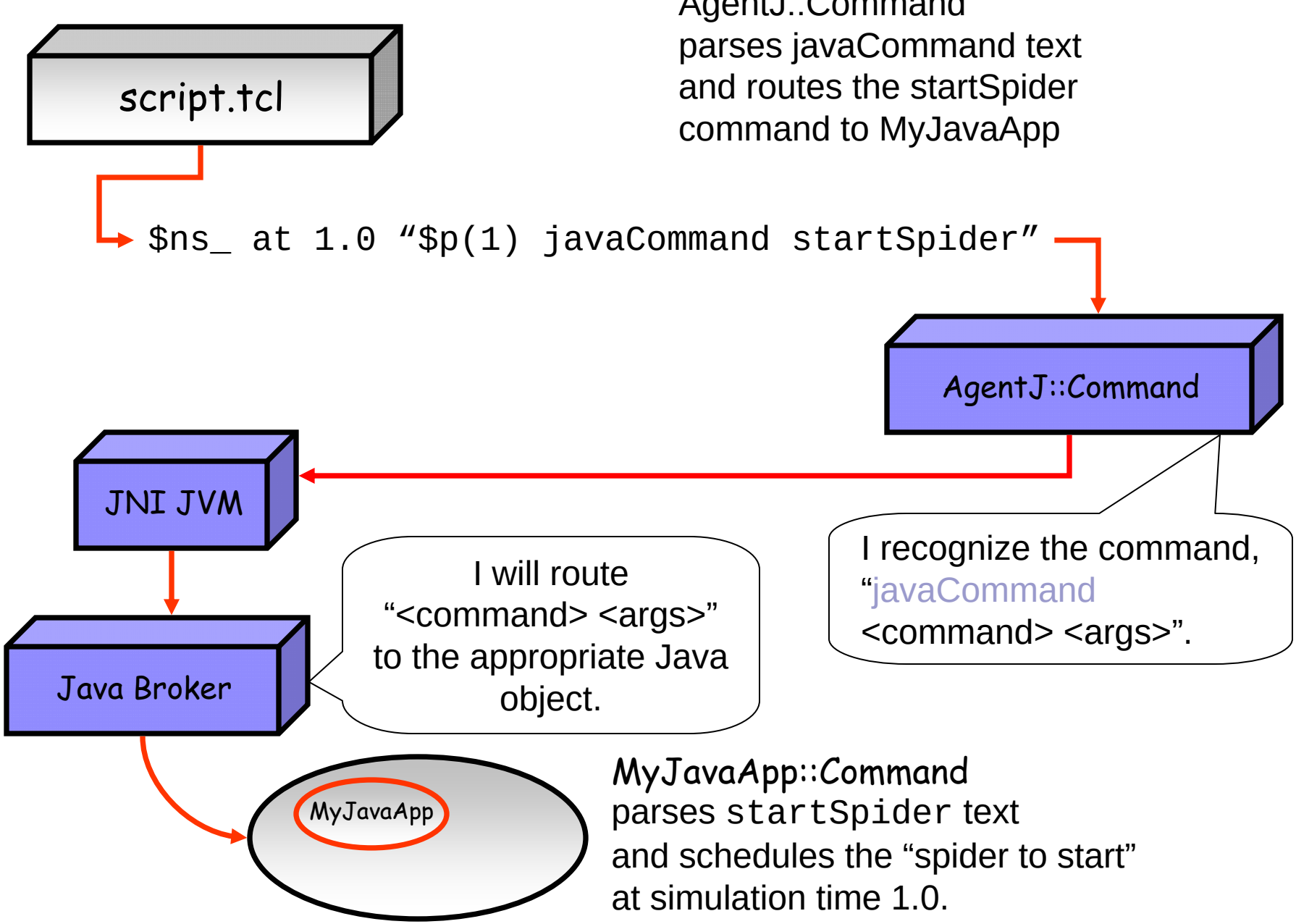
Sending commands from TCL to Java Agents



Sending commands from TCL to Java Agents




script.tcl
$ns_ at 0.0 "$p(1) setClass \
/path/to/classes/ MyJavaApp
AgentJ::Command
Creates a JVM
JNI JVM
Java Broker
Registers MyJavaApp
class with Java Broker
MyJavaApp


AgentJ::Command
parses javaCommand text
and routes the startSpider
command to MyJavaApp
script.tcl
$ns_ at 1.0 "$p(1) javaCommand startSpider"
AgentJ::Command
JNI JVM
I recognize the command,
"javaCommand
<command> <args>".
I will route
"<command> <args>"
to the appropriate Java
object.
Java Broker
MyJavaApp
MyJavaApp::Command
parses startSpider text
and schedules the "spider to start"
at simulation time 1.0.

Sending commands from TCL to Java Agents

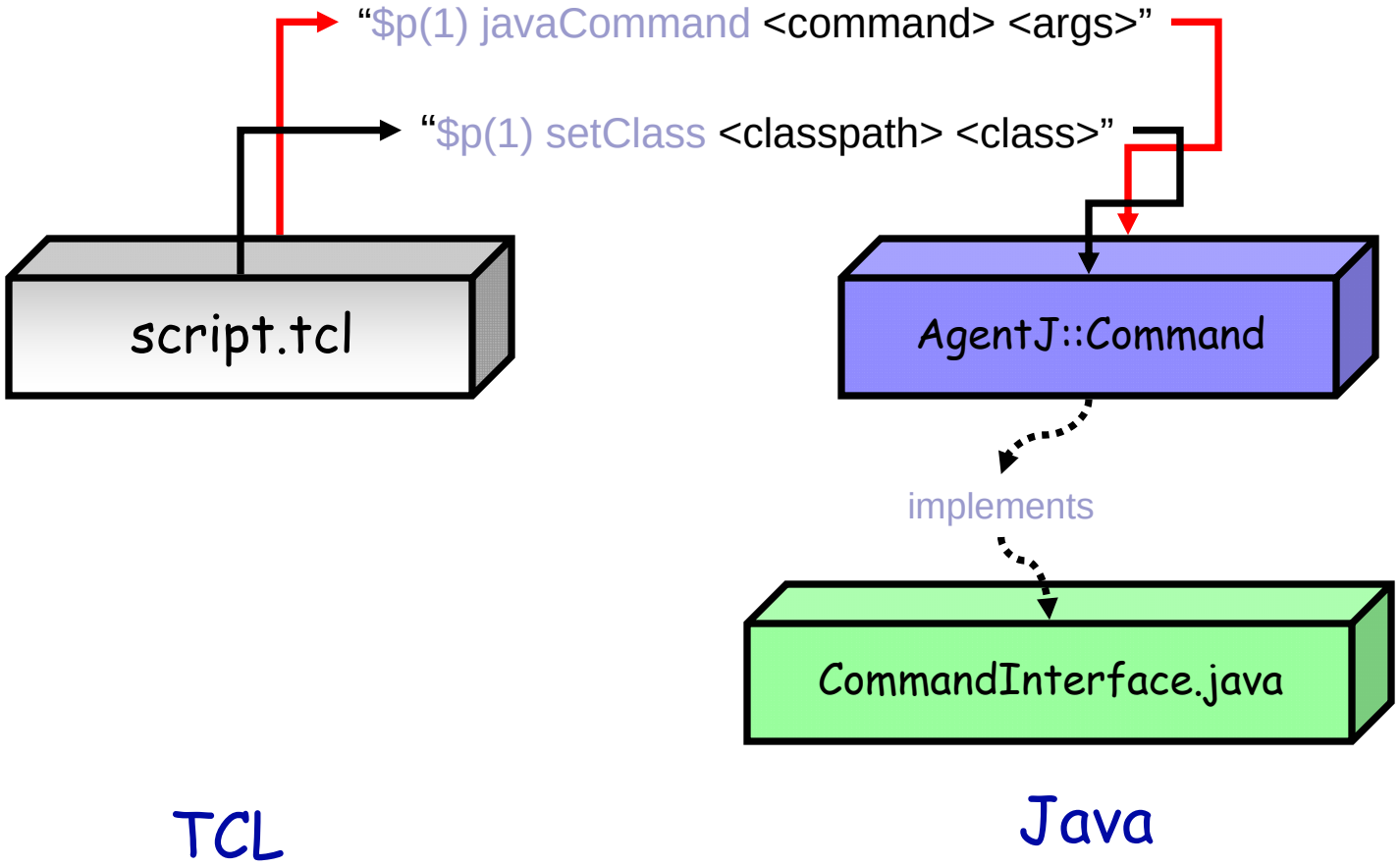


Figure by Ian Taylor

Developing Simulations

1. Create and attach java agent object to ns-2 node

```
set p(1) [new Agent/AgentJ]
$ns_ attach-agent $n(1) $p(1)
```

2. Define the class for the agent

```
$ns_ at 0.0 “$p(1) setClass /path/to/classes/ \
  MyJavaApp
```

3. Send the agent commands

```
$ns_ at 1.0 “P(1) javaCommand startSpider”
```

Writing Java Apps for PAI

- Use Java-PAI socket implementation
  - For existing applications, simply replace *java.net* with *pai.net*

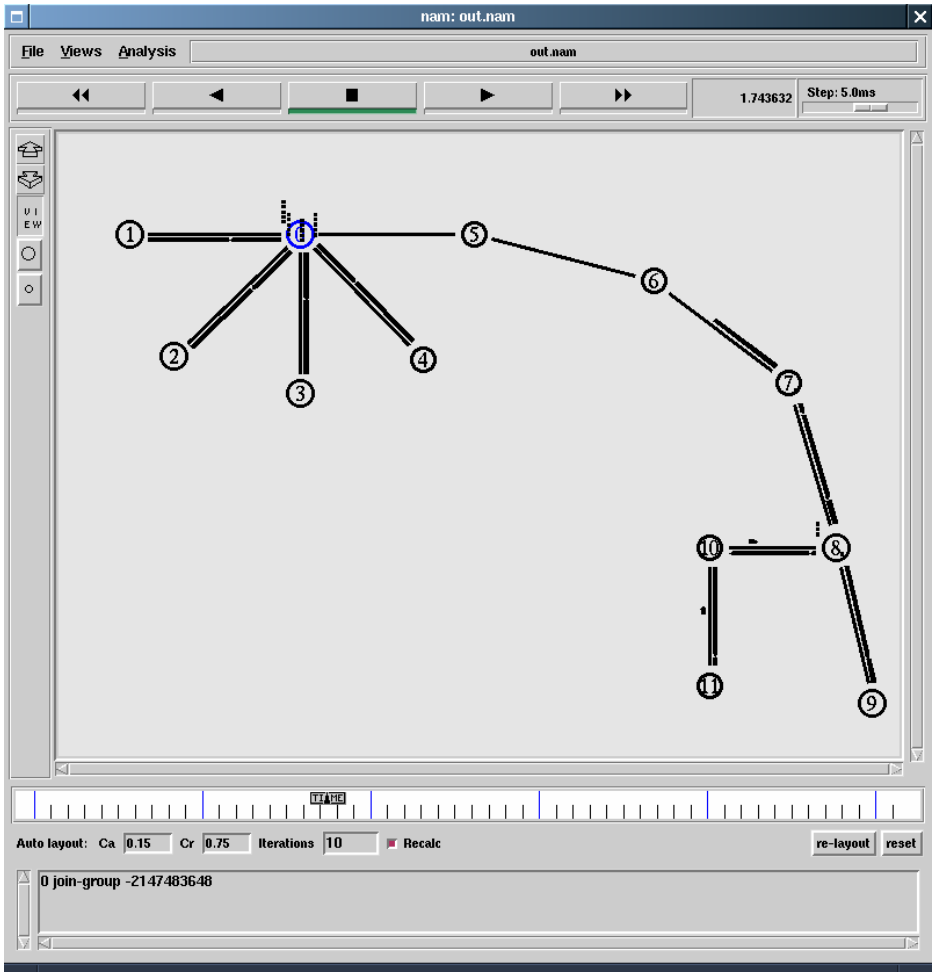
Writing Java Apps for PAI

- Remove threads (or “serialize” them)
  - Cannot simulate threads in ns2. The program counter in simulated applications needs to serially follow through the program. Spawned threads might not complete by the time the rest of the application’s code is simulated.
- Remove accesses to hardware devices (like serial ports)
  - Including some system calls
    - Such as, gettimeofday, java.util.date,
  - Cannot simulate in ns2. You can’t “spawn” new hardware!

Writing Java Apps for PAI

- Remove accesses to hardware devices (like serial ports)
  - Including some system calls
    - Such as, gettimeofday, java.util.date,
  - Cannot simulate in ns2. You can't "spawn" new hardware!

Demonstration of P2PS Service Discovery over a wired network



Peer-to-Peer Simplified

Questions?

PROTEAN
RESEARCH GROUP

CARDIFF UNIVERSITY
PRIFYSGOL CAERDYDD