Building a Scalable Infrastructure for Autonomous Adaptive Agents

Jun Suzuki, Ph.D.
jsuzuki@ics.uci.edu
www.ics.uci.edu/~jsuzuki/
netresearch.ics.uci.edu/bionet/
School of Information and Computer Science
University of California, Irvine

Overview

- Motivation to and overview of the Bio-Networking Architecture
- Design and implementation of the Bio-Networking platform
- Initial measurement results
- Adaptation through an evolutionary process
- Standardization effort

Introduction

- Computer network environment is seamlessly spanning locations engaged in human endeavor.
- Need a self-organizing network that supports
  - *Scalability*
    - in terms of # of objects and network nodes.
  - *Adaptability*
    - to changes in network conditions.
  - *availability/survivability*
    - from massive failures and attacks.
  - *Simplicity*
    - to design and maintain.

Autonomous Adaptive Agents

- One of the promising solution is to
  - deploy autonomous adaptive agents, and
  - construct network applications with them.
- Autonomous adaptive agent
  - *a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future.*
    - from *"Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents,"* (S. Franklin et al.)
      - http://www.msci.memphis.edu/~franklin/AgentProg.html

Our Observation and Goal

- A lot of research efforts have
  - successfully clarified autonomous adaptive agents,
  - showed they work well in many applications.
- However, the number of large-scale agent systems is currently very limited
  - Even in agent simulation systems, the scale of agents involved is often kept small, except several exceptions.
  - The scale of agent systems running on actual networks is usually much smaller.
    - e.g. The claim that Auctionbot is scalable is supported by an experiment with only 90 agents.
- Our goal
  - use autonomous adaptive agents, beyond simulations, for Internet-based distributed computing.

Our Approach

- Our approach
  - Design an architecture for autonomous adaptive agents
  - Implement an infrastructure to support our architecture and agents.
- *Architecture*, the Bio-Networking Architecture
  - models autonomous adaptive agents after several biological concepts and mechanisms.
- *Infrastructure*, the Bio-Networking Platform
  - a middleware platform that aids developing, deploying and executing our biologically-inspired agents by providing a rich set of reusable software components.

The Bio-Networking Architecture

The Bio-Networking Architecture

- Approach
  - *apply biological concepts and mechanisms to the design of autonomous adaptive agents (network application design)*

- motivated by the observation that the desirable properties in future network applications (e.g. scalability and adaptability) have already been realized in various biological systems.
  - e.g. bee colony, bird flock, fish school, etc, etc.

- Designed to allow for deploying large-scale, highly distributed and dynamic network applications using autonomous adaptive agents.

Biological Concepts Applied

- Decentralized system organization
  - Biological systems
    - consist of autonomous entities (e.g. bees in a bee colony)
    - no centralized (leader) entity (e.g. a leader in a bird flock)
      - Decentralization increases scalability and survivability of biological systems.
  - The Bio-Networking Architecture
    - biological entities = cyber-entities (CEs)
      - the smallest component in an application
      - provides a functional service related to the application
      - autonomous with simple behaviors
        - replication, reproduction, migration, death, etc.
        - makes its own behavioral decision according to its own policy
    - no centralized entity among CEs

- Emergence
  - Biological systems
    - Useful group behavior (e.g. adaptability and survivability) emerges from autonomous local interaction of individuals with simple behaviors.
      - i.e. not by direction of a centralized (leader) entity
      - e.g. food gathering function
        - When a bee colony needs more food, a number of bees will go to the flower patches to gather nectar.
        - When food storage is near its capacity, only a few bees will leave the hive.
  - The Bio-Networking Architecture
    - CEs autonomously
      - sense local/nearby environment
        - e.g. existence of neighboring CEs, existence/movement of users, workload, availability of resources (e.g. memory space), etc.
      - invoke behaviors according to the condition in a local/nearby environment
      - interacts with each other

- Lifecycle
  - Biological systems
    - Each entity strives to seek and consume food for living.
    - Some entities replicate and/or reproduce children with partners.
  - The Bio-Networking Architecture
    - Each CE stores and expends *energy* for living.
      - gains energy in exchange for providing its service to other CEs
      - expends energy for performing its behaviors, utilizing resources (e.g. CPU and memory), and invoking another CE's service.
    - Each CE replicates itself and reproduce a child with a partner.

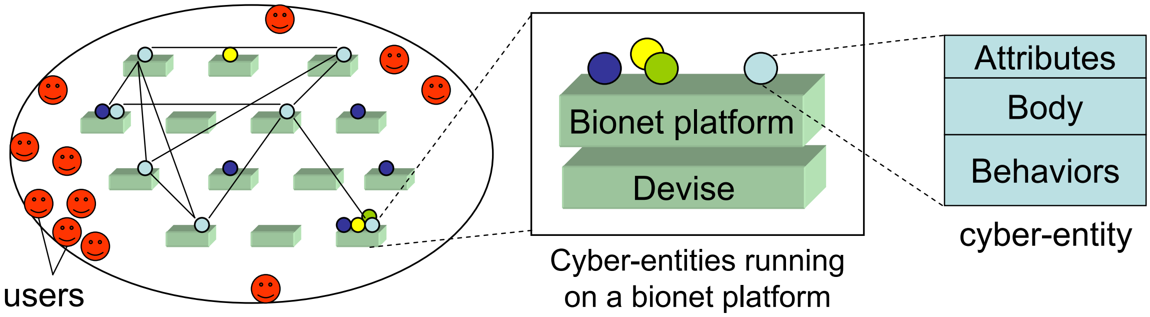
- Evolution
  - Biological system
    - adjusts itself for environmental changes through species diversity and natural selection
  - The Bio-Networking Architecture
    - CEs evolve by
      - generating behavioral diversity among them, and
        - » CEs with a variety of behavioral policies are created by human developers manually, or through mutation (during replication and reproduction) and crossover (during reproduction)
      - executing natural selection.
        - » death from energy starvation
        - » tendency to replicate/reproduce from energy abundance

- Social networking
  - Biological systems (social systems)
    - Any two entities can be linked in a short path through relationships among entities.
      - not through any centralized entity (e.g. directory), rather in a decentralized manner.
      - six decrees of separation
  - The Bio-Networking Architecture
    - CEs are linked with each other using *relationships*.
      - A relationship contains some properties about other CEs (e.g. unique ID, name, reference, service type, etc.)
    - Relationships are used for a CE to search other CEs.
      - Search queries originate from a CE, and travel from CE to CE through relationships.

Key Features of Cyber-entities

- Decentralized
  - No centralized entity (e.g. directory) on networks
- Autonomous
  - No intervention from other cyber-entities
- Adaptive
  - through an evolutionary process
- Self-describing
  - through a set of descriptive information

CE’s Structure and Behaviors



- Attributes
  - ID
  - Relationship list
  - Age
  - …etc.
- Body
  - Executable code
  - Non-executable data

- Behaviors
  - Energy exchange and storage
  - Migration
  - Replication and reproduction
  - Death
  - Relationship establishment
  - Social networking (discovery)
  - Resource sensing

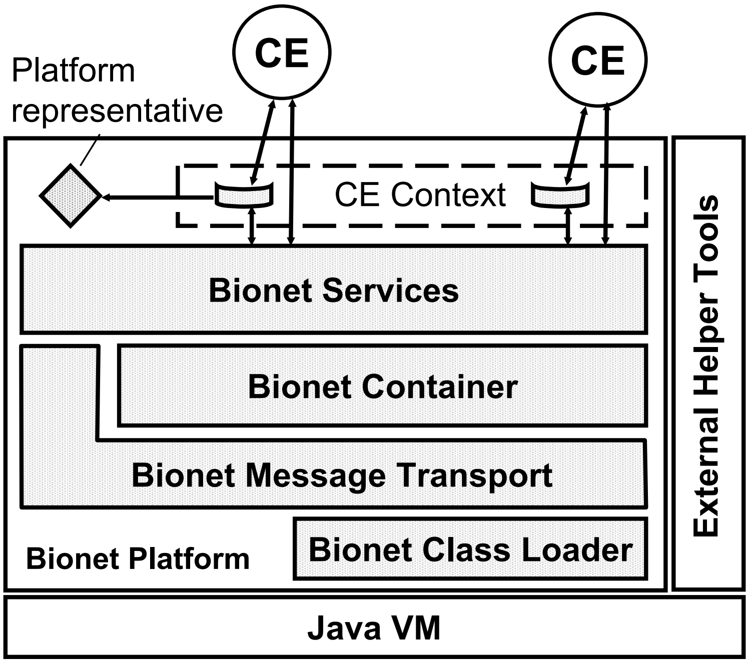
The Design and Implementation of the Bio-Networking Platform

Design Approach

- Separate cyber-entity (CE) and Bio-Networking Platform (bionet platform)
  - Cyber-entity (CE)
    - mobile object (agent) that provides any service logic
  - Bionet platform
    - middleware system for deploying and executing cyber-entities

- Identify the common networking, operating and biological functionalities required to deploy and execute CEs.
  - e.g. I/O, concurrency, messaging, network connection management, reference management, etc.
  - e.g. energy management, relationship maintenance, migration, replication, reproduction, etc.
- Design and implement those platform functionalities as a set of reusable objects.
- Implement CE and bionet platform in Java
- Empirically measures the platform functionalities.

Architecture



A *Cyber-entity (CE)* is an autonomous mobile object. CEs communicate with each other using FIPA ACL.

A *platform rep* keeps references to bionet services and container.

A *CE context* provides references to available bionet services.

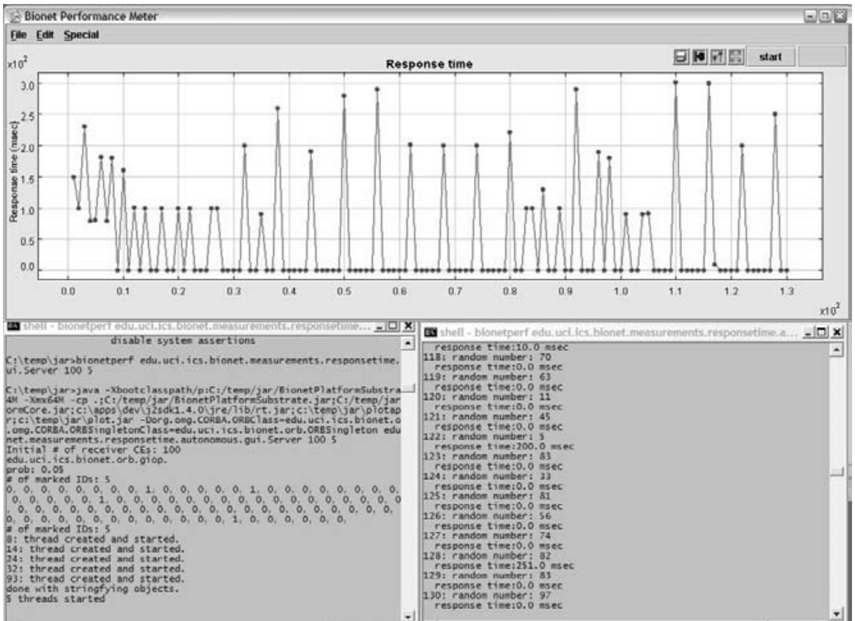
Bionet services are runtime services that CEs use frequently.

Bionet container dispatches incoming messages to target CEs.

Bionet message transport takes care of I/O, low-level messaging and concurrency.

Bionet class loader loads byte code of CEs to Java VM.

- An example of external tools
  - A graphical performance monitoring tool



```
                   disable system assertions

C:\temp\jar>bionetperf edu.uci.ics.bionet.measurements.responsetime.
ui.Server 100 5

C:\temp\jar>java -Xbootclasspath/p:C:/temp/jar/BionetPlatformSubstra
4M -Xmx64M -cp .;C:/temp/jar/BionetPlatformSubstrate.jar;C:/temp/jar
ormCore.jar;c:\apps\dev\j2sdk1.4.0\jre/lib/rt.jar;c:\temp\jar\plotap
r;c:\temp\jar\plot.jar -Dorg.omg.CORBA.ORBClass=edu.uci.ics.bionet.o
.omg.CORBA.ORBSingletonClass=edu.uci.ics.bionet.orb.ORBSingleton edu
net.measurements.responsetime.autonomous.gui.Server 100 5
Initial # of receiver CEs: 100
edu.uci.ics.bionet.orb.giop.
prob: 0.05
# of marked IDs: 5
0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
# of marked IDs: 5
8: thread created and started.
14: thread created and started.
24: thread created and started.
32: thread created and started.
93: thread created and started.
done with stringfying objects.
5 threads started
```

```
  response time:10.0 msec
118: random number: 70
  response time:0.0 msec
119: random number: 63
  response time:0.0 msec
120: random number: 11
  response time:0.0 msec
121: random number: 45
  response time:0.0 msec
122: random number: 5
  response time:200.0 msec
123: random number: 83
  response time:0.0 msec
124: random number: 33
  response time:0.0 msec
125: random number: 81
  response time:0.0 msec
126: random number: 56
  response time:0.0 msec
127: random number: 74
  response time:0.0 msec
128: random number: 82
  response time:251.0 msec
129: random number: 83
  response time:0.0 msec
130: random number: 97
  response time:0.0 msec
```

Status

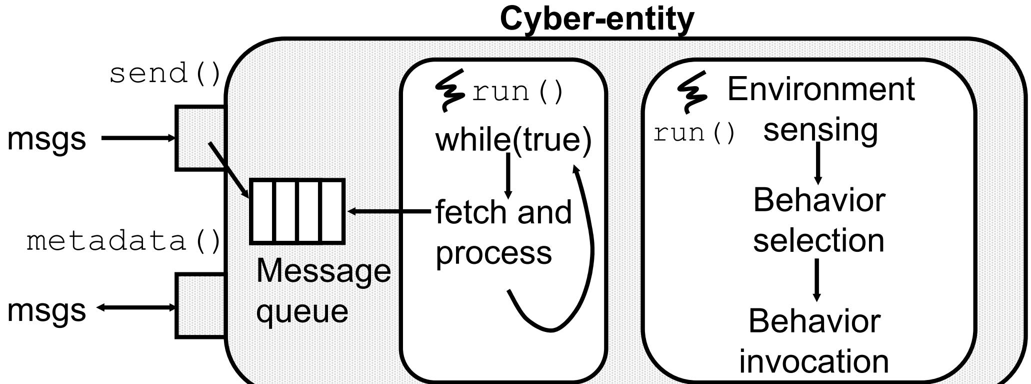
- Every platform component has been implemented.
  - The current code base
    - contains approx. 38,700 semicolons.
    - is the work of one full-time research staff and five part-time undergraduate students.
    - was implemented and tested on Windows 2000/XP PCs.
    - was ported onto Solaris in under a week of part-time.
      - primarily thanks to Java's portability
    - has been open for public use at UC Irvine since 2002.
    - will be released soon for researchers who explore the design space of autonomous adaptive agents and investigate them on actual networks.
- Started an initial set of measurements

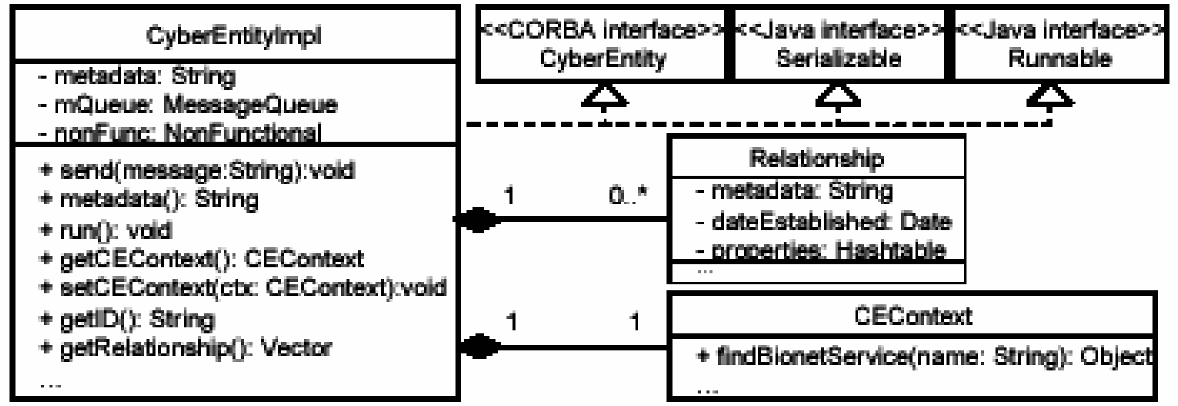
Cyber-entity

- CEs communicate with each other through:
  -
    ```
    interface CyberEntity {
         oneway void send(in string message);
         string metadata(); };
    ```
  - **`send()`**
    - used to send a message in an asynchronous (non-blocking) manner.
    - Messages are formatted with a subset of FIPA ACL with some extensions.
  - **`metadata()`**
    - used to obtain cyber-entity's attributes (self-descriptive information).
    - The mandatory attributes
      - cyber-entity's GUID (globally unique ID)
      - cyber-entity's reference
      - type of service that the cyber-entity provides, and
      - the energy units that the cyber-entity requires to provide its service.
    - CEs can specify any additional info as their optional attributes.

- A GUID is a 32-digits string data made from hexadecimal representations of
  - IP address
  - JVM identity hash code
    - » through calling `System.identityHashCode()`
  - the current time in milliseconds, and
  - a random number.
- A cyber-entity’s reference is formatted as a stringfied CORBA IOR.
- Attributes are represented as name-value pairs based on the OMG constraint language.
  - **`GUID='sti3sdr98rd56fn...'`**
  - **`ref='IOR:daforimklcmd...'`**
  - **`serviceType='HTTP/1.1'`**
  - **`serviceCost=100.0`**

- When a CE receive a message through `send()`, it put the message to its message queue.
- Each CE uses 2 threads to:
  - (1) fetch and process a message
  - (2) behave
    - sense the nearby environment,
    - identify behaviors suitable for the current environment condition, and
    - invoke the most suitable behavior





CyberEntityImpl

- metadata: String
- mQueue: MessageQueue
- nonFunc: NonFunctional

+ send(message:String):void
+ metadata(): String
+ run(): void
+ getCEContext(): CEContext
+ setCEContext(ctx: CEContext):void
+ getID(): String
+ getRelationship(): Vector
...

<<CORBA interface>>
CyberEntity

<<Java interface>>
Serializable

<<Java interface>>
Runnable

Relationship

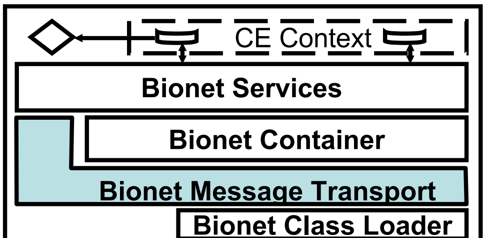
- metadata: String
- dateEstablished: Date
- properties: Hashtable
...

CEContext

+ findBionetService(name: String): Object
...

Bionet Message Transport

- Bionet message transport abstracts low-level networking details such as I/O, concurrency, messaging, network connection management.
  - Marshaling/unmarshaling messages issued by a CE
    - IIOP version 1.1 used currently
  - TCP connection setup and management
  - Message delivery on a TCP connection
    - One-to-one messaging, currently
    - One-to-many broadcasting/multicasting (future work)
  - Threading (thread pooling) to accept incoming messages



CE Context

Bionet Services

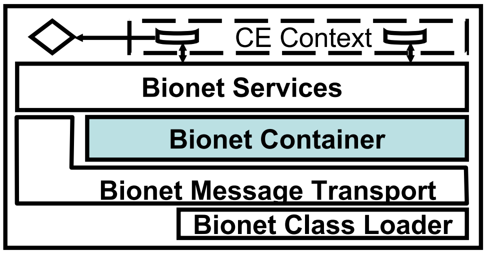
Bionet Container

Bionet Message Transport

Bionet Class Loader

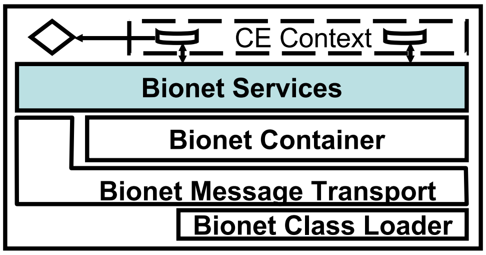
Bionet Container

- Bionet container
  - contains a table listing all the CEs running on the same platform.
  - complies with the interfaces of the CORBA POA to
    - demultiplex incoming messages,
    - dispatch incoming messages to target CEs,
    - create CE references.
  - keeps track of the current traffic load by counting
    - the size of incoming IIOP messages
    - the number of method dispatches.



Bionet Services

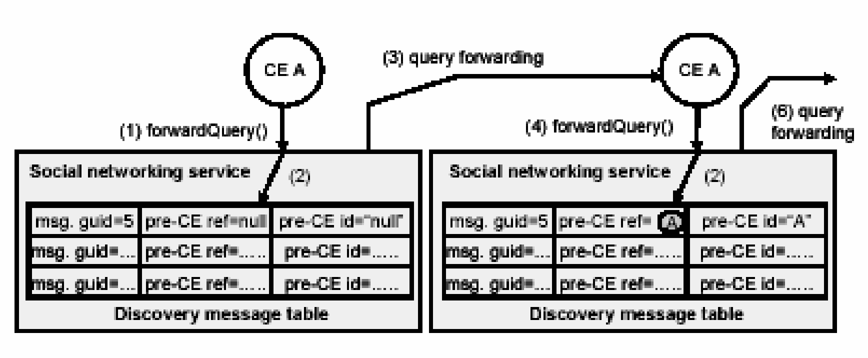
- CEs use bionet services to invoke their behaviors.
  - e.g. bionet lifecycle service when a CE replicates
- Each bionet platform provides 8 bionet services



- Bionet Lifecycle Service
  - – used to initialize a CE.
  - – maintains a thread pool that contains threads assigned to autonomous CEs
  - – allows a CE to replicate itself.
  - – allows a CE to reproduce a child CE with a partner
  - – Mutation and crossover during replication and reproduction
- Bionet Relationship Management Service
  - – allows a CE to establish, examine, update and eliminate their relationships with other CEs.

- Bionet Energy Management Service
  - – keeps track of energy level of the CEs running on a local platform.
  - – allows a CE to pay energy amounts for
    - invoking a service provided by another CE,
    - using resources, and
    - performing behaviors (i.e. invoking a bionet service).
- Bionet Resource Sensing Service
  - – allows CEs to sense the type, amount and unit cost of available resources.
    - CPU cycles and memory space
- Bionet CE Sensing Service
  - – allows a CE to discover other CEs running on the same platform.

- Bionet Pheromone Emission/Sensing Service
  - allows a CE to leave its pheromone (or trace) on a local platform when it migrates to another platform
    - so that other CEs can find the CE at a destination platform
- Bionet Migration Service
  - allows a CE to migrate to another bionet platform.
- Bionet Social Networking Service
  - allows a CE to search other CEs through their relationships.



msg. guid=5	pre-CE ref=null	pre-CE id="null"
msg. guid=...	pre-CE ref=.....	pre-CE id=.....
msg. guid=...	pre-CE ref=.....	pre-CE id=.....

Discovery message table

msg. guid=5	pre-CE ref= A	pre-CE id="A"
msg. guid=...	pre-CE ref=.....	pre-CE id=.....
msg. guid=...	pre-CE ref=.....	pre-CE id=.....

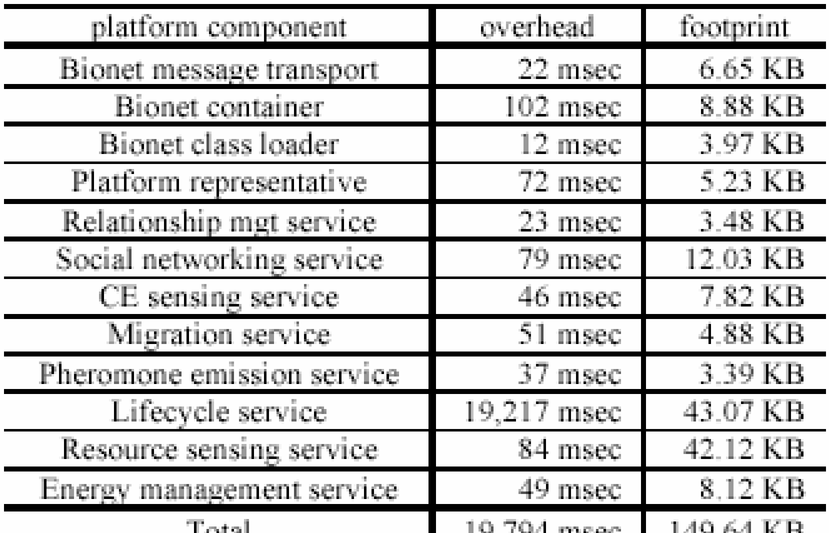
Discovery message table

- Search criteria are described with the OMG constraint language.
  - **`GUID='sti3sdr98rd56fn...'`**
  - **`serviceType=='HTTP/1.1' and serviceCost<100.0`**
  - **`serviceType=='HTTP/1.1' or serviceType=='HTTP/1.0'`**

Initial Measurement Results

Measurement (1)

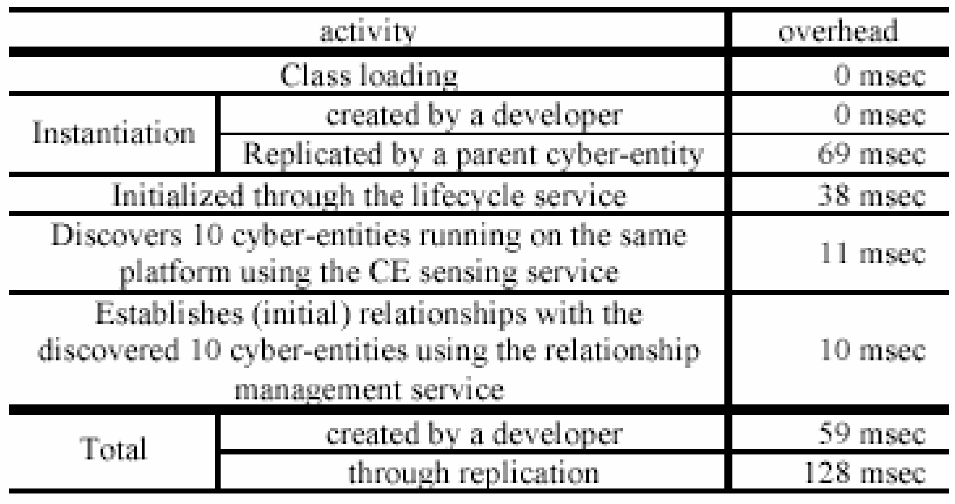
- Bootstrap overhead
  - how long does it take to initialize each platform components
- Bootstrap footprint
  - how much memory space is consumed for each component



platform component	overhead	footprint
Bionet message transport	22 msec	6.65 KB
Bionet container	102 msec	8.88 KB
Bionet class loader	12 msec	3.97 KB
Platform representative	72 msec	5.23 KB
Relationship mgt service	23 msec	3.48 KB
Social networking service	79 msec	12.03 KB
CE sensing service	46 msec	7.82 KB
Migration service	51 msec	4.88 KB
Pheromone emission service	37 msec	3.39 KB
Lifecycle service	19,217 msec	43.07 KB
Resource sensing service	84 msec	42.12 KB
Energy management service	49 msec	8.12 KB
Total	19,794 msec	149.64 KB

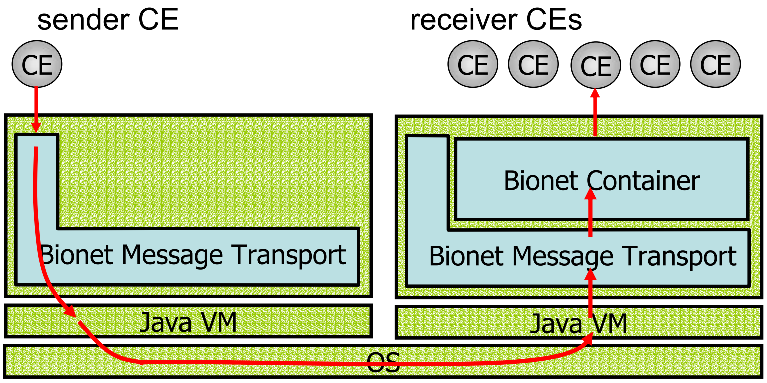
Measurement (2)

- Overhead to initialize and install a CE on a bionet platform



activity		overhead
Class loading		0 msec
Instantiation	created by a developer	0 msec
	Replicated by a parent cyber-entity	69 msec
Initialized through the lifecycle service		38 msec
Discovers 10 cyber-entities running on the same platform using the CE sensing service		11 msec
Establishes (initial) relationships with the discovered 10 cyber-entities using the relationship management service		10 msec
Total	created by a developer	59 msec
	through replication	128 msec

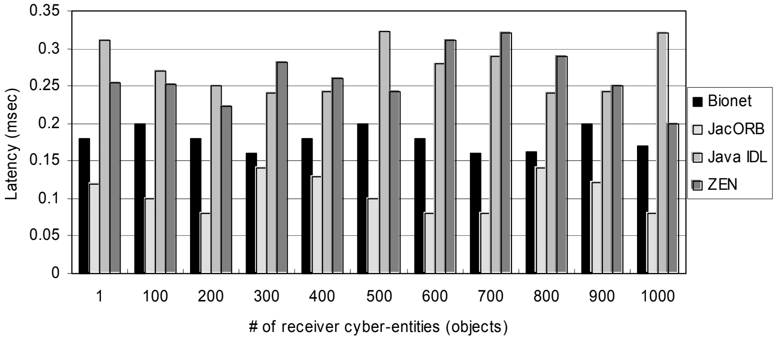
Measurement (3)



A sender CE dynamically selects a receiver CE at random, and sends an empty string message to the selected CE.

Measurements were conducted within a single machine with Java 1.4 VM, Win XP, and 1GHz Pentium 3 CPU. 64 MB heap allocated to each Java VM

- What we measured
  - Latency for message transmission between two CEs
    - How long it takes for a CE to send a message to another CE?
- Overhead sources to message transmission
  - Marshaling a message issued by a sender CE,
  - TCP connection setup,
  - message delivery on the connection,
  - message dispatching to a receiver CE, and
  - unmarshaling of an incoming message.

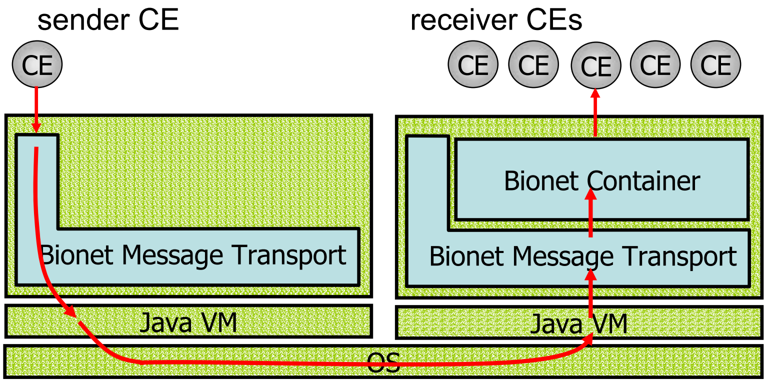


A single sender CE and a range of receiver CEs (1, 100, …1000) were deployed.

Bionet platform was compared with existing distributed object platforms implemented in Java.

- Measurement results and observations
  - Bionet message transport and container are fairly efficient and comparable with existing distributed object platforms.
    - Message transmission latency was 0.17 msec when 1,000 receiver CEs were deployed.
  - Bionet message transport and container are scalable in terms of the number of receiver CEs.
    - Latency is relatively constant when the number of CEs grows, rather than it increases linearly (the average of latency was 0.179 msec.).
      - In general, increasing the # of receiver CEs increases the effort to establish TCP connections to receiver CEs (in sender side) and demultiplex/dispatch incoming messages to target CEs (in receiver side).
    - Implementation techniques such as connection sharing and hash-based demultiplexing work well.

Measurement (4)



sender CE

receiver CEs

CE

CE CE CE CE CE

Bionet Message Transport

Bionet Container

Bionet Message Transport

Java VM

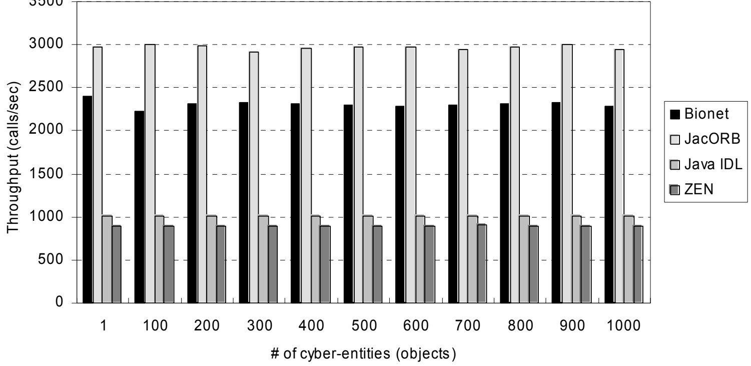
Java VM

OS

A sender CE selects a receiver CE at random before a measurement, and sends empty string messages to the selected CE.

Measurements were conducted within a single machine with Java 1.4 VM, Win XP, and 1GHz Pentium 3 CPU. 64 MB heap allocated to each Java VM

- What we measured
  - throughput of a CE
    - How many messages a CE can receive and process in a second?
- The throughput is dominantly affected by
  - message demultiplexing in bionet container.



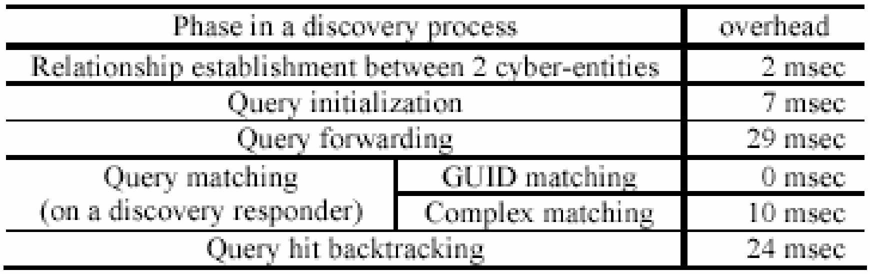
A single sender CE and a range of receiver CEs (1, 100, …1000) were deployed.

Bionet platform was compared with existing distributed object platforms implemented in Java.

- Measurement results and observations
  - The throughput of a CE on a bionet platform is competitive with existing distributed object platforms.
    - Throughput was 2279.99 messages/sec when 1,000 receiver CEs were deployed.
  - Bionet container are scalable in terms of the number of receiver CEs.
    - Increasing the number of receiver CEs increases the demultiplexing effort. Throughput remains relatively constant as the number of receiver CEs grows (2309.27 messages/sec in average), rather than it increases linearly.
    - Implementation technique of hash-based demultiplexing works well.

Measurement (5)

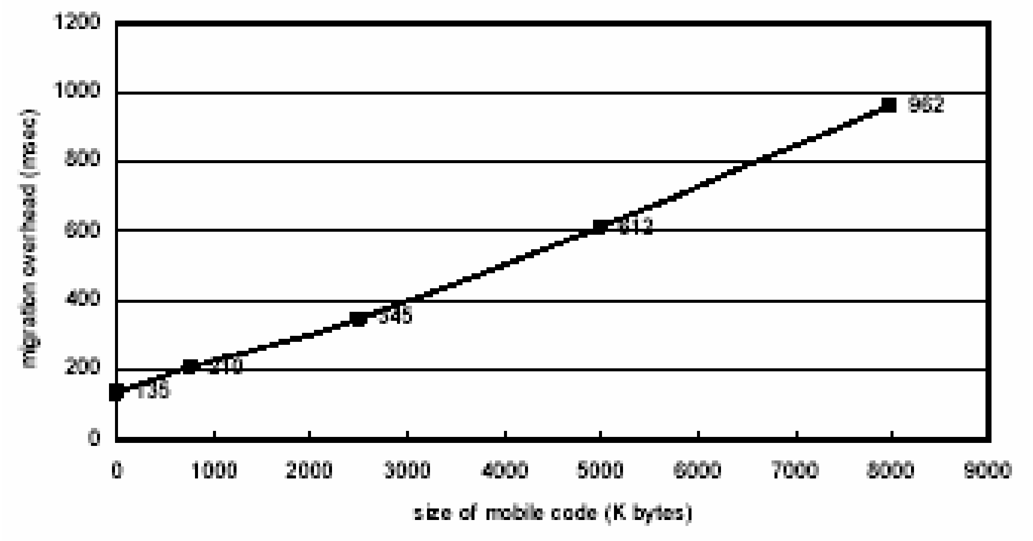
- Overhead in each of a discovery process



Phase in a discovery process		overhead
Relationship establishment between 2 cyber-entities		2 msec
Query initialization		7 msec
Query forwarding		29 msec
Query matching (on a discovery responder)	GUID matching	0 msec
	Complex matching	10 msec
Query hit backtracking		24 msec

Measurement (6)

- Overhead of a migration process



Adaptability of Cyber-entities through an Evolutionary Process

Adaptability of Cyber-entities

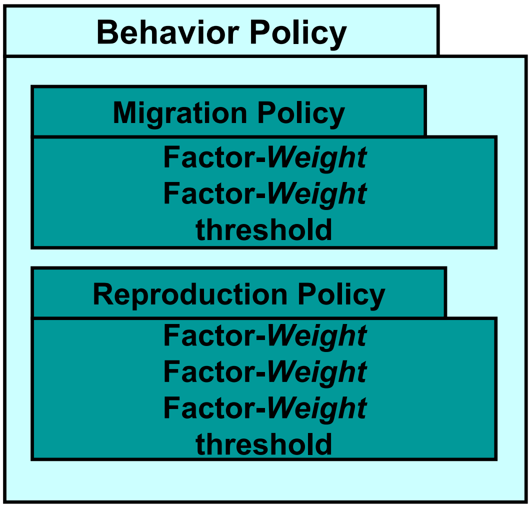
- Evolution as a means to reconfigure behaviors of cyber-entities
  - Biological entities adjust themselves for environmental changes through *behavioral diversity* and *natural selection*.
  - CEs evolve by
    - generating behavioral diversity among them, and
      - CEs with a variety of behavioral policies are created
        - » by human developers manually, or
        - » through *mutation* and *crossover* (automatically).
    - executing natural selection.
      - death from energy starvation
      - tendency to replicate/reproduce from energy abundance

Cyber-Entity's Behavior Policy

Each CE has its own policy for each behavior.

A behavior policy consists of *factors* (F), *weights* (W), and a *threshold*.

- If $\sum_i F_i \cdot W_i$ > threshold, then migrate.



Behavior Policy

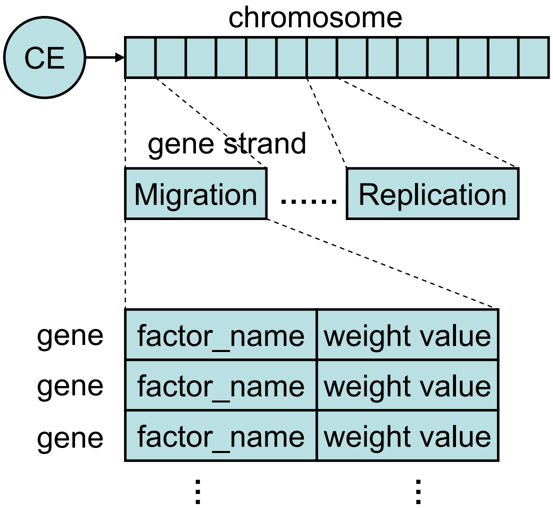
Migration Policy
Factor-*Weight*
Factor-*Weight*
threshold

Reproduction Policy
Factor-*Weight*
Factor-*Weight*
Factor-*Weight*
threshold

Example migration factors:

- *Migration Cost*
  - A higher migration cost (energy consumption) may discourage migration.
- *Distance to Energy Sources*
  - encourages CEs to migrate toward energy sources (e.g. users).
- *Resource Cost*
  - encourages CEs to migrate to a network node whose resource cost is cheaper.

Chromosome and Genes



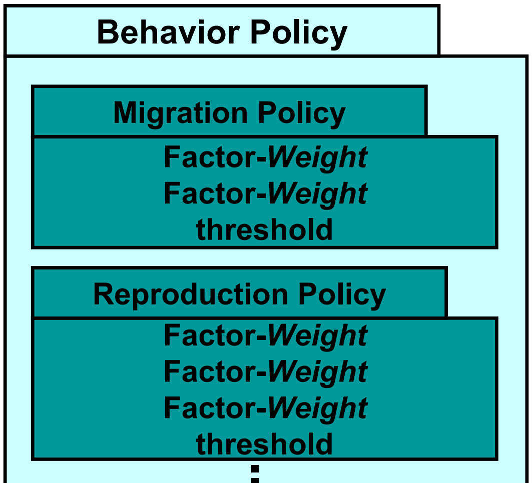
Each CE has its own chromosome.
It is given by a human developer or
contributed by its parent(s).

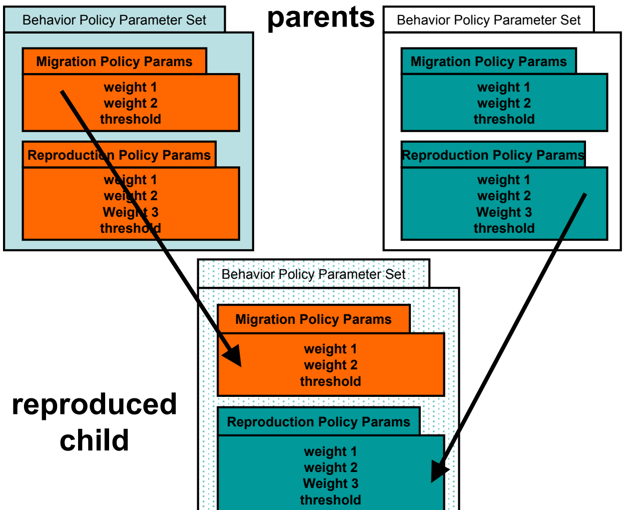
A chromosome consists of gene strands.
Each gene strand corresponds to
a behavior.

A gene strand consists of genes.
Each gene encodes factor name,
and weight value.
Weight values are mutable.

Mutation and Crossover

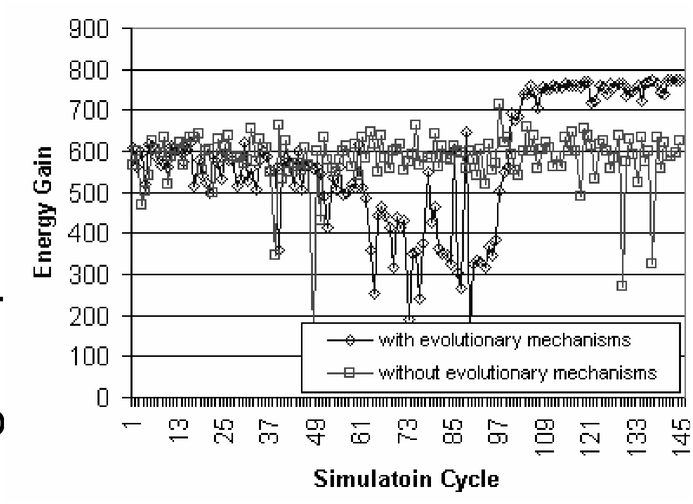
- Weight values in each behavior policy change dynamically through mutation.
- Mutation occurs during replication and reproduction.
- Crossover occurs during reproduction.
- A child CE inherits different behaviors from different parents through crossover.





A Simulation Result

- Users (energy sources move around network randomly.
- Evolutionary CEs gain more energy than non-evolutionary ones;
- Evolutionary CEs adap better to dynamic network conditions.
  - by moving closer to users and avoiding network nodes whose resource cost is expensive.
  - by increasing weight values of *distance-to-user* and *resource cost* factors.



Status

- Through simulations, we have already confirmed
  - Effectiveness of energy concept
  - Effectiveness of mutation and crossover
  - Adaptability of CEs through evolutionary reconfiguration mechanisms in dynamic networks
- Now, we are implementing evolutionary mechanisms within CEs.
- We will soon start adaptation experiments on actual networks.

Standardization effort

Standardization Effort at OMG

- The goals of the OMG Super Distributed Objects (SDOs) working group are to
  - provide a standard computing infrastructure that incorporates massive numbers of objects (SDOs) including hardware devices and software components
  - deploy them in a highly-distributed and ubiquitous environment, and
  - allow them to seamlessly interwork with each other.

Status

- The SDO RFI issued ('00), and responses gathered ('01)
  - from 10 organizations including UCI
- The SDO white paper published ('01)
  - by Hitachi, GMD Fokus and UCI
- The first RFP published (Jan. 02).
  - solicits resource data model for SDOs, discovery interfaces, etc.
- The initial proposals submitted (Sept. 02)
  - by Hitachi, GMD Fokus and UCI
    - 28 organizations on the voting list
- The revised joint proposal was submitted and adopted (through a vote-to-vote process) in March 2003.
  - by Hitachi, GMD Fokus and UCI
  - "PIM and PSM of SDOs" dtc/03-04-02

Bionet Platform as an Implementation of the SDO Spec

- The current bionet platform supports the PIM (UML model) and CORBA PSM (CORBA interfaces) in the SDO spec.
  - an SDO as a CE
  - organization as relationship between CEs

Future Work

- Extended set of measurements for the bionet platform
- Adaptation experiments on actual networks
- Reconfigurability of bionet platform comonents
- Deployment of the bionet platform on PDAs and/or cell phones
- Efficient algorithms for decentralized discovery of CEs
- Dynamic composition of CEs to provide a collective application
- Model-driven development for agents (CEs)
  - to automatically (or semi-automatically) source code skeleton for a CE from models
- Finalization of the SDO spec

Thank you

- All the papers/documents related to the Bio-Networking Architecture are available at:
  - **netresearch.ics.uci.edu/bionet/**
  - **netresearch.ics.uci.edu/bionet/resources/platform/**
- Sponsors
  - NSF (National Science Foundation)
  - DARPA (Defense Advanced Research Program Agency)
  - AFOSR (Air Force Office of Science Research)
  - State of California (MICRO program)
  - Hitachi
  - Hitachi America
  - Novell
  - NTT (Nippon Telegraph and Telephone Corporation)
  - NTT Docomo
  - Fujitsu
  - NS Solutions Corporation