Middleware Support for Super Distributed Autonomic Services in Pervasive Networks*

Junichi Suzuki and Tatsuya Suda
University of California, Irvine
{jsuzuki, suda}@ics.uci.edu

Abstract

This paper describes our empirical research effort to design, implement and deploy a middleware platform that addresses several key issues in pervasive computing. We designed an architecture, called the Bio-Networking Architecture, which models a pervasive network application as a collection of autonomic agents designed after several biological concepts and mechanisms. The biologically-inspired agents inherently address the key issues in pervasive computing, and our middleware platform aids developing and executing the agents on networks. We identify a set of requirements to the middleware platform through analyzing the features of our agents, and describe the design and implementation of the platform, showing how the platform satisfies the identified requirements. We also present some measurement results to illustrate scalability and efficiency of the platform.

1. Introduction

As computing devices and networks become more pervasive, the computing landscape is evolving into an environment in which a huge number of networked computing devices sense, interact with and control the physical world in such a way that the physical world is merged and augmented with the virtual world [1, 2]. In order to make this vision a reality, literatures have identified several key issues for network applications and software platforms in the area of pervasive computing [3−6]. They address that software platforms need to allow application components to move around the network, discover other components dynamically, adapt to dynamic changes in environment (e.g. workload) and scale well (e.g. in terms of the size of applications). They also need to make the development and deployment of application components more productive (i.e. faster and easier).

This paper describes our research effort to investigate a software platform that hosts each pervasive network application as a collection of *autonomic agents* [7−9] and allows pervasive network applications (i.e. autonomic agents) to support the above key requirements; *mobility*, *dynamic discovery*, *adaptability*, *scalability* and *ease of development and deployment*. The agents are designed after several biological concepts and mechanisms in our novel network application architecture called the Bio-Networking Architecture [10, 11], which is motivated by the observation that the above key requirements in pervasive computing have already been realized in various biological systems. We overview the features of our autonomic agents and identify a set of functional requirements to a software platform for them, called the Bio-Networking Platform (or bionet platform). The bionet platform is a middleware that aids developing and deploying large-scale, highly-distributed and dynamic applications (i.e. autonomic agents) in pervasive networks by abstracting low-level operating and networking details (e.g. concurrency and messaging) and providing a series of high-level runtime services. We describe the design and implementation of the bionet platform, showing how the platform satisfies the functional requirements derived from the features of our agents. We also present some of the measurement results to illustrate the efficiency and scalability of the bionet platform.

This paper is organized as follows. Section 2 presents the features of our agents. Section 3 describes the design and implementation of our agents as well as the bionet platform. Measurement results are shown in Section 4. In Sections 5 and 6, we conclude with comparison with existing work and future work.

2. Assumed Features of Autonomic Agents

In the Bio-Networking Architecture, each autonomic agent, called *cyber-entity*, consists of attributes, body and behaviors [10]. Attributes carry descriptive information of a cyber-entity (e.g. identifier). A body implements a cyber-entity's functional service(s). Behaviors implement non-functional biological actions such as reproduction and migration. Each cyber-entity lives on a specific bionet platform to execute its service implemented in its body. A bionet platform runs on each network node. Cyber-entities maintain the following four key features.

(1) Decentralized. A network application is modeled as a decentralized collection of cyber-entities in the Bio-Networking Architecture. This is analogous to a bee colony (an application) consisting of multiple bees (cyber-entities). The advantages of decentralization are scalability and fault tolerance [14]. Centralized systems can fail when central entities (e.g. directory server) are overwhelmed, but decentralized systems can survive by

* Research supported by the NSF through Grants ANI-0083074 and ANI-9903427, by DARPA through Grant MDA972-99-1-0007, by AFOSR through Grant MURI F49620-00-1-0330, and by grants from the California MICRO program, Hitachi, Hitachi America, Novell, Nippon Telegraph and Telephone Corporation (NTT), NTT Docomo, Fujitsu, NS Solutions Corporation, and DENSO IT Laboratory.

spreading the load [15]. Central entities also suffer from mobility of agents. They cannot eventually keep track of agents if they often join and leave the network [16]. Decentralized systems have an organizational advantage as well. Users need no complicated setup work; they can simply develop and run their agents without knowing any central coordination. This lowers the barrier for users to develop and deploy agents. Keeping these advantages in our mind, the Bio-Networking Architecture is designed not to assume any central entities on the network.

(2) Autonomous. Autonomy is the ability of agents to act without any interventions from their users and other agents [17]. Autonomous agents are goal-oriented and control themselves proactively [18]. Cyber-entities are autonomous in the sense that each of them has its own goal (e.g. staying close to users and living long), senses surrounding network conditions, and performs its behaviors, according to the sensed network conditions, which will support future goal achievement [11]. Our previous simulation study has confirmed the desirable system properties (e.g. adaptability) emerge through cyber-entities' autonomous behavior invocations [11].

(3) Adaptive. Adaptability is the ability of agents to increase their fitness to environment. Cyber-entities adapt themselves to environmental changes in short-term and long-term fashions. The short-term adaptation is achieved by performing behaviors according to the current network conditions [11, 13]. For example, a cyber-entity may migrate to a neighboring platform when traffic volume grows or resource availability becomes scarce. The long-term adaptation is achieved by applying biological evolutionary process. Cyber-entities evolve by generating behavioral diversity and executing natural selection [12]. Behavioral diversity means that it is likely different cyber-entities implement different policies on their behaviors. It is generated through mutation and crossover, which dynamically modify behavior policies during replication and reproduction. Natural selection is executed based on the concept of *energy*. Cyber-entities gain energy in exchange for performing their services, and expend energy to consume resources such as CPU cycles and memory space. The abundance and scarcity of stored energy affects contributes to the natural selection process. For example, energy abundance is an indication of higher demand for the cyber-entity; thus the cyber-entity may be designed to favor reproduction in response to higher level of energy. Energy scarcity (an indication of lack of demands or ineffective behavior policies) may eventually cause the cyber-entity's death. Our previous simulation work has shown our evolutionary process allows cyber-entities to adapt to dynamic environmental changes (e.g. changes in workload, users' location and resource availability) [12].

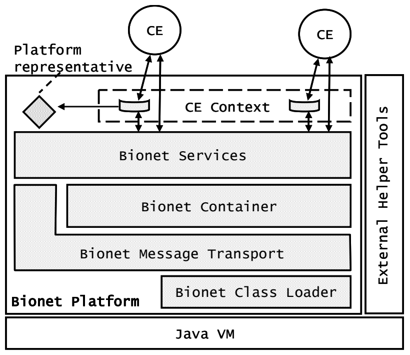

Figure 1. Architecture of the bionet platform

(4) Self-descriptive. In order to make agents autonomous and decentralized, they need to be loosely coupled with each other. As a result, the agents that an agent interacts with may not exist when it is developed, and they may not always be available in the future, for example, due to their migrations. Therefore, agents should be able to dynamically discover and interact with other agents without recompiling or changing any lines of code. In the Bio-Networking Architecture, each cyber-entity keeps its own descriptive information as attributes, and makes it available to other cyber-entities. It also maintains relationships with other cyber-entities. A relationship is established between two cyber-entities, and it contains attributes about a peer cyber-entity. With relationships and attributes, cyber-entities dynamically discover others and interact with each other [19].

Through the above four features of cyber-entities, the Bio-Networking Architecture inherently addresses the key issues in pervasive computing (see Section 1), such as mobility, dynamic discovery, adaptability, scalability and ease of development and deployment.

3. The Bio-Networking Platform

Given an initial set of successful simulation results [11, 12, 13, 19], we built the bionet platform in order to host the Bio-Networking Architecture on the real network for empirically evaluating the features of cyber-entities. It is implemented in Java, and each platform runs on a Java virtual machine (JVM) atop a network node. It consists of five components (Figure 1).

A *platform representative* is an object that represents a bionet platform and runs on per-platform basis. It keeps a table listing all the bionet services and bionet container (see below) on a local platform with their names and references. It is initialized when a bionet platform boots.

A *CE context* is an entry point for a cyber-entity to access underlying bionet services. It examines if a bionet service requested by a cyber-entity is available, and if it is, the CE context returns a reference to the service. Each CE context performs this lookup for bionet services through the local platform representative. Each cyber-entity has its own CE context. A CE context is created and associated with a cyber-entity by the lifecycle service (one of the bionet services), when the cyber-entity is created, replicated or reproduced.

The *bionet services* provide a set of runtime services that cyber-entities use for performing their behaviors. Each bionet service implements one or more behaviors of cyber-entities. The behaviors the bionet services support are *energy exchange/storage*, *migration*, *replication and reproduction*, *relationship maintenance*, *discovery of cyber-entities* and *resource sensing*.

The *bionet message transport* abstracts low-level networking and operating details such as network I/O, concurrency, messaging and network connection management. The current bionet platform uses the CORBA IIOP 1.1 [20] to transmit messages on TCP.

The *bionet container* maintains references to the cyber-entities running on a local platform, and dispatches incoming messages to them. It also monitors the network traffic by counting the size of received IIOP packets and the number of message dispatches.

The *bionet class loader* is a custom class loader that extends JVM's system (default) class loader. It is used to dynamically load a cyber-entity's class definition into a JVM when it is newly created or completes a migration.

The current code base of the bionet platform contains approximately 29,700 semicolons, and is the work of one full-time research staff and six part-time students.

3.1. Design of Cyber-entities

Since the bionet platform uses Java as a programming implementation language and CORBA IIOP as a message transport protocol, a cyber-entity is designed as a Java object implementing a CORBA interface. Every cyber-entity implements the following CORBA interface.

```
interface CyberEntity {
  oneway send(in string message);
  string metadata();};
```

Cyber-entities use `send()` to communicate with each other in an asynchronous manner. The operation accepts a message from another cyber-entity as its parameter. We use a subset of the FIPA agent communication language for the message format. Due to space limitation, please see [19] for more details about the message format. The `send()` operation inserts a received message in cyber-entity's message queue (Figure2). The cyber-entity fetches the message to process it on an individual thread. When no message is available, the thread `waits` for a new message on the queue.

Each cyber-entity maintains another thread to perform its non-functional logic including environment sensing, behavior selection and behavior invocation

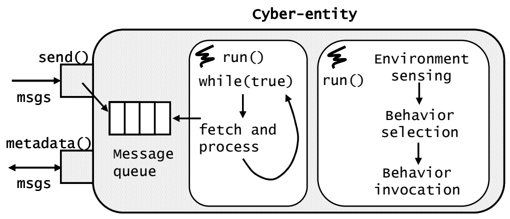

Figure 2. Internal design of a cyber-entity

(Figure 2). It is implemented as a subclass of `java.util.TimerTask` and executed at certain intervals. We assigned different threads to functional and non-functional aspects, because it is different how often these aspects need to be executed; the functional aspect should be executed immediately when a message is queued and the non-functional aspect can be executed on the order of seconds, minutes or even hours, depending on application requirements. Please note that it is beyond of the scope of this paper to describe non-functional aspect. Please see [9, 10, 11] for details about this issue.

The `metadata()` operation of `CyberEntity` is used to obtain a cyber-entity's attributes. The mandatory attributes that every cyber-entity must maintain are (1) the cyber-entity's GUID (globally unique ID)[1], (2) the cyber-entity's reference, (3) the type of service the cyber-entity provides, and (4) the energy units that the cyber-entity requires to provide its service. Cyber-entities can specify any other information as their optional attributes. Attributes are represented as name-value pairs based on the OMG constraint language [21]. A sample of mandatory attributes is described as follows:

```
GUID='sti3sdr98rd56fn...'
ref='IOR:daforimklcmd...'
serviceType='HTTP/1.1'
serviceCost=100.0
```

Figure 3 shows the design of the base class for cyber-entities, `CyberEntityImpl`. This class defines a set of variables and methods that are common among all the cyber-entities. Developers define their own cyber-entities by extending this class. We have proposed our designs of the bionet platform to Object Management Group (OMG) as the fundamental building blocks for Super Distributed Objects (SDOs) [30]. Super distribution means incorporating massive numbers of objects on highly distributed environments in a decentralized manner [30]. The goals of OMG SDOs are to represent heterogeneous hardware devices and software services as objects (SDOs) in a uniform object model, map them onto higher-level overlay networks and allow them seamlessly interwork with each other. OMG adopted our proposal and will publish the final version as its official standard specification in 2004. Figure 3 also shows how key components in the Bio-Networking Architecture (e.g.

[1] See [26] for the detailed design of GUID.

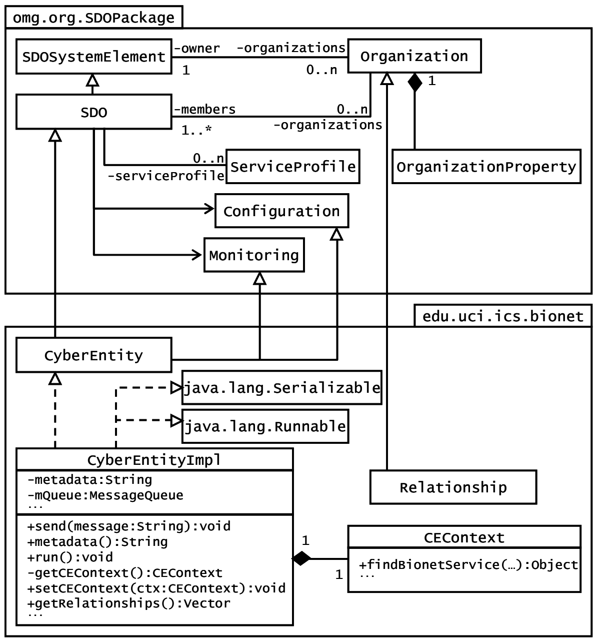

Figure 3: Class diagram around CyberEntityImpl

cyber-entity and relationship) are designed based on the specification.

3.2. Bionet Services

The bionet platform provides eight bionet services that cyber-entities use for performing their behaviors (Table 1). They are designed to support the key requirements in pervasive computing (see Section 1). Their designs are guided by five functional requirements derived from the features and behaviors of cyber-entities (see Section 2).

Each bionet service runs on per-platform basis. Since decentralization is a key design principle for us, we implemented all the bionet services in a decentralized manner; no centralized entities exist. We describe the design of bionet services along with the requirements.

(1) Relationship management. As described in Section 2, cyber-entities use their relationships to represent their acquaintances, discover other cyber-entities and interact with them. Therefore, the bionet platform provides the relationship management service, which allows cyber-entities to establish, examine, update and eliminate their relationships (Table 1). Each cyber-entity has a list of relationship objects, each of which represents a relationship with another cyber-entity. A relationship object contains the attributes of a partner cyber-entity. It can contain any additional information (e.g. keywords describing their partner cyber-entities).

When a cyber-entity establishes a relationship with another one, it calls a relationship management service with its partner's GUID (global unique identifier) and/or

Table 1. A list of the bionet services

Name	Functionality
Relationship management	allows cyber-entities to establish, examine, update and eliminate their relationships.
Social networking	allows cyber-entities to locate other cyber-entities through their relationships with their search criteria.
CE sensing	allows cyber-entities to locate the cyber-entities running on the local platform.
Migration	allows cyber-entities to move to another platform.
Pheromone emission	allows cyber-entities to emit their pheromones and sense pheromones emitted by other cyber-entities.
Lifecycle	provides cyber-entities lifecycle operations.
Resource sensing	allows cyber-entities to sense the type, amount and unit cost of available resources.
Energy management	keeps track of energy level of the cyber-entities running on the local platform.

reference. The service checks if the partner exists, and if it does, obtains the partner's attributes and instantiates a relationship object.

(2) Dynamic discovery. The autonomy and decentralization features of cyber-entities produce the need for a method to locate cyber-entities. Therefore, the bionet platform provides the social networking service, which allows cyber-entities to dynamically discover others with various search criteria in a decentralized manner (Table 1). The design of this service is similar to that of peer-to-peer systems [22, 23]. Cyber-entities construct an overlay network with their relationships for routing discovery queries among them. A discovery process consists of *query initialization*, *query matching*, *query forwarding* and *query hit backtracking*.

In *query initialization*, a discovery originator (i.e. a cyber-entity) begins a discovery process by generating a query with the social networking service. Each query contains its GUID to distinguish it from other queries, hops-to-live count to determine discovery termination, and search criteria. Search criteria are described based on the OMG constraint language [21]. Examples of search criteria are as follows:

```
GUID=='sti3sdr98rd56fn...'
serviceType=='HTTP/1.1' and serviceCost<150.0
```

The *query matching* is performed when a cyber-entity receives a query from another cyber-entity. The social networking service examines if the received query's search criteria match a given cyber-entity. If it does, a query hit is returned to the discovery originator. Otherwise, the query is forwarded to other cyber-entities.

In *query forwarding*, queries are routed from cyber-entity to cyber-entity through their relationships, seeking the cyber-entities that satisfy search criteria. Each cyber-entity uses the social networking service to forward a query. The service decrements the hops-to-live value in a received query, and if the value becomes zero, the query is discarded. Also, if the query forms a loop in its forwarding path, it is discarded. Otherwise, the query is forwarded to the relationship partners. The social networking service keeps a record of the query's GUID, the cyber-entity from which the query is received, and the cyber-entity to which the query is forwarded.

The *query hit backtracking* phase is performed when a query matches a cyber-entity. A query hit is generated and returned back to the discovery originator, following the reverse route of the forwarding path that led to the cyber-entity being returning the query hit.

In addition to the social networking service, the bionet platform provides another service, called the CE sensing service to locate cyber-entities (Table 1). This service keeps track of the cyber-entities that exist on a local platform. This service is typically used for cyber-entities to establish their initial relationships.

(3) Migration. Since cyber-entities move around the network, the bionet platform provides the migration service, which allows them to migrate from a platform to another. This service implements *weak migration* [25], in which data state associated with a cyber-entity is transferred between different bionet platforms.

The migration service is responsible for sending out a cyber-entity and receiving a migrating cyber-entity. It transfers a cyber-entity's class name, class definition and runtime data state to the migration service running on a destination platform. The class definition and data state are serialized at an origin platform and de-serialized on a destination by using Java serialization mechanism. The transferred class definition is loaded into a JVM on a destination platform using the bionet class loader. After the class definition is loaded and data state of a cyber-entity is de-serialized, a destination-side migration service instantiates the cyber-entity.

Since cyber-entities are autonomous, they move around the network without any intervention from others. As a result, after a cyber-entity moves, the relationships (references contained in the relationships) associated with the cyber-entity become invalid. In this case, by using the social networking service, cyber-entities may locate the missing cyber-entity or may locate other cyber-entities that implement the service the missing one provides.

The bionet platform provides another option for cyber-entities to locate missing cyber-entities through the pheromone emission service (Table 1). Due to space limitation, please see [26] for more detailed design.

(4) Lifecycle management. As cyber-entities are dynamically initialized, replicated or reproduced, the bionet platform provides the lifecycle service, which provides several lifecycle operations to them (Table 1). The service is used to initialize a cyber-entity when it is newly created or when it completes a migration. The service accepts a cyber-entity's instance, creates a CE context to associate it with the cyber-entity, assigns a GUID to the cyber-entity, and registers the cyber-entity to the bionet container.

The lifecycle service is also used to replicate a cyber-entity or reproduce a child cyber-entity from two parent cyber-entities. The service makes a deep copy of a parent cyber-entity using Java serialization mechanism. Mutation may happen on a child cyber-entity during replication and reproduction. For example, inherited set of relationships and other properties (e.g. behavior policies) may be randomly modified. Crossover happens during reproduction to inherit relationships and other properties from two parents. The evolutionary aspect of cyber-entities is beyond the scope of this paper. Please see [9, 10] for more details about this issue.

(5) Environment sensing. Since cyber-entities need to sense their surrounding network conditions to perform their behaviors, the bionet platform provides a series of mechanisms for environment sensing. They allow for each cyber-entity to sense (1) its current energy level, (2) resource availability on a local platform, (3) the current traffic load on a local platform, and (4) the number of cyber-entities running on a local platform.

The current energy level of a cyber-entity is available through the energy management service (Table 1). This service keeps track of the energy level of every cyber-entity running on a local platform. The resource sensing service allows cyber-entities to monitor the type, amount and unit cost of resources (CPU cycles and memory space) available on a local platform (Table 1). Due to space limitation, please see [26] for more details. Cyber-entities can also sense the current traffic load and the number of cyber-entities on a local platform. As described earlier, the traffic load is available through the bionet container, and the number of local cyber-entities is available through the CE sensing service (Table 1).

4. Measurement Results

This section describes some of the measurement results to evaluate the footprint, efficiency and scalability of the bionet platform. The measurements were conducted with two bionet platforms running on different Windows 2000 PCs, each of which hosts Java 2 SDK (version 1.4.2_01 from Sun Microsystems) with an Intel Pentium 4 processor (1.8 GHz) and 512 MB RAM. The PCs were connected through a 100Mbps Ethernet switch.

Table 2 shows the bootstrap overhead and memory footprint of each platform component. The bootstrap overhead measures the time for the bionet platform to initialize each component, and the bootstrap memory footprint measures the amount of memory space each component consumes when it is initialized. Table 2 shows that both of the measures are fairly small.

Figure 4 shows the throughput of the bionet platform per cyber-entity (i.e. how many interactions two cyber-entities can perform per sec.). In this measurement, we deployed a single cyber-entity (sender cyber-entity) on a platform and a range of cyber-entities (from 1 to 1000 receiver cyber-entities) on the other platform. The sender randomly chose one of the remote receivers and sent an empty message to the chosen receiver. Then, the receiver sends back an empty message to the sender.

Table 2. Bootstrap overhead and memory footprint of each platform component

platform component	overhead	footprint
Bionet message transport	22.98 msec	6.65 KB
Bionet container	127.06 msec	8.88 KB
Bionet class loader	9.11 msec	3.97 KB
Platform representative	82.31 msec	5.23 KB
Relationship mgt service	23.17 msec	4.48 KB
Social networking service	69.85 msec	12.03 KB
CE sensing service	56.43 msec	7.82 KB
Migration service	33.13 msec	4.88 KB
Pheromone emission service	37.79 msec	7.39 KB
Lifecycle service	91.92 msec	44.07 KB
Resource sensing service	64.36 msec	42.12 KB
Energy management service	59.02 msec	8.12 KB
Total	677.13 msec	154.64 KB

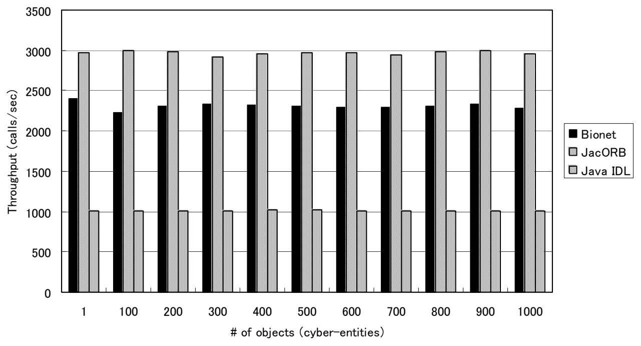

Figure 4. Throughput of message exchanges

As Figure 4 shows, two cyber-entities running on different platforms can send approximately 2,200 messages (i.e. 1,100 roundtrip interactions) per second with each other. This result is competitive with well-known Java-based distributed object platforms (JacORB[2] and Java IDL[3]), and we believe the bionet message transport and bionet container are efficient enough. Figure 4 also shows that the throughput remains mostly constant as the number of cyber-entities grows up to 1,000, indicating that the bionet platform scales.

In the next measurement, we deployed a bionet platform on a PC and multiple cyber-entities on the platform. Each cyber-entity implements a web server function that processes the HTTP GET request message. An emulated user was deployed on the same PC, and it sent GET requests to the cyber-entities. Upon receiving a request, each cyber-entity locates, reads and returns a requested file. It keeps five different files whose sizes are 500B, 5KB, 50KB, 500KB and 5MB. These five sizes are representative in Webstone [27], a well-known web server profiling tool. The request rate was 10 requests per second.

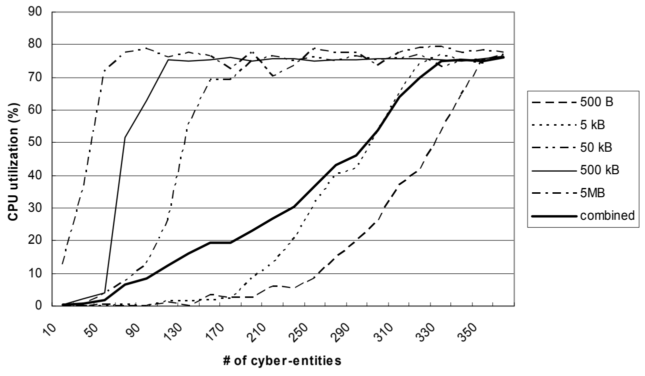

Figure 5. CPU utilization of the cyber-entities that implement web server functions

Table 3. Probability to request different sized files

File size (bytes)	Probability (%)
500	35
5 K	50
50 K	14
500 K	0.9
5 M	0.1

Figure 5 shows the CPU utilization of the web server cyber-entities and bionet platform. When the CPU utilization goes around 75%, the total utilization on the testbed PC reaches 100%; the other 25% is consumed by the operating system. In the case of 500B file requests, 350 cyber-entities can be executed under 75% CPU utilization. In 5M file requests, 50 cyber-entities can be executed. A heavy line in Figure 5 shows the CPU utilization in the case that a user requests different-sized files based on the probability shown in Table 3, which is defined by WebStone. In this measurement, 320 cyber-entities can work simultaneously under 75% CPU utilization. Also, the CPU utilization increases almost linearly as the number of cyber-entities grows. Given these results, we confirmed the bionet platform is scalable enough in terms of the number of cyber-entities.

5. Related Work

The bionet platform is similar to existing mobile agent platforms, such as Aglets[4] and AgentSpace [28], in the sense that it implements a weak migration mechanism for agents. However, unlike them, the bionet platform emphasizes on decentralized organization of agents. Almost all the existing agent platforms assume the existence of centralized entities (e.g. directories). Hive

[2] www.jacorb.org
[3] java.sun.com/products/jdk/idl/

[4] http://sourceforge.net/projects/aglets/

addresses decentralization of agents [29], but its implementation depends on a centralized directory (Java RMI registry). In contrast, the bionet platform allows agents (i.e. cyber-entities) to form a decentralized overlay network among agents using their relationships and perform distributed discoveries through the relationships with the social networking service.

Pole is similar to our social networking service in the sense that it implements a decentralized agent discovery mechanism [31]. Its discovery process is performed on a *structured* overlay[5] with a distributed hash function. In the discovery mechanisms based on distributed hash functions (e.g. Chord [23]), it is expensive to maintain their overlay structures in dynamic environment where peers (or agents) often join and leave the network [32]. Also, they do not allow each peer to specify multiple search criteria for each query. Unlike them, our social networking service is designed on a *loosely-structured* overlay[5] among cyber-entities in order to assume dynamic networks. It also provides a flexible discovery scheme that allows cyber-entities to specify multiple search criteria (as name-value pairs) for each query.

5. Concluding Remarks

This paper described our research effort to investigate a platform for autonomic agents running on pervasive networks. We presented the designs of our platform and showed that the platform is efficient, scalable and lightweight through measurement results.

As future work, we plan an extended set of measurements. We evaluated scalability and efficiency of our platform mechanisms in terms of the number of cyber-entities running on platforms, but the network size is still small. We will deploy the bionet platforms and cyber-entities on larger-scale networks to identify the impact of network size on the platform performance.

References

[1] M. Weiser, "The Computer for the 21st Century," Scientific American September, 1991.

[2] D. Norman, *The Invisible Computer*, MIT Press, 1998.

[3] M Satyanarayanan, "Pervasive Computing: Vision and Challenges," IEEE Personal Communications, August, 2001.

[4] K. Henricksen, J. Indulska and A. Rakotonirainy, "Infrastructure for Pervasive Computing: Challenges," *Proc. of Workshop on Pervasive Computing INFORMATIK 01*, 2001.

[5] S. Acharya, "Application and Infrastructure Challenges in Pervasive Computing," *Proc. of NSF Workshop on Context-Aware Mobile Database Management*, January 2002.

[6] G. Banavar and A. Bernstein, "Software Infrastructure and Design Challenges for Ubiquitous Computing Applications," CACM, vol. 45, no. 12, December 2002.

[7] P. Maes, "Modeling Autonomous Adaptive Agents," *Artificial Life, I (1&2)9, 1994.*

[8] S. Franklin and A. Graesser, "Is it an agent or just a program?: A Taxonomy for Autonomous Agents," *Proc. of ATAL'96*, 1996.

[9] A. G. Ganek and T. A. Corbi, "The dawning of the Autonomic Computing Era," *IBM System Journal*, vol. 42, no. 1, 2003.

[10] T. Suda, T. Itao and M. Matsuo, "The Bio-Networking Architecture: The Biologically Inspired Approach to the Design of Scalable, Adaptive, and Survivable/Available Network Applications," In K. Park (ed.) *The Internet as a Large-Scale Complex System*, Princeton University Press, 2002

[11] M. Wang and T. Suda, "The Bio-Networking Architecture: A Biologically Inspired Approach to the Design of Scalable, Adaptive, and Survivable/Available Network Applications," *Proc. of the 1st IEEE SAINT*, 2001.

[12] J. Suzuki, T. Nakano, K. Fujii, N. Ikeda and T. Suda, "Dynamic Reconfiguration of Network Applications and Middleware Systems in the Bio-Networking Architecture," *Proc. of IEEE LARTES*, 2002.

[13] J. Suzuki and T. Suda, "Adaptive Behavior Selection of Autonomous Objects in the Bio-Networking Architecture," *Proc. of AINS*, 2002.

[14] T. Hong, "Performance," *Peer-to-Peer*, A. Oram (ed.), Chapter 14, Wiley, 2001.

[15] N. Minar, K. H. Kramer and P. Maes, "Cooperating Mobile Agents for Dynamic Network Routing," *Software Agents for Future Communications Systems*, 1999.

[16] G Cabri, L. Leonardi and F Zambonelli, "Mobile-Agent Coordination Models for Internet Applications," Computer 33(2):82-89, February 2000.

[17] C. Castelfranchi, "Guarantees for Autonomy in Cognitive Agent Architecture," *Proc. of ECAI-94 Workshop on Agents Theories, Architectures, and Languages*, Springer, 1995.

[18] M. Luck and M. P. D'Inverno, "A Formal Framework for Agency and Autonomy," *Proc. of MAS"95*, 1995.

[19] T. Itao, T. Nakamura, M. Matsuo, T. Suda and T. Aoyama, "The Model and Design of Cooperative Interaction for Service Composition," *Proc. of the DICOMO*, 2001.

[20] OMG, *The CORBA Specification, version 3.0*, 2002.

[21] OMG, *The Trading Object Service*, 2000.

[22] I. Clarke et al., "Freenet: A Distributed Anonymous Information Storage and Retrieval System in Designing Privacy Enhancing Technologies," *Proc. International Workshop on Design Issues in Anonymity and Unobservability*, LNCS 2009, Springer, 2001.

[23] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," *Proc. of ACM SIGCOMM 2001*, 2001.

[24] OMG, *The Trading Object Service*, 2000.

[25] A. Fuggetta, G. P. Picco, and G. Vigna, "Understanding Code Mobility,". *IEEE Trans. on Software Engineering, 24*(5), May 1998.

[26] J. Suzuki and T. Suda, "Design and Implementation of an Scalable Infrastructure for Autonomous Adaptive Agents," Proc. of the 15th IASTED International Conference on Parallel and Distributed Computing and Systems, November 2003.

[27] G. Trent and M Sake, "WebStone: The First Generation in HTTP Server Benchmarking," Mindcraft, Inc., 1995.

[28] N.J.E. Wijngaards, B.J. Overeinder, M. van Steen, and F.M.T. Brazier, "Supporting Internet-Scale Multi-Agent Systems," *Data Knowledge Engineering (4)2-3,* 2002.

[29] N. Minar, M. Gray, O. Roup, R. Krikorian and P. Maes, "Hive: Distributed Agents for Networking Things," *Proc. of ASA99*, 1999.

[30] S. Sameshima, J. Suzuki, S. Steglich and T. Suda, "Platform Independent Model (PIM) and Platform Specific Model (PSM) for Super Distributed Objects," OMG final adopted specification, September 2003.

[31] B. Overeinder et al, "Integrating Peer-to-Peer Networking and Computing in AgentSpace Framework," *Proc. of IEEE International Conference on Peer-to-Peer Computing*, 2002.

[32] S. Androutsellis-Theotokis, "A Survey of Peer-to-Peer File Sharing Technologies," Athens University of Economics and Business, 2002.

[5] See [32] for the details in structured and loosely-structured overlays.