

Service Emergence based on Cooperative Interaction of Self-Organizing Entities

Tomoko Itao[†] Tetsuya Nakamura[†] Masato Matsuo[†]

Tatsuya Suda^{1,†,††} Tomonori Aoyama^{†††}

[†]NTT Network Innovation Laboratories,

^{††}University of California, Irvine,

^{†††} University of Tokyo

E-mail: {tomoko, tetsuya, matsuo}@ma.onlab.ntt.co.jp, suda@ics.uci.edu,
aoyama@mlab.t.u-tokyo.ac.jp

Keywords: autonomous entities, distributed, self-organizing, service emergence, evolution

Technical area: Agents on the Internet

Contact person's addresses/numbers:

Name: Tomoko Itao

Address: NTT Network Innovation Laboratories, 3-9-11, Midori-cho, Musashino-shi, Tokyo 180-8585, Japan

Tel: 0422-59-3588

Fax: 0422-59-4810

E-mail: tomoko@ma.onlab.ntt.co.jp

In this paper, we describe Jack-in-the-Net (Ja-Net) architecture for service emergence and evolution in a large scale, open network environment. In Ja-Net, a service is implemented by a collection of cyber-entities and provided through cooperative interaction of cyber-entities. Cyber-entities interact with each other using Ja-Net ACL (Agent Communication Language). To ensure that service composition is efficient, cyber-entities establish relationships with interaction partner cyber-entities and store information about a partner cyber-entity in a

relationship record. Then, a cyber-entity selects interaction partner cyber-entities by examining relationship records. The utility of a partner cyber-entity is evaluated and also stored in a relationship record. In Ja-Net, cyber-entities self-organize by gradually narrowing the cyber-entities to interact with based on the utility of partner cyber-entities. Consequently, an emergent service by a group of cyber-entities results. Presently, we are implementing a prototype of Ja-Net to verify the feasibility of cooperative interaction and self-organization features of Ja-Net.

1 Introduction

Recent advancement of computer networks and network aware ubiquitous devices [1] put forth a vision of a universal network which openly interconnects every human being and most human-made objects. For instance, new devices such as light weight sensors, wearable computers, mobile phones, vehicles, and home appliances, as well as conventional computers, may become components of a universal network. Software and information content are also components of a universal network.

With the increasing number of components becoming a part of a universal network that dynamically changes, a universal network is required to be self-organizing with inherent support for mobility, scalability, and adaptation to short and long term changes in user and network conditions.

Jack-in-the-Net (Ja-Net) [2][3][5] is a radically new paradigm of a network with service emergence and evolution capability. In Ja-Net, major components of a universal network are represented by cyber-entities. Cyber-entities are autonomous and self-organizing. They have functionality related to their service and follow biologically inspired, simple behavior rules (e.g., migration, relationship establishment, replication, reproduc-

¹A part of Prof. Suda's research conducted at UCI and presented in this paper was supported by the National Science Foundation 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 University of California MICRO Program, and Nippon Telegraph and Telephone Corporation (NTT).

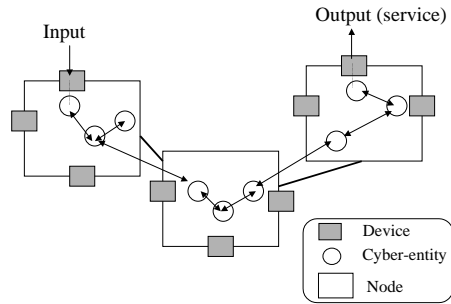


Fig. 1: Ja-Net architecture

tion, mutation, death, etc.). In Ja-Net, a service is implemented collectively by a group of cyber-entities through autonomous interactions of cyber-entities. Figure 1 shows an architecture of Ja-Net. Major architectural components include nodes, devices, and cyber-entities. A node is a runtime environment of cyber-entities, which may run on computers and other devices such as a PDA and a cellular phone. Some cyber-entities may represent a device (e.g., sensor, stereo, monitor) and provide functionality to control such a device, while other cyber-entities may represent a logical component (e.g., hotel reservation, coupon, MP3-encoded music file).

Autonomous interaction In Ja-Net, an event in the physical world triggers cyber-entities to invoke their actions associated with a service and/or behaviors. Examples of such an event includes reception of user request and reception of sensor data. A cyber-entity's action also triggers actions of other nearby cyber-entities. For instance, a cyber-entity may invoke its action triggered by a receipt of a message from other cyber-entity, a notification of changes in the status of the cyber-entity and the notification of changes in its environment. In Ja-Net, a cyber-entity may simply broadcast its service result, which may be picked up by cyber-entities in the environment, while another cyber-entity may actively find interaction partners by monitoring its local environment. In Ja-Net, a service is implemented collectively by a group of cyber-entities through such autonomous interactions of cyber-entities.

Service emergence Cyber-entities establish relationships with interaction partner cyber-entities and store information about a partner cyber-entity, such as cyber-entity ID, service description and age, in a relationship

record. The strength of a relationship is calculated based on the usefulness of a partner cyber-entity and stored in a relationship record too. Relationship records are updated through interactions. As relationships grow, cyber-entities self-organize by gradually narrowing the cyber-entities to interact with based on the strength of relationship. Consequently, an emergent service by a group of cyber-entities results. For instance, cyber-entities representing a personal identification tag, a monitor, a screen-saver, and a TV news clip may interact and provide a screen saver service and display a news clip on a monitor near a user. Being used by mass users, the news clip service may self-organize based on strong relationship and repeatedly provide the same service. The screen saver service may be customized based on user preference, resulting in an emergence of a personalized service.

Besides above features, Ja-Net has following characteristics.

Locality:

In a large scale system that Ja-Net assumes, we may assume that there exist many cyber-entities that are similar to a given cyber-entity in attributes and service capability. Moreover, in Ja-Net, multiple copies of a given cyber-entity may exist since logical cyber-entities may replicate under certain conditions (such as when it accumulates a large amount of energy). Since we assume abundance of copies of cyber-entities and cyber-entities with similar services in Ja-Net, Ja-Net relies on interactions of cyber-entities that are located physically close together.

Energy:

Energy is a parameter to autonomously control the system of Ja-Net. Cyber-entities may gain energy in exchange for performing a service, and they may consume energy to invoke behaviors. If the energy consumption of a cyber-entity is not balanced by the energy income it receives by providing services, it dies from lack of energy. Therefore, cyber-entities attempt to optimize their behaviors so that their energy gain is maximized.

Service evolution through emergence and natural selection:

In Ja-Net, services adapt to heterogeneous and dynamic conditions of a universal network through evolution. Evolution is achieved through processes of emergence and natural selection. In Ja-Net, diverse services may emerge through dynamically creating and modifying relationships of cyber-entities (i.e., through dynamically creating and removing relationships with partner cyber-entities to provide a service). Only successful (popular) services remain in Ja-Net, and non-successful services are eliminated from Ja-Net through a built in natural selection mechanism of Ja-Net. Note that a service disappears when any one of relationships between cyber-entities that collectively provide a service is eliminated. Evolution of services promotes the adaptability and survivability of the system.

In this paper, we focus on service emergence features in Ja-Net and describe how services autonomously emerge from interactions of cyber-entities. The rest of the paper is organized in the following manner. In section 2, an overview of existing approaches to cooperative interaction of distributed entities are introduced. In section 3, the model of cooperative interaction for service composition in Ja-Net is addressed. Section 4 describes self-organization methods employed in Ja-Net. Section 5 describes the system design of Ja-Net. Future work is mentioned in section 6.

2 Approaches for cooperative interaction

Various frameworks and architectures exist for achieving cooperation among distributed and heterogeneous entities on a network to provide services. Hive [9] is a framework to create distributed applications. In Hive, a service is provided through multiple distributed agents where agents interact by specifying the Java interface object of the partner. Thus, interaction in Hive is limited to among those agents that mutually implement the interface object of the partner. Unlike Hive, Ja-Net supports message-based communication for cyber-entity interactions. In Ja-Net, a cyber-entity may communicate with other cyber-entities without implementing a special interface for a particular cyber-entity to maximize the flexibility of cyber-entity interactions.

The Data Field (DF) architecture of ADS (Autonomous Decentralized Systems) [12] proposes information

sharing facility for indirect cooperation of anonymous entities. In ADS, all entities join the Data Field (DF) where data is broadcast into the DF. Entities that join the DF receive the data and invoke a local process if possible. That is, in DF architecture, interactions of entities are indirect and message-driven. DF architecture is similar to blackboard architecture [13] in that entities cooperate indirectly. Ja-Net also supports message-driven interaction model like ADS and blackboard architecture. In addition to that, Ja-Net is distinguished from these architectures due to its self-organizing capability where cyber-entities autonomously narrow down the partner cyber-entities to interact with, which results in an emergent service.

KQML [7][8] is a language protocol for exchanging information and knowledge. FIPA ACL [14] is derived from KQML. KQML and FIPA ACL define Speech Act[16][17]-based message called *performative*. An expected sequence of performatives is defined by an Interaction Protocol (IP). Agents cooperate through exchanging performatives according to an IP. In Ja-Net, cyber-entities interact using performatives like KQML and FIPA ACL to achieve flexible interactions between cyber-entities. Basically, Ja-Net ACL performatives are subset of FIPA ACL, but Ja-Net architecture is not compliant with the FIPA reference model. For instance, in FIPA reference model, Directory Facilitator which is a specific agent to facilitate communication between agents is introduced. On the contrary, in Ja-Net, cyber-entities facilitate conversation in peer-to-peer manner to maximize diversity in cyber-entity interactions. For instance, the broadcast area of a given cyber-entity may be limited to by the amount of energy stored in the cyber-entity. Thus, cyber-entity may behave to obtain enough amount of energy to expand its broadcast area or migrate to a node with large cyber-entity population.

Cool [10] proposes a language to define IPs and a facility to coordinate interactions of multiple agents. Although Cool achieves flexible cooperation between agents, IPs in Cool often are complex because it is necessary to define every possible sequence of interaction, from regular sequence to irregular sequence, in each IP to guarantee that IPs collectively process an expected task. Unlike Cool, IPs in Ja-Net are simple due to reduced dependency between IPs, which is achieved by a relatively coarse granularity of cyber-entity services.

Bee-gent [11] proposes a centralized IP model for cooperation of multiple agents to reduce complexity in IPs.

In Bee-gent, a mediator agent maintains a centralized IP, and coordinates interaction of wrapper agents (i.e., wrapper of a local process). With this centralized IP approach, Bee-gent restricts the flexibility and scalability of wrapper agent cooperation. Unlike Bee-gent, no centralized IP is used in Ja-Net so that interaction sequences are non-deterministic and diverse so that emergent services result. In Ja-Net, each cyber-entity implements one or more IPs but such IPs are simple unlike Cool by limiting the granularity of cyber-entities. As a result, in Ja-Net, the strict coordination facility as in the Bee-gent is not necessary.

3 The model of cooperative interaction

3.1 Cyber-entity

A cyber-entity consists of *attributes*, *body* and *behavior* [3]. Attributes describe characteristics of a cyber-entity such as its cyber-entity ID (CEID), service description, cyber-entity type, stored energy level and age. Body implements a service that a cyber-entity provides. For instance, a cyber-entity may implement a control software for a device in its body, while another cyber-entity may implement hotel reservation service in its body. In Ja-Net, in order to make service-related cyber-entity interactions simple, cyber-entity services (bodies) implement a unit of service at a relatively coarse level so that they avoid unnecessary dependency between them and maintain the communication overhead between cyber-entities at a manageable level. Cyber-entity behavior implements non-service related actions that are inherent to all cyber-entities. Examples of behavior includes migration, relationship establishment, replication, etc. In Ja-Net, each cyber-entity service and cyber-entity behavior is defined by an Interaction Protocol (IP) (see section 3.3). Cyber-entities may exchange two kinds of messages during the execution of a cyber-entity service and a cyber-entity behavior.

In Ja-Net, a cyber-entity exchanges two types of messages, a message for communication with another cyber-entity and a message to notify an event such as changes in the status of a cyber-entity itself or changes in the status of another cyber-entity. Upon receiving a message, a cyber-entity interprets the message and invokes an appropriate action (that is either a cyber-entity service or a cyber-entity behavior). The result of

Table 1: Parameters of a Ja-Net ACL performative

| Name | Meaning |
|---------------------------------|---|
| <code>:performative-name</code> | Speech acts |
| <code>:receiver</code> | CEID of the receiver |
| <code>:sender</code> | Self CEID |
| <code>:reply-with</code> | The expected label in a response to the current message |
| <code>:in-reply-to</code> | The expected label in a response to a previous message (same as the <code>:reply-with</code> value of the previous message) |
| <code>:language</code> | Content description language. XML is used in Ja-Net. |
| <code>:ontology</code> | Content ontology |
| <code>:content</code> | XML data or performative |

an action may be broadcast/multicast or unicast, which in turn is received by other cyber-entities and triggers their actions. Thus, cyber-entities indirectly cooperate by exchanging messages with other cyber-entities to compose a service. Different services may be composed of depending on which cyber-entities are present in a given environment.

3.2 Ja-Net ACL

Cyber-entities communicate using Speech Act-based Ja-Net ACL (Agent Communication Language). Table 1 shows parameters of a Ja-Net ACL performative. One of the parameters, language denotes the language in which the content parameter is described. In Ja-Net ACL, XML [15] is used as a default language. For each performative, an ontology may be specified in the parameter, ontology. Ontology for a service domain may be defined by either developers of cyber-entity services or Ja-Net.

Table 2: Ja-Net ACL performatives

| Category | Name | Meaning |
|--------------|-----------------------|--|
| Dialog | request | requests a receiver cyber-entity to perform an action (body or behavior) |
| | inform | informs what a message sender cyber-entity knows |
| | agree | agrees to perform a requested action |
| | refuse | refuses to perform a requested action, and explains the reason for the refusal |
| | failure | notifies that an action was attempted but failed |
| | query-ref | asks another cyber-entity for information that matches a referential expression (referential expression is stored in a :content) |
| | query-if | asks another cyber-entity whether or not a given proposition is true (proposition is stored in a :content). |
| | not-understood | notifies that a receiver failed to understand a received message. |
| Facilitation | advertise | notifies attributes of a cyber-entity itself to nearby cyber-entities. |
| | recruit | notifies that a sender is looking for a cyber-entity with specific attributes. |

Performatives available in Ja-Net ACL are listed in Table 2. They are classified into two categories: dialog and facilitation. Dialog type performatives are used during communication between cyber-entities while facilitation type performatives are used to establish communication between cyber-entities. Dialog type of Ja-Net ACL performatives is a subset of FIPA ACL [14] performatives. Typical sequence of dialog type of performatives are described as follows. A cyber-entity may send a **request** performative to other cyber-entities that it has a relationship with, to invoke an action associated with the service of a receiver cyber-entity. The receiver cyber-entity responds with either an **agree** or a **refuse** to the sender cyber-entity. In case where the receiver cyber-entity responds with **agree**, then, the receiver cyber-entity sends an **inform** to notify the result of the action when the action is successfully completed or **failure** to inform that the action failed.

Performatives are used not only for services but for behaviors. In Ja-Net, facilitation type performatives are used to establish initial relationships between cyber-entities (See section 3.4). **advertise** is used to notify nearby cyber-entities of itself. **recruit** is used to get appropriate cyber-entities to respond to the performative. Cyber-entities broadcast an **advertise** or a **recruit** in their environment to find appropriate interaction partners. Facilitation type of Ja-Net ACL performatives are not in FIPA ACL performatives and specific to Ja-Net ACL.

Upon receiving a Ja-Net ACL performative, a cyber-entity parses the performative and interprets its content. The semantics of a content data in a performative is specific to each cyber-entity because different services have different semantics. For instance, the attribute of “date” may be interpreted as a birthday in a horoscope service, while it may be interpreted as a part of schedule in a PIM (Personal Information Management) service.

3.3 Interaction protocol (IP)

In Ja-Net, cyber-entity services are defined by an Interaction Protocol (IP) called **Service IP** (SIP) as described in section 3.1. Similarly, in Ja-Net, cyber-entity behaviors are also defined by an Interaction Protocol (IP) called **Behavior IP** (BIP). Different SIPs implement different cyber-entity services while BIPs are built-in

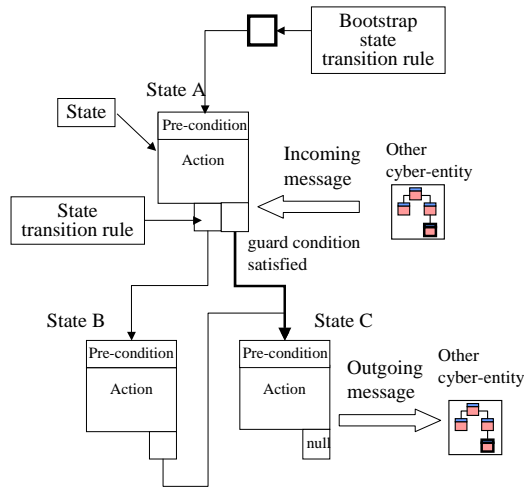


Fig. 2: Ja-Net interaction protocol

and common to all cyber-entities.

IP model In Ja-Net, an IP (either SIP or BIP) is modeled as a finite state machine that consists of *states* and *state transition rules* (See Figure 2).

A state consists of a *pre-condition* and an *action*. An action is a local process of a cyber-entity with input/output. The state of an IP transit to a next state only if the pre-condition of the next state holds. A pre-condition is a condition on an incoming message (either performative or event notification). For instance, a pre-condition on an incoming performative in a SIP may be described as follows.

```

if (performative-name == 'request' &&
    content.action-name == 'discount'){
    return true;
} else {
    return false;
}

```

In the above example, a SIP examines the parameter values of an incoming performative, such as `:performative-name` and `:action-name` in a `:content`.

A state transition rule for a given state specifies a next state to transit to and an event to trigger state transition called a *guard condition*. As described in section 3.1, examples of such an event include receiving a performative from another cyber-entity and receiving a notification of status change of either self or another cyber-entity. Different state transition rules may be defined for different processing results of an action in the current state.

Upon receiving a message, a cyber-entity firstly examines a guard condition of a state transition rule for a current state. When the guard condition holds, a pre-condition for a next state is examined. Then, if the pre-condition holds, the current state moves to the next state. Upon state transition, an action for the current state is invoked.

In an IP, roles of interaction partner cyber-entities (called CE roles) may be explicitly defined and used to describe cyber-entity services and cyber-entity behaviors. By CE role, a set of cyber-entities satisfying specific attributes or having a specific behavior is meant.

IP bootstrap Each IP has a bootstrap state transition rule. The bootstrap state transition rule is a special state transition rule that is not associated with any state in a given IP. In the bootstrap state transition rule, the next state is set to an initial state of a given IP. The bootstrap state transition rule is invoked when a cyber-entity receives an event (i.e., receipt of a performative or notification of status change). Upon receiving an event, a cyber-entity examines the bootstrap state transition rule of an IP. If the guard condition of a bootstrap state transition rule holds, then the pre-condition of an initial state is examined. If the pre-condition holds, an action for the initial state is invoked.

3.4 Relationship

In Ja-Net, a service is implemented collectively by a group of cyber-entities through such autonomous interactions of cyber-entities as mentioned earlier in section 3.3. To ensure efficient service composition in Ja-Net, a cyber-entity establishes relationships with other cyber-entities through interaction. A relationship is

implemented by a relationship record where each cyber entity maintains a table of relationship records called a relationship table. A relationship may be viewed as (a cyber-entity's) information cache of other cyber-entities and is used by a cyber-entity to select interaction partners (receivers) to send a performative.

Table 3 shows the attributes of a relationship (key-value pairs) stored in a relationship record. *CEID* is a globally unique identifier of a cyber-entity. *CE-role-name* denotes a CE role value associated with a cyber-entity as mentioned earlier in section ip-section. *IP-name* denotes an IP that may be used to interact with the partner. *CE-attributes* is a list of attributes of a cyber-entity. *Energy-gain* denotes total amount of energy that is forwarded from a partner cyber-entity. *Access-count* denotes number of interactions with a partner cyber-entity. *Time-of-creation* denotes the time that this relationship record is created. *Direction* denotes whether a relationship is uni-directional or bi-directional. Two cyber-entities that have bi-directional relationship to each other are aware that they have bi-directional relationship to each other and update each other of its new location when they migrate. *Strength* evaluates utility of a partner cyber-entity and is used to help cyber-entities to self-organize (See section 4 for a strength evaluation method and how relationship strength helps self-organization). *Service-specific-properties* is used to store information that is specific to a cyber-entity service. Cyber-entity may freely store in its *Service-specific-properties* semantic information regarding the service it provides. Example information stored in *Service-specific-properties* may include response time, information transferred from a partner, query issued by a partner, etc.

Relationship establishment As mentioned earlier in section 3.2, cyber-entities create initial relationships by exchanging facilitation type performatives. For instance, a cyber-entity that has just migrated to a given node may broadcast an `advertise` performative to establish relationship with nearby cyber-entities. Upon receiving `advertise` performative from a cyber-entity that it has encountered for the first time, a cyber-entity creates a new relationship record and stores CEID of sender in a relationship record.

In order to find a suitable partner to establish a relationship with, a cyber-entity may broadcast a `recruit` performative. A CE that receives a `recruit` performative creates an `inform` performative, which contains

Table 3: Attributes of a relationship

| Attribute | |
|-----------------------------|---|
| CEID | M |
| CE-role-name | O |
| IP-name | O |
| CE-attributes | O |
| Energy-gain | O |
| Access-count | O |
| Time-of-creation | M |
| Direction | M |
| Strength | M |
| Service-specific-properties | O |

M = Must O = Optional

cyber-entity attributes of itself, sends the `inform` performative to the sender cyber-entity of the `recruit` performative. Upon receiving the `inform` performative, the sender cyber-entity of the `recruit` performative establishes a relationship with the sender cyber-entity of the `inform` performative and stores necessary information about a partner cyber-entity in the relationship record.

Relationship update Through interaction, additional information on the partner CE may be obtained, and the relationship may be updated. For instance, a cyber-entity may store service specific information in a relationship as *Service-specific-properties*. Some relationship attributes such as *Energy-gain*, *Access-count* and *Time-of-creation* may be automatically recorded in a relationship. Note that *Strength* is updated when a service is completed based on feedback from an end user who received the service (See section 4).

Partner selection When a cyber-entity sends a performative to other cyber-entities, it may either broadcast/multicast the performative or unicast the performative. When sending a performative, a cyber-entity may

select receiver cyber-entities by examining a relationship record to each cyber-entity that it has relationships to. A cyber-entity may specify one or more relationship attributes as keys and select relationships that match the keys. The CEIDs of receiver cyber-entities are obtained from the selected relationship. Various keys may be used to select receiver cyber-entities such as *CE-attributes*, *Service-specific-properties*, *Access-count*, and *Strength*. For instance, suppose that a cyber-entity stores “response-time” as a property of an *Service-specific-properties* in a relationship record. And it may select a receiver cyber-entity that has a relationship with the smallest response time.

4 Self-organization methods

4.1 The strength of a relationship

Although a cyber-entity may initially interact with all other cyber-entities in its environment, it gradually narrows the cyber-entities to interact with based on the strength of a relationship.

As mentioned earlier in section 3.4, the strength of a relationship indicates the utility of a partner cyber-entity in providing an end service to a (human) user. Because Ja-Net attempts to facilitate emergence of popular end services, the strength of each relationship among a group of cyber-entities that provide an end-service is adjusted based on how satisfied end-users are with the service provided. (In Ja-net, *happiness* is used to measure the degree of end user satisfaction with the service.) In Ja-Net, the strength of a relationship is either strengthened with positive happiness or weakened with negative happiness. The strength of a relationship may vary from negative real to positive real. With a pre-determined threshold value Θ_{TH} , the strength s of a given relationship is considered to be strong when

$$f_O(s) \geq \Theta_{TH}$$

holds where f_O is a mapping function of s within an appropriate range to compare with a given Θ_{TH} . Examples

of f_O include a sigmoid function defined as below.

$$f_{sig}(x) = \frac{1}{1 + e^{-x}}$$

In Ja-Net, strength value is examined by self-organization methods as described in section 4.2.

4.2 Service emergence based on strong relationships

As mentioned earlier in section 1, cyber-entities gradually narrows interaction partners as described below as relationships grow. There are two phases to narrow interaction partners.

Narrowing at IP invokation When invoking an IP in response to an incoming message, a cyber-entity firstly examines the strength of relationship to a sender cyber-entity. If the strength is considered to be strong (i.e., $f_O(s) \geq \Theta_{TH}$ holds for a given s and Θ_{TH}), the cyber-entity examines the state transition rule and corresponding pre-condition for a current state in the IP. Otherwise, the incoming message is discarded.

Narrowing at partner selection When selecting receiver cyber-entities from relationships for an outgoing message, a cyber-entity examines the strength of a relationship of each partner cyber-entity and selects only those that have strong relationships (i.e., $f_O(s) \geq \Theta_{TH}$ holds for a given s and Θ_{TH}). Therefore, cyber-entities with less strong relationships are not selected as interaction partners any more.

By narrowing interaction partner cyber-entities based on the strength of relationships, cyber-entities self-organize to provide popular end-services, which results in an emergent service.

4.3 A strength evaluation method

As described in section 4.1, in Ja-net, *happiness* is used to measure the degree of end user satisfaction with a service. Whenever cyber-entities provide an end service, the strength of relationships are adjusted based on the happiness of end user that received a service (see Figure 3). When an end-user is satisfied with a provided end-service, the value of happiness is set to 1 (positive happiness) otherwise the value of happiness

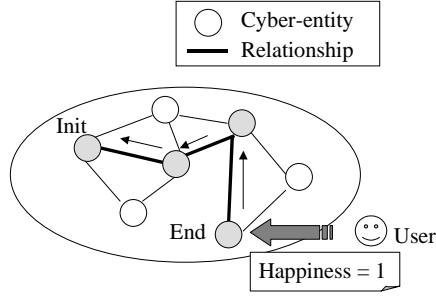


Fig. 3: Strength evaluation based on happiness of end user

is set to 0 (negative happiness). Whenever an end user receives a service, the user feedbacks happiness. Then, happiness is back forwarded from a cyber-entity at the end point of the service to a cyber-entity at an initial point of the service. Upon receiving a happiness value, each cyber-entity adjusts the strength of relationships to cyber-entities that it actually interacted with in providing the end service.

The strength of relationship s is modified by Δs such as

$$s = s + \Delta s$$

where s is initially set 0. There are various methods to calculate Δs . An example equation to calculate Δs is described below.

Example Δs equation In this method, Δs is calculated by applying the Steepest Decent Method to a squared error of an output value of $f_O(s)$ against a happiness value h_p returned by an end user. Here, h_p is either 0 or 1. The value of strength s is mapped to a value between 0 and 1 by using f_{sig} as f_O . Then, the Δs equation is shown below.

$$\Delta s = -\epsilon \frac{\partial (f_{sig}(s) - h_p)^2}{\partial s}$$

where $s_0 = 0$, ϵ is a very small positive real. Since $f'_{sig}(x) = f_{sig}(x)(1 - f_{sig}(x))$ holds, the above equation is represented as follows.

$$\Delta s \equiv -\epsilon (f_{sig}(s) - h_p) f_{sig}(s) (1 - f_{sig}(s))$$

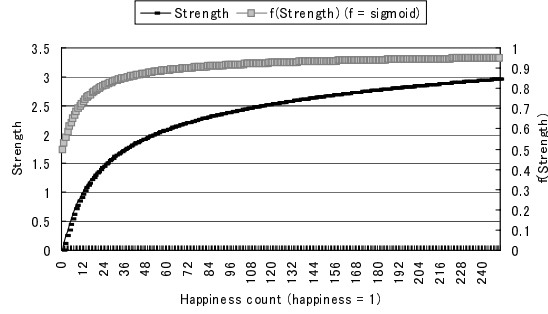


Fig. 4: Results of strength adjustment (1)

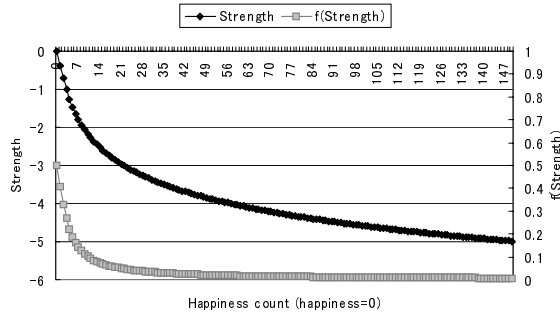


Fig. 5: Results of strength adjustment (2)

Figure 4 shows results of strength adjustment during simulation run of 250 happiness feedbacks where the happiness value h_p is always 1. Figure 5 shows results of strength adjustment during simulation run of 150 happiness feedbacks where the happiness value h_p is always 0. Figure 6 shows results of strength adjustment during simulation run of 100 happiness feedbacks where the happiness value h_p is alternation of seven consequent 1s (positive) and two consequent 0s (negative). For all simulations, ϵ is set to 1.0.

With positive happiness ($h_p = 1$), the strength value s increases quickly when the value of s is near 0. With negative happiness ($h_p = 0$), the strength value s decreases quickly when the value of s is near 0. With 70% probability of receiving positive happiness, the strength and the output of $f_{sig}(s)$ gradually increase and the output of $f_{sig}(s)$ increases toward 0.7.

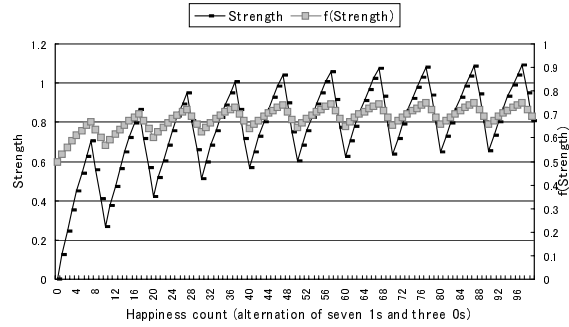


Fig. 6: Results of strength adjustment (3)

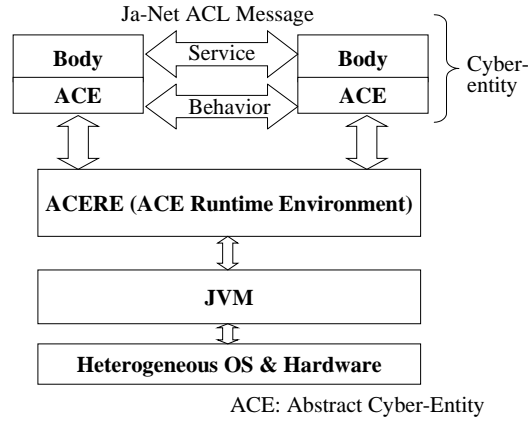


Fig. 7: System architecture

5 The system design

5.1 The system architecture

Figure 7 shows an overview of the system architecture of Ja-Net. Body is implemented as a SIP in each cyber-entity. Abstract Cyber-Entity (ACE) is a base class of a cyber-entity where all cyber-entities are derived from the ACE class. Cyber-entity attributes and BIPs are implemented in the ACE class. A message layer concept is introduced for message exchange between two cyber-entities where service messages and behavior messages are exchanged at service sublayer and behavior sublayer, respectively. ACE Runtime Environment (ACERE) provides a communication facility and other core functionalities for cyber-entity interaction, such as cyber-entity directory and energy management.

5.2 The internal collaboration of ACE

Figure 8 shows the internal structure of ACE. A state model is defined for each SIP and BIP where body is implemented by one or more SIPs while behaviors are implemented by BIPs. Relationship table manages a list of relationship records. An IP dispatcher receives an incoming message from ACE communication service in ACERE and determines which IP (state model) to invoke by examining the bootstrap state transition rules of each state model (both SIP and BIP). Then the IP dispatcher creates an IP object for a selected state model. An IP object in ACE is a generic entity to control the execution of a given state model. It is created for a given state model and destroyed when a current state reaches a final state. An incoming message is interpreted by each IP.

Collaboration of ACE components to establish a relationship and to update a relationship are briefly described below, respectively.

Upon receiving an `advertise` performative at behavior sublayer (indicated as (1) in Figure 8), an IP dispatcher selects a state model for “Relationship establishment” BIP (indicated as (2) in Figure 8) and creates an IP object by specifying the selected state model (indicated as (3) in Figure 8). Creating a relationship record about a sender cyber-entity, the IP object registers the record in the relationship table (indicated as (4) in Figure 8).

Upon receiving a performative at service sublayer, an IP dispatcher selects a state model for “Relationship update” BIP and creates an IP object for the state model. “Relationship update” BIP stores information such as *Access-count* in a relationship record of a sender cyber-entity. Then, an IP dispatcher selects an appropriate state model of a SIP by examining the bootstrap state transition rule of all SIPs. Then, a SIP that satisfies the bootstrap state transition rule and a pre-condition is selected. An IP object for the SIP is created.

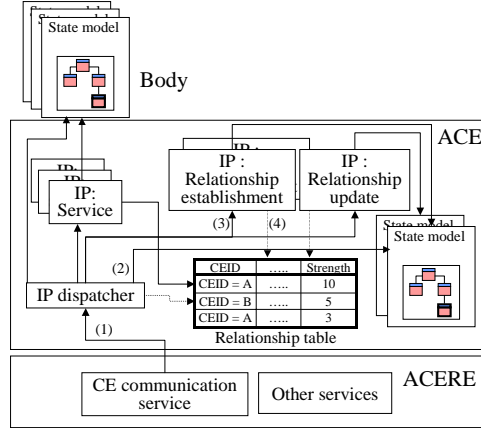


Fig. 8: The structure of ACE (Abstract Cyber-Entity)

5.3 Example sequence of a service emergence process

Figure 9 shows an example of message exchange sequence for providing a service, where cyber-entities representing a ticket sales (Ticket CE), coupon distribution (Coupon CE), and a user’s PDA (PDA CE) interact and provide a discount ticket sales service. Each cyber-entity implements different SIPs and exchanges Ja-Net ACL performatives.

(1) Initial relationship establishment The Ticket CE invokes “Advertisement” BIP when the access to its service exceeds a pre-defined threshold value, and sends an `advertise` message (indicated as (1) advertise in Figure 9).

Upon receiving an `advertise` message from the Ticket CE, the Coupon CE invokes “Relationship establishment” BIP. Then, an initial relationship with the Ticket CE is created by the Coupon CE.

(2) Service provision Triggered by the establishment of a new relationship, the “Make-partner” SIP of the Coupon CE sends a `request` performative to ask for discount sales, which is accepted through `inform` from Ticket CE (indicated as (2) `request(discount)` and (3) `inform(discount)` respectively in Figure 9). “Relationship establishment” BIP may be invoked by the Ticket CE and the Coupon CE when receiving performatives (1),(2) and (3) respectively. Consequently, the Ticket CE and the Coupon CE establish a relationship. Triggered by

the establishment of a new relationship, the Coupon CE invokes the “Distribute-coupon” SIP and sends a coupon information to a PDA CE (indicated as (4) `inform(ticket coupon)` in Figure 9). On receiving the coupon, the PDA CE invokes an UI (User Interface) SIP and displays the coupon on the screen. The PDA CE interacts with a human user, who issues a purchase request for the ticket using the coupon. The PDA CE sends a user purchase request to the Coupon CE (indicated as (5) `request(purchase)` in Figure 9). This request in turn is forwarded to the Ticket CE (indicated as (6) `request(purchase)` in Figure 9). The certificate of a purchase is sent back from the Ticket CE to the PDA CE via Coupon CE (indicated as (7) `inform(certificate)` and (8) `inform(certificate)` respectively in Figure 9). During this interaction, a new relationship is established between the PDA CE and the Coupon CE.

(3) Happiness feedback Upon receiving the certificate of ticket sales, the end user returns happiness to the Coupon CE, which in turn is forwarded to the Ticket CE (indicated as (9) `inform(happiness)` and (10) `inform(happiness)` respectively in Figure 9). Then, the Coupon CE modifies the strength of relationships to the PDA CE and to the Ticket CE, respectively. The Ticket CE modifies the strength of a relationship to the Coupon CE.

(4) Service emergence When relationships are strengthened based on feedback of happiness, the Coupon CE may selectively issue a coupon of a ticket provided by the Ticket CE, and send a coupon to the PDA CE. Also, the Ticket CE may selectively send its advertisement to the Coupon CE. Consequently, the Ticket CE, Coupon CE and the PDA CE form a group to provide a discount ticket service.

6 Future work

We are currently implementing basic mechanisms for service emergence described in this paper. Issues that we are currently investigating include how to represent pre-conditions and state transition rules. We are also investigating algorithms for partner selection and strength evaluation, and various algorithms will be empirically evaluated for their efficiency in service composition and emergence. To realize the self-organizing

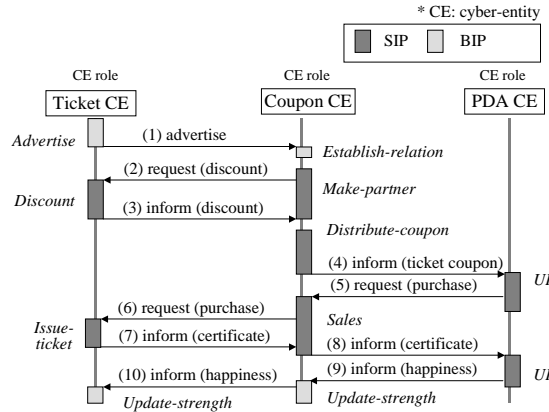


Fig. 9: Example of service composition sequence

capability of cyber-entities, we are investigating mechanisms to achieve natural selection.

Reference

- [1] Mark Weiser, "The Computer for the Twenty-First Century," Scientific American, pp.94–101, Sep. 1991.
- [2] Tatsuya Suda , Tomoko Itao, Tetsuya Nakamura , and Masato Matsuo, "A Network for Service Evolution and Emergence: Jack-in-the-Net," Journal of IEICEJ, B , Vol. J84-B, No.3, pp. 310-320, 2001.
- [3] Tomoko Itao, Tatsuya Suda, Tetsuya Nakamura, Masato Matsuo, "Adaptive Networking Service Architecture for Service Emergence," Symposium on Applications and the Internet Workshops (SAINT),Jan., 2001.
- [4] K. Kaneko and T. Ikegami, "Evolution of Complex Systems," Asakura Publishing, Japan, 1998.
- [5] The BNA project web site,
<http://netresearch.ics.uci.edu/bionet>
- [6] Michael Wang and Tetsuya Suda, "The Bio-Networking Architecture: A Biologically Inspired Approach to the Design of Scalable, Adaptive, and Survivable/Available Network Applications," Symposium on Applications and the Internet (SAINT), pp.43–53, Jan., 2001.
- [7] T. Finin and G. Wiederhold, "An overview of KQML: A knowledge query and manipulation language," Dept. of Computer Science, Stanford University, 1993.
- [8] Yannis Labrou, "Semantics for an Agent Communication Language," Ph.D. Thesis, UMBC, 1996.

- [9] Nelson Minar, Matthew Gray, Oliver Roup, Raffi Krikorian, and Pattie Maes, "Hive: Distributed Agents for Networking Things," Proc. of ASA/MA, 1999.
- [10] Mihai Barbuceanu and Mark S. Fox, "COOL: A Language for Describing Coordination in Multi Agent Systems," Proc. of the First International Conference on Multi-Agent Systems, 1995.
- [11] Takahiro Kawamura, Yasuyuki Tahara, Tetsuo Hasegawa, Akihiko Ohsuga, and Shinichi Honiden, "Bee-gent : Bonding and Encapsulation Enhancement Agent Framework for Development of Distributed Systems," Journal of IEICEJ, D-I , Vol. J82-D-I, No.9, 1999.
- [12] Kinji Mori, "Autonomous Decentralized Systems: Concept, Data Field Architecture and Future Trends," Proc. of ISADS'93, 1993.
- [13] Barbara Hayes-Roth, " A Blackboard Architecture for Control," Artificial Intelligence, 1985, 26.
- [14] FIPA-ACL web site, <http://www.fipa.org>
- [15] XML web site, <http://www.xml.org>
- [16] Austin J. L., "How to Do Things with Words," Clarendon Press 1962.
- [17] Searle J. R., "Speech Acts," Cambridge University Press, 1969.