An Evolutionary Game Theoretic Framework for Adaptive, Cooperative and Stable Network Applications

Chonho Lee[1], Junichi Suzuki[1], and Athanasios V. Vasilakos[2]

[1] Department of Computer Science
University of Massachusetts, Boston, USA
{chonho,jxs}@cs.umb.edu
[2] Department of Computer and Telecommunication Engineering
University of Western Macedonia, Greece
vasilako@ath.forthnet.gr

Abstract. This paper investigates a bio-inspired framework, iNet-EGT/C, to build adaptive, cooperative and stable network applications. In this framework, each application is designed as a decentralized set of agents, each of which provides a functional service and possesses biological behaviors such as migration, replication and death. iNet-EGT/C implements an adaptive behavior selection mechanism for agents. Its selection process is modeled as a series of evolutionary games among behaviors. iNet-EGT/C allows agents to seek to operate at evolutionarily stable equilibria and adapt to dynamic networks by invoking evolutionarily stable behaviors. It is theoretically proved that each behavior selection process retains stability (i.e., reachability to at least one evolutionarily stable equilibrium). iNet-EGT/C leverages the notion of coalitions for agents to select behaviors as coalitional decisions in a cooperative manner rather than individual decisions in a selfish manner. This cooperative behavior selection accelerates agents' adaptation speed by up to 78%.

Key words: Evolutionary game theory, Adaptive and cooperative network applications, Stability in adaptive networking

1 Introduction

Large-scale network applications such as data center applications and cloud computing applications are required to autonomously adapt to dynamic changes in network conditions such as workload and resource availability [1]. To address this requirement, this paper investigates a bio-inspired framework for autonomous adaptive network applications. In the framework, each application is designed as a decentralized group of software agents. This is analogous to a bee colony (an application) consisting of multiple bees (agents). Each agent implements a functional service (e.g., web service) and follows biological behaviors such as migration, replication and death. This paper focuses on an adaptive behavior selection mechanism for agents, called iNet-EGT/C (iNet-EGT, coalitional edition). It is

designed after immunological *antigen-antibody reaction*, which produces antibodies specific to antigens (e.g., viruses) for eliminating them. iNet-EGT/C models a set of network conditions (e.g., workload and resource availability) as an antigen and models an agent behavior as an antibody. Each agent contains iNet-EGT/C as its behavior selection mechanism (or its immune system). iNet-EGT/C allows each agent to autonomously sense its surrounding network conditions (an antigen) and select a behavior (an antibody) suitable for the sensed conditions. For example, agents may invoke the replication behavior at the network hosts that accept a large number of user requests for their services. This leads to the adaptation of agent availability. As a result, agents can improve their throughput. Also, agents may invoke the migration behavior to move toward the network hosts that receive a large number of user requests for their services. This leads to the adaptation of agent locations; agents can improve their response time.

In iNet-EGT/C, antigen-antibody reaction (i.e., behavior selection) process is modeled with evolutionary game theory. Each agent contains a set (or population) of behaviors. In the population, randomly-selected two behaviors play a game. The game distinguishes a winning behavior and a losing behavior based on their *payoffs*, which indicate the likelihood for the behaviors to adapt an agent to the current network conditions. The winner replicates itself and increases its share in the population. The loser disappears in the population. This way, the population state (i.e., behavior distribution in the population) changes as a series of games are repeatedly performed. iNet-EGT/C theoretically proves that the population state converges to an *evolutionarily stable equilibrium*. It is the state that, regardless of the initial population state, the population always converges to. In that state, no other behaviors except a dominant behavior, called an *evolutionarily stable behavior*, can dominate the population. Thanks to this property, iNet-EGT/C allows each agent to seek to operate at equilibria by invoking evolutionarily stable behaviors as rational and adaptive decisions.

iNet-EGT/C leverages the notion of *coalitions* to compute payoffs. A coalition is a group of agents that reside on neighboring hosts. Each agent computes the payoffs for its behaviors based on (1) its surrounding network conditions and (2) the behaviors that agents in its coalition intend to invoke. This way, agents seek evolutionarily stable behaviors as coalitional decisions in a cooperative manner rather than individual decisions in a selfish manner. This coalitional payoff computation is designed to accelerate the adaptation speed of agents.

This paper describes the design of iNet-EGT/C and evaluates it through theoretical and simulation studies. Both studies verify that iNet-EGT/C allows agents to seek to operate at evolutionarily stable equilibria and adapt to dynamic networks. The notion of coalitions accelerates agents' adaptation speed by 78%.

2 Related Work

iNet-EGT/C extends its predecessors: iNet [2] and iNet-EGT [3]. It shares the same goal with them; bio-inspired adaptive behavior selection However, they are different in their approaches to the goal. iNet takes a stochastic approach with

a genetic algorithm; it does not guarantee stability (i.e., reachability to at least one equilibrium) in behavior selection. iNet-EGT takes an evolutionary game theoretic approach; however, it does not consider the notion of agent coalitions.

Game theoretic approaches have been studied for several aspects in networks; for example, job allocation [4], security [5–7] and routing [8, 9]. They seek rational networking strategies in static networks, but do not consider adaptation in dynamic networks. [9] does not guarantee stability in its strategic decision making. None of [4–9] considers the notion of coalitions in games.

[10, 11] leverage evolutionary game theory for adaptive routing in dynamic networks. Unlike [10,11], iNet-EGT/C performs the mutation operation to better adapt to future changes in network conditions. Moreover, iNet-EGT/C considers the notion of agent coalitions, which is beyond the scope of [10, 11].

[12–16] study behavior selection mechanisms for agent-based systems. [12,13] propose rule-based mechanisms, which are similar to iNet-EGT/C in that they implement deterministic behavior selection. However, unlike [12, 13], iNet-EGT/C guarantees stability in behavior selection. Moreover, in general, it maintains lower time complexity in behavior selection ($O(k)$ where k denotes the number of games) than rule-based mechanisms ($O(MlogM)$ where M denotes the number of behavior types). [14–16] propose non-deterministic behavior selection with stochastic algorithms. Due to their stochastic processes, they fail to retain stability; they often search inconsistent solutions under the same problem setting (or the same set of network conditions) in different runs/trials. In contrast, iNet-EGT/C considers determinism in behavior selection to retain stability.

3 Preliminaries: Evolutionary Game Theory

In a conventional game, the objective of a player is to choose a strategy that maximizes its payoff. In contrast, evolutionary games are played repeatedly by players randomly drawn from a population [17]. This section overviews key elements in evolutionary games: evolutionarily stable strategies and replicator dynamics.

3.1 Evolutionarily Stable Strategies (ESS)

Suppose all players in the initial population are programmed to play a certain (incumbent) strategy k. Then, let a small population share of players, $x \in (0, 1)$, mutate and play a different (mutant) strategy ℓ. When a player is drawn for a game, the probabilities that its opponent plays k and ℓ are $1 - x$ and x, respectively. Thus, the expected payoffs for the player to play k and ℓ are $U(k, x\ell + (1-x)k)$ and $U(\ell, x\ell + (1-x)k)$, respectively.

Definition 1. *A strategy k is called evolutionarily stable if, for every strategy $\ell \neq k$, a certain $\bar{x} \in (0, 1)$ exists, such that the inequality*

$$U(k,\ x\ell + (1-x)k) > U(\ell,\ x\ell + (1-x)k) \tag{1}$$

holds for all $x \in (0, \bar{x})$.

If the payoff function is linear, Equation 1 derives

$$(1-x)U(k,k) + xU(k,\ell) > (1-x)U(\ell,k) + xU(\ell,\ell) \tag{2}$$

If x is close to zero, Equation 2 yields either

$$U(k,k) > U(\ell,k),\ \textit{or}\ U(k,k) = U(\ell,k)\ \textit{and}\ U(k,\ell) > U(\ell,\ell) \tag{3}$$

This indicates that a player associated with the strategy k gains a higher payoff than the ones associated with the other strategies. Thus, no players can benefit by changing their strategies from k to the others. This means an ESS is a solution on a Nash equilibrium. An ESS is a strategy that cannot be invaded by any alternative (mutant) strategies that have small population shares.

3.2 Replicator Dynamics

The replicator dynamics describes how population shares associated with different strategies evolve over time [18]. Let $\lambda_k(t) \geq 0$ be the number of players that plays the strategy $k \in K$, where K is the set of available strategies. The total population of players is given by $\lambda(t) = \sum_{k=1}^{|K|} \lambda_k(t)$. Let $x_k(t) = \lambda_k(t)/\lambda(t)$ be the population share of players that play k at time t. The population state is defined by $\mathbf{x}(t) = [x_1(t), \cdots, x_k(t), \cdots, x_K(t)]$. Given $\mathbf{x}$, the expected payoff of playing k is denoted by $U(k, \mathbf{x})$. The population's average payoff, which is same as the payoff of a player drawn randomly from the population, is denoted by $U(\mathbf{x}, \mathbf{x}) = \sum_{k=1}^{|K|} x_k \cdot U(k, \mathbf{x})$. In the replicator dynamics, the dynamics of the population share x_k is described as follows. $\dot{x}_k$ is the time derivative of x_k.

$$\dot{x}_k = x_k \cdot [U(k, \mathbf{x}) - U(\mathbf{x}, \mathbf{x})] \tag{4}$$

This equation states players increase (or decrease) their population shares when their payoffs are higher (or lower) than the population's average payoff.

Theorem 1. *If a strategy k is strictly dominated, then $x_k(t)_{t\to\infty} \to 0$.*

A strategy is said to be *strictly dominant* if its payoff is strictly higher than any opponent strategies. As its population share grows, it dominates the population over time. Conversely, a strategy is said to be *strictly dominated* if its payoff is lower than that of a strictly dominant strategy. Thus, strictly dominated strategies disappear in the population over time.

There is a close connection between Nash equilibria and the steady states of the replicator dynamics, in which the population shares do not change over time. Since no players change their strategies on Nash equilibria, every Nash equilibrium is a steady state of the replicator dynamics. As described in Section 3.1, an ESS is a solution on a Nash equilibrium. Thus, an ESS is a solution at a steady state of the replicator dynamics. In other words, an ESS is the strictly dominant strategy in the population on a steady state.

In iNet-EGT/C, an agent maintains a population of its behaviors. In a single behavior selection process, behaviors are randomly drawn from the population to play games repeatedly until the population reaches a steady state. Then, iNet-EGT/C allows each agent to identify a strictly dominant behavior in the population and invoke it as an evolutionarily stable behavior (or ESS).

4 iNet-EGT/C

The immune system is an adaptive defense mechanism that regulates the body against dynamic environmental changes such as antigen invasions. Through a number of interactions among antibodies, the immune system evokes *antigen-antibody reaction* to produce antibodies specific to detected antigens. In each interaction, an antibody stimulates or suppresses another one according to its *affinity* to an antigen. A stimulated antibody replicates itself and increases its population. Conversely, a suppressed one dies and decreases its population. This way, the population of specific antibodies rapidly increases following the recognition of an antigen and decreases again after eliminating the antigen. Adaptive immune response is an emergent product of interactions among antibodies.

iNet-EGT/C is designed after antigen-antibody reaction. Each agent contains iNet-EGT/C as its own behavior selection mechanism (or as its own immune system). An antigen is modeled as a set of network conditions: $C = \{c_1, c_2, \cdots, c_L\}$ where L denotes the number of network conditions that each agent senses on a host. This paper considers three network conditions ($L = 3$):

- *Queue length*: The number of user requests in a request queue, which each host operates to store incoming user requests until they are processed by the agents running on the same host.
- *Workload change rate*: The rate of workload change per a unit time. Workload is computed as the number of incoming user requests per minute given to the agents running on the same host.
- *Resource utilization*: Memory consumption, in percentage, by the agents running on the same host.

An antibody is modeled as an agent behavior. $B = \{b_1, b_2, \cdots, b_M\}$ denotes a set of available behavior types. This paper considers four types ($M = 4$):

- *Migration*: Agents may migrate from one platform to another.
- *Replication*: Agents may make a copy of themselves. A replicated (child) agent is placed on the host that its parent agent resides on.
- *Death*: Agents may die and disappear in the network. When an agent dies, its underlying host releases the resources (e.g. memory space) it consumes.
- *Do-nothing*: Agents may choose to do nothing.

Each agent maintains a population (P) of behaviors, each of which is of a certain behavior type. The population's size is given by $N = \sum_{b \in B} n_b$ where n_b denotes the number of behaviors of the behavior type $b \in B$.

4.1 Evolutionary Games in iNet-EGT/C

An interaction (stimulation or suppression) between antibodies is modeled as an evolutionary game between behaviors. Listing 1 shows how iNet-EGT/C allows each agent to select/invoke a specific behavior under a given set of network conditions through a series of evolutionary games.

The population P is initialized with `initializePopulation()` (Line 2). Initially, all behavior types have the equal population share. ($n_{b \in B}$ is equal for every b.) `randomlySelect()` draw two behaviors randomly from the population (Line 6), and `performGame()` distinguishes them to a winning one and a losing one according to their payoffs, which indicate the likelihood for the behaviors to adapt an agent to the current network conditions (Line 8). The notion of payoffs is modeled after the notion of affinity in the immune system. The loser behavior is suppressed and disappears in the population. The winner behavior is stimulated and replicated to increase its population share (Line 9). It is also mutated at the probability of p_m (Lines 10 and 11).

A behavior whose population share is the largest is called a *current major behavior*. An agent invokes the current major behavior when its population share (x_b) exceeds a threshold (t_s) (Lines 15 and 16). Similar to the immune system, the behavior selection in iNet-EGT/C is designed as an emergent product of games (interactions) among behaviors (antibodies).

```
1  function selectBehavior()
2    P ← initializePopulation(N)
3    while true do
4      W ← ∅
5      for i ← 1 to |P|/2 do
6        {behavior_1, behavior_2} ← randomlySelect(P)
7        P ← P \ {behavior_1, behavior_2}
8        winner ← performGame(behavior_1, behavior_2)
9        replica ← replicate(winner)
10       if random() ≤ p_m then
11         replica ← mutate(replica)
12       W ← W ∪ winner ∪ replica
13     end for
14     P ← W
15     if ∃b where b ∈ P and x_b > t_s then
16       return b
17     end if
18   end while
19 end function
20
21 function performGame(behavior_1, behavior_2)
22   C ← getNetworkConditions()
23   O ← getCurrentMajorBehaviors()
24   p_1 ← |{behavior_1, R}| − rank(behavior_1, O, C)
25   p_2 ← |{behavior_1, R}| − rank(behavior_2, O, C)
26   if p_1 > p_2 then return behavior_1
27   else if p_1 < p_2 then return behavior_2
28   else return randomlySelect({behavior_1, behavior_2})
29 end function
```

Listing 1. Pseudocode of Behavior Selection

4.2 Coalitional Payoff Functions

In order to compute the payoffs of an agent's behaviors, iNet-EGT/C considers the agent's coalition, which consists of the agents running on the local host and direct neighbor hosts. Equation 5 shows the coalitional payoff function $F_i(b)$ for the behavior type b of agent i under a set of network conditions C.

$$F_i(b) = |\{b, O\}| - rank(b, O, C) \tag{5}$$

$O = \{o_1, o_2, \cdots, o_q\}$ denotes a set of current major behaviors of the other agents in the same coalition. (q indicates the number of those agents.) See also Line 22 to 25 in Listing 1.

The function `rank()` in Equation 5 compares b and O (i.e., $q + 1$ behaviors in total) with respect to *domination factors* and yields b's *domination rank*. The domination ranking is a ranking scheme that considers the Pareto optimality among multiple factors (or objectives) [19]. A behavior $b \in B$ is said to dominate a behavior $b' \in B$ if b's factor values are better than, or equal to, b''s in all domination factors, and b's factor values are better than b''s in at least one domination factors. This paper considers three domination factors:

- *Queue length*
- *Resource utilization*
- *Load balancing*: The variance of queue lengths in the local and neighboring hosts

A behavior b's factor values are computed as the network conditions in the case where agent i invokes b and the other agents in its coalition invokes O.

Figure 1 shows an example that illustrates how to use the proposed coalitional payoff functions. In this example, four agents (a, b, c and d) runs on a host. (For simplicity, this example ignores neighbor nodes.) Agent b's current major behavior is migration (M). c's and d's are death (D) and replication (R), respectively. Thus, agent a's payoff functions ($F_a(R)$, $F_a(M)$, $F_a(N)$ and $F_a(D)$) are formulated for its four behaviors, as shown in Figure 1.

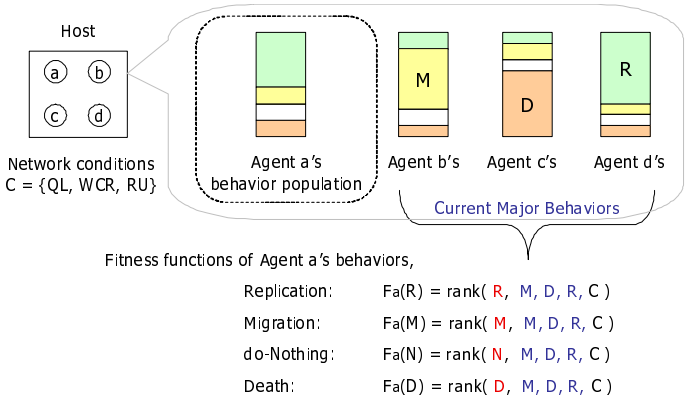

Fig. 1. Example Payoff Functions

Figure 2 depicts an example domination ranking among agent a's four behaviors. In a three dimensional space whose axises represent domination factors, each behavior is plotted based on its factor values. (In this space, the smaller factor values, the better.) For example, R's factor values are computed as the network conditions in the case where agent a invokes the replication behavior and the other three agents invoke their current major behaviors.

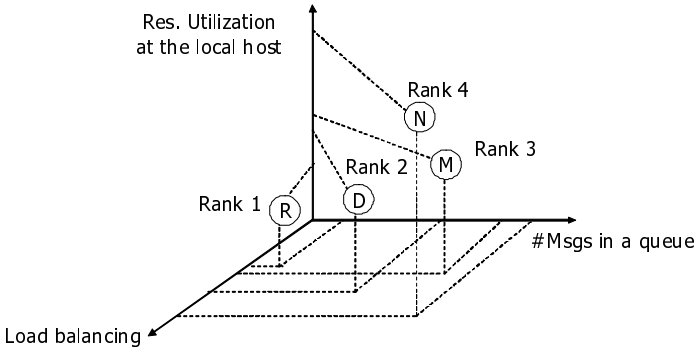

Fig. 2. An Example Domination Ranking

R dominates all the other behaviors; it is given a rank value of 1. D dominates N, but does not dominate M. However, D is superior to M in two factors, and M is superior in one factor. Thus, D and M are given the rank values of 2 and 3, respectively. N is given a rank value of 4. Given rank values, $F_a(R) = 4-1 = 3$, $F_a(D) = 4-2 = 2$, $F_a(M) = 4-3 = 1$, and $F_a(D) = 4-4 = 0$.

5 Stability Analysis

This section theoretically analyzes the stability of cooperative behavior selection in iNet-EGT/C. More specifically, this theoretical analysis is intended to prove that the population states of all agents in the same coalition converge to evolutionarily stable equilibria. The proof consists of three steps: (1) designing a set of differential equations that describe the dynamics of the population state, (2) proving a cooperative behavior selection process has equilibria, and (3) proving the the equilibria are asymptotically (or evolutionarily) stable. The proof uses the following terminology and variables.

- $X^{(i)}(t) = \{x_1^{(i)}(t), x_2^{(i)}(t), \cdots, x_M^{(i)}(t)\}$ denotes the population state of agent i at time t, where $x_b = \frac{n_b}{N}$ is the population share of $b \in B$ ($\sum_b x_b = 1$).
- $F_b^{(i)}$ denotes the coalitional payoff of $b \in B$ in agent i.
- $p_k^b = x_b \cdot \phi(F_b - F_k)$ denotes the probability that a behavior of $b \in B$ replicates itself by winning a game against a behavior of $k \in B$. $\phi(F_b - F_k)$ is the conditional probability that the fitness value of b is higher than that of k.

The dynamics of b's population share is described as follows.

$$\begin{aligned}\dot{x}_b &= \sum_{k \in B, k \neq b} \{x_k \cdot p_k^b - x_b \cdot p_b^k\} = x_b \sum_{k \in B, k \neq b} x_k \{\phi(F_b - F_k) - \phi(F_k - F_b)\} \\ &= x_b \sum_{k \in B, k \neq b} x_k \cdot c_{bk} \quad \textit{where } c_{bk} = \phi(F_b - F_k) - \phi(F_k - F_b)) \end{aligned} \tag{6}$$

Note that, if k is strictly dominated, $x_k(t)_{t \to \infty} \to 0$. (See Theorem 1.)

Theorem 2. *The population state of an agent converges to an equilibrium.*

Proof. It is true that, given the design of fitness functions (Figure 1), different behavior types have different fitness values under the same set of network conditions. In other words, given a particular set of network conditions, a behavior type becomes strictly dominant. Assume that $F_1 > F_2 > \cdots > F_M$, and by Theorem 1, the population state converges to an equilibrium: $X(t)_{t\to\infty} = \{x_1(t), x_2(t), \cdots, x_M(t)\}_{t\to\infty} \to \{1, 0, \cdots, 0\}$.

Theorem 3. *The equilibrium found in Theorem 2 is asymptotically stable.*

Proof. At an equilibrium where $X = \{1, 0, \cdots, 0\}$, Equation 6 can be downsized by substituting $x_1 = 1 - x_2 - \cdots - x_M$.

$$\dot{z}_b = z_b[c_{b1}(1 - z_b) + \sum_{i=2, i\neq b}^{M} z_i \cdot c_{bi}] \quad where\ b = 2, ..., M \tag{7}$$

$Z(t) = \{z_2(t), z_3(t), \cdots, z_M(t)\}$ denotes the downsized population state. Given Theorem 1, $Z(t)$ converges to an equilibrium: $Z(t)_{t\to\infty} = Z_{eq} = \{0, 0, \cdots, 0\}$.

If all Eigenvalues of the Jaccobian matrix of $Z(t)$ has negative Real parts, Z_{eq} is asymptotically stable. The Jaccobian matrix J's elements are:

$$J_{bk} = \left[\frac{\partial \dot{z}_b}{\partial z_k}\right]_{|Z=Z_{eq}} = \left[\frac{\partial z_b[c_{b1}(1 - z_b) + \sum_{i=2, i\neq b}^{M} z_i \cdot c_{bi}]}{\partial z_k}\right]_{|Z=Z_{eq}} \quad where\ b, k = 2, ..., M \tag{8}$$

Therefore, J is given as follows, where $c_{21}, c_{31}, \cdots, c_{M1}$ are J's Eigenvalues.

$$J = \begin{bmatrix} c_{21} & 0 & \cdots & 0 \\ 0 & c_{31} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & c_{M1} \end{bmatrix} \tag{9}$$

$c_{b1} = -\phi(F_1 - F_b) < 0$ for every b; therefore, Z_{eq} is asymptotically stable.

6 Simulation Evaluation

This section evaluates iNet-EGT/C through simulations. Figure 3 shows a simulated server farm (or cloud computing environment) that consists of 100 (10×10) hosts in a grid topology. User requests travel from users to agents via user access point. This simulation study assumes that a single (emulated) user runs on the access point and transmits user requests to agents. At the beginning of a simulation, four agents are deployed on randomly-selected hosts. Each agent has

its own iNet-EGT/C that contains the population of 100 behaviors. ($N = 100$ in Listing 1). 25 behaviors are of each of four behavior types: migration, replication, death and do-nothing. Mutation rate (p_m in Listing 1) and behavior selection threshold (t_s in Listing 1) are set to 0.05 and 0.95, respectively.

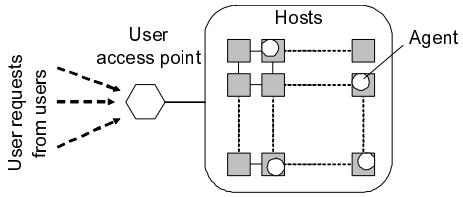

Fig. 3. Simulated Server Farm

Figure 4 shows a trace of workload (the number of user requests) given to agents. It follows an empirical workload measurement at www.ibm.com [20]. The largest workload spike occurs at 12:00 from 3,000 to 9,000 messages/min.

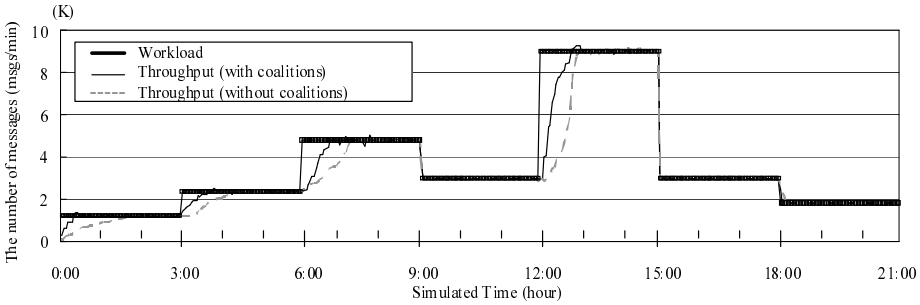

Fig. 4. Workload and Throughput

6.1 Adaptability and Stability

Figure 5 shows how the population state (i.e., behavior distribution) changes over time (from 0:00 to 6:00) in an agent deployed at the beginning of a simulation. The number of replication behaviors increases in the first 15 minutes, and the population converges to an evolutionarily stable state. This means that the agent in question replicates itself to process a given workload. (Initially-deployed four agents are not enough to efficiently process a given workload.) Then, the do-nothing behavior takes over the replication behavior to dominate the population; the population converges to another evolutionarily stable state. At this point, agents have replicated enough to process the current workload; the agent in question does not replicate itself anymore. As illustrated in Figure 5, iNet-

EGT/C allows agents to successfully seek evolutionarily stable equilibria in their behavior selection according to dynamic network conditions.

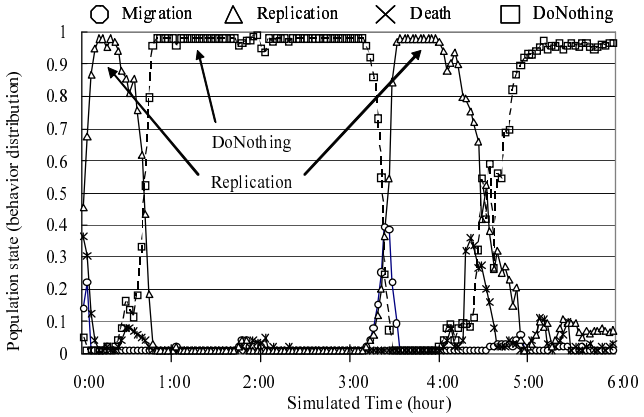

Fig. 5. Changes in Population State

Figure 6 shows how agent availability (i.e., the number of agents) changes dynamically. It increases when the replication behavior dominates the behavior population (e.g., around 0:00 and 3:00; See Figure 5.) in response to workload spikes. Conversely, agent availability decreases when the death behavior dominates the behavior population (e.g., around 9:00 and 15:00) in response to workload drops. Figure 6 demonstrates that iNet-EGT/C allows agents to dynamically adapt their availability by invoking their behaviors according to the evolutionarily stable states they are on.

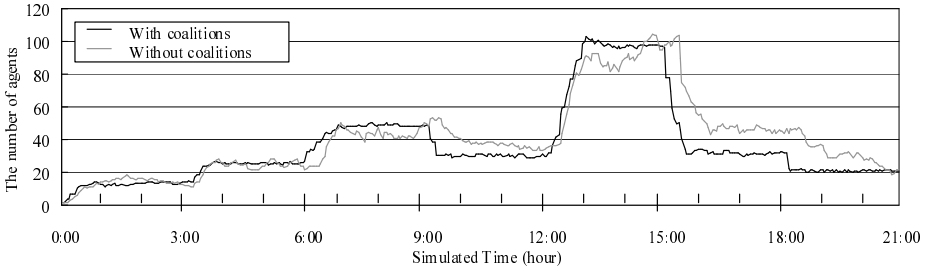

Fig. 6. Agent Availability

Figures 4 and 7 show the throughput (i.e., the number of processed requests) and response time that agents yield. At the beginning of a simulation, they yield low throughput and high response time because four agents are not enough to efficiently process all user requests. During a simulation, throughput and response time degrade when workload spikes. However, as agents perform their

behaviors by seeking evolutionarily stable equilibria, they adapt their throughput and response time to dynamic workload changes.

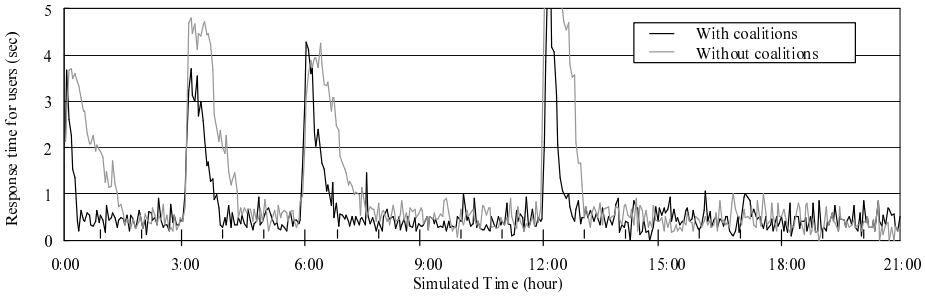

Fig. 7. Response Time

Figure 8 depicts the average behavior population state among agents: $S_{avg}(t) = \frac{1}{A(t)} \sum_i \max_{b \in B}\{x_b(t)\}$. $A(t)$ denotes the total number of agents running in the network. i and b index agents and behavior types, respectively. $\max_{b \in B}\{x_b(t)\}$ denotes the share of the current major behavior in an agent's behavior population. $S_{avg}(t)$ increases as the current major behavior's share increases in each agent. It approaches 1.0 (e.g., > 0.95) when the current major behaviors dominate behavior populations and remain effective for agents to adapt to the current network conditions. This means that agents' behavior population states reaches evolutionarily stable equilibria. $S_{avg}(t)$ decreases when the current major behaviors are no longer effective and the other behaviors take over to dominate behavior populations. For example, $S_{avg}(t)$ remains high from 1:00 to 3:00 because agents have adapted to network conditions by 1:00 and the do-nothing behavior is effective until 3:00 (See also Figure 5.) However, the do-nothing behavior becomes ineffective when workload spikes at 3:00; $S_{avg}(t)$ decreases until another behavior (the replication behavior in this case; See Figure 5.) takes over and dominates the behavior population. $S_{avg}(t)$ stays over 0.95 during 82% of the total simulation time. Figure 8 demonstrates that iNet-EGT/C allows agents to seek to operate at evolutionarily stable equilibria in dynamic networks.

6.2 Impacts of Agent Coalitions on Adaptability

Figures 4 and 7 illustrate agents' throughput and response time with the notion of coalitions disabled. Both figures show that, upon workload spikes (at 0:00, 3:00, 6:00 and 12:00), agents adapt their performance faster by computing payoffs in a cooperative manner with the notion of coalitions. Figure 9 depicts throughput in percentile (i.e., ratio of throughput over workload). The throughput of 100% indicates that agents process all of user requests. Similar to Figure 4, Figure 9 demonstrates that agents improve their throughput faster with coalitions enabled. With coalitions enabled, throughput stays over 90% during 85% of the total simulation time, while 69% with coalitions disabled.

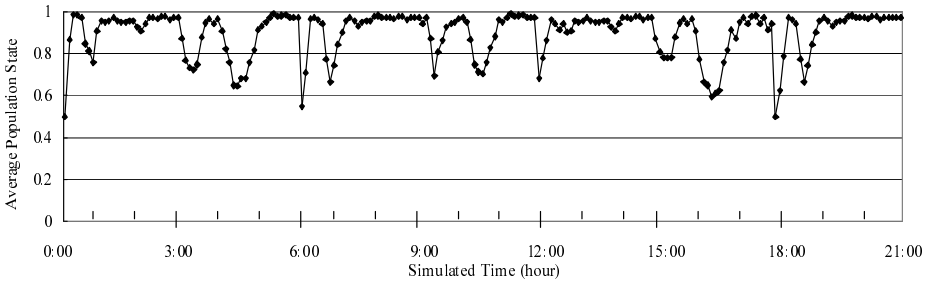

Fig. 8. Average Behavior Population State

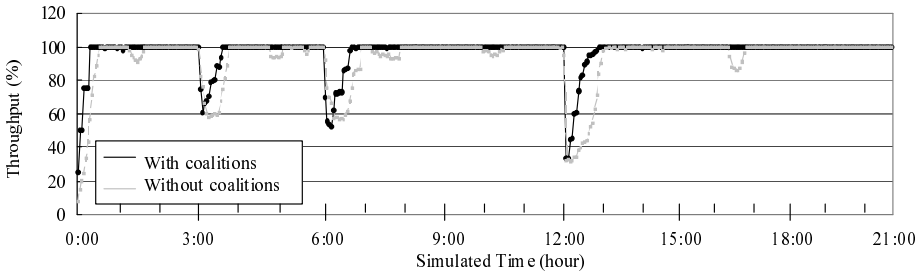

Fig. 9. Throughput (%)

Table 1 shows how soon agents yield the throughput of 100% and 75% upon workload spikes. Agents consistently accelerate their throughput adaptation by leveraging the notion of coalitions. The average speedups of 100% and 75% throughput adaptation are 138% and 241%, respectively.

Table 1. Adaptation Speed in Throughput (Minutes)

Workload Spike at	0:00		3:00		6:00		12:00	
Throughput Improvement Rate	100%	75%	100%	75%	100%	75%	100%	75%
Without coalitions	32	22	46	32	64	36	58	48
With coalitions	18	10	36	12	48	14	51	22
Speedup	178%	220%	127%	267%	133%	257%	114%	218%

Figure 6 depicts agent availability with coalitions disabled. It verifies that, upon workload spikes and drops (at 0:00, 3:00, 6:00, 9:00, 12:00, 15:00 and 18:00), agents adapt their availability faster by leveraging the notion of coalitions.

Figure 10 shows agent availability in percentile. It is computed as the ratio of expected throughput over workload. The expected throughput is calculated as $A(t)/t_{service}$ where $A(t)$ denotes the total number of agents running in the network and $t_{service}$ denotes the time that an agent is expected to spend to pro-

cess a single user request. If agent availability is over or under 100%, the number of agents is excess or insufficient, respectively, to process a given workload. The agent availability of 100% indicates that the number of agents perfectly fits with the current workload. Similar to Figure 6, Figure 10 demonstrates that agents adapt their availability faster with coalitions enabled. With coalitions enabled, agent availability stays at 100% during 88% of the total simulation time, while 73% with coalitions disabled.

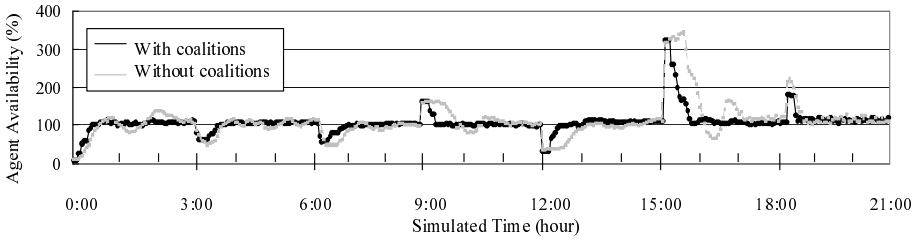

Fig. 10. Agent Availability (%)

Table 2 shows how soon agents yield 100% availability upon workload spikes and drops. Agents consistently accelerate their availability adaptation by leveraging the notion of coalitions. The average speedup is 144%.

Table 2. Adaptation Speed in Agent Availability (Minutes)

Workload Spike/Drop at	0:00	3:00	6:00	9:00	12:00	15:00	18:00
Without coalitions	32	42	62	50	56	68	34
With coalitions	20	30	46	28	50	54	22
Speedup	160%	140%	135%	179%	112%	126%	155%

7 Conclusion

This paper proposes and evaluates an evolutionary game theoretic framework, iNet-EGT/C, which aids building adaptive, cooperative and stable network applications. Both theoretical and simulation studies demonstrate that iNet-EGT/C allows network applications to seek to operate at evolutionarily stable equilibria and adapt to dynamic network conditions. The notion of agent coalitions in payoff computation allows agents to yield the speedup of up to 178% in adaptation to dynamic network conditions.

References

1. A. Weiss, "Computing in the clouds," *ACM netWorker Magazine*, vol. 11(4), 2007.
2. C. Lee and J. Suzuki, "An immunologically-inspired autonomic framework for self-organizing and evolvable network applications," *ACM Trans. Autonomous and Adaptive Systems*, vol. 4(4), 2009.
3. C. Lee, A. V. Vasilakos, and J. Suzuki, "iNet-EGT: An evolutionarily stable adaptation framework for network applications," in *Proc. of Int'l Conference on Bio-inspired Models of Network, Information and Computing Systems*, December 2009.
4. R. Subrata, A. Y. Zomaya, and B. Landfeldt, "Game theoretic approach for load balancing in computational grids," *IEEE Trans. Parallel Distrib. Syst.*, 19(1), 2008.
5. M. Kodialam and T.V.Lakshman, "Detecting network intrusions via sampling: a game theoretic approach," in *Proc. of IEEE Int'l Conf. on Computer Comm.*, 2003.
6. A. Agah, K. Basu, and S. Das, "Preventing dos attack in sensor networks: a game theoretic approach," in *Proc. of IEEE Int'l Conf. on Comm.*, May 2005.
7. H. Otrok, M. Mehrandish, and e. a. C. Assi, "Game theoretic models for detecting network intrusions," *Elsevier Computer Communications*, vol. 31(10), June 2008.
8. R. Kannan and S. Iyengar, "Game theoretic models for reliable path-length and energy constrained routing with data aggregation in wireless sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 22(6), 2004.
9. L.Yan and S.Hailes, "Cooperative packet relaying model for wireless ad hoc networks," in *Proc. of ACM International workshop on Foundations of wireless ad hoc and sensor networking and computing*, May 2008.
10. A.V.Vasilakos and M. Anastasopoulos, "Application of evolutionary game theory to wireless mesh networks," in *Advances in Evolutionary Computing for System Design*, L. C. Jain, V. Palade, and D. Srinivasan, Eds. Springer, 2007.
11. M. P. Anastasopoulos, D. K. Petraki, R. Kannan, and A. V. Vasilakos, "Tcp throughput adaptation in wimax networks using replicator dynamics," *IEEE Transactions on Systems, Man, and Cybernetics*, January 2010.
12. Z. Li and M. Parashar, "Rudder: A rule-based multi-agent infrastructure for supporting autonomic grid applications," in *Proc. of IEEE International Conference on Autonomic Computing*, May 2004.
13. D. Hagimont, S. Bouchenak, N. Palma, and C. Taton, "Self-sizing of clustered databases," in *Proc. of IEEE International Symposium on World of Wireless, Mobile and Multimedia Networks*, June 2006.
14. Y. Wang, S. Li, Q. Chen, and W. Hu, "Biology inspired robot behavior selection mechanism: Using genetic algorithm," in *Proc. of CASS International Conference on Life System Modeling and Simulation (LSMS)*, September 2007.
15. B. D. Damas and L. Custódio, "Emotion-based decision and learning using associative memory and statistical estimation," *Informatica*, vol. 27(2), 2003.
16. K.-J. Kim and S.-B. Cho, "Bn+bn: Behavior network with bayesian network for intelligent agent," in *Proc. of AFOSR Australian Joint Conference on Artificial Intelligence*, December 2003.
17. J. W. Weibull, *Evolutionary Game Theory*. MIT Press, 1996.
18. P. Taylor and L. Jonker, "Evolutionary stable strategies and game dynamics," *Elsevier Mathematical Biosciences*, vol. 40(1), 1978.
19. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6(2), 2002.
20. J. Chase, D. Anderson, P. Thakar, A. Vahdat, and R. Doyle, "Managing energy and server resources in hosting centers," in *Proc. of ACM Symposium on Operating Systems Principles*, October 2001.