2013 IEEE 25th International Conference on Tools with Artificial Intelligence

Network Layer-Oriented Task Allocation for Multiagent Systems in Undependable Multiplex Networks

Yichuan Jiang, *IEEE Senior Member*, Yifeng Zhou and Yunpeng Li
Laboratory for Complex Systems and Social Computing,
School of Computer Science and Engineering, Southeast University, Nanjing 211189, China.
Email: yjiang@seu.edu.cn, yfzhouseu@hotmail.com, yunpengli.seu@gmail.com

***Abstract*—In a multiplex network, agents are connected by multiple types of links, and the network can be split into more than one network layer which is composed of the same type of links and involved agents. Traditional task allocation methods of multiagent systems only consider the situations of agents themselves, but neglect the effects of network layers in multiplex networks. To solve such a problem, this paper takes network layers into account and presents a novel network layer-oriented task allocation model for multiplex agent networks; with such a model, first the network layers that can satisfy the objectives of task allocation will be allocated, then the final agents will be selected from the allocated network layers. Moreover, this paper deals with the situation of undependable networks, where the resource access of tasks may be undependable, and implements a task allocation based on negotiation reputation. It shows that the network layer-oriented task allocation model leads to an improvement in the success rate and execution time of tasks in multiplex networks when compared to traditional agent-oriented task allocation methods; moreover, such a model has good scalability for the size of tasks and robustness for dynamic undependability.**

***Keywords*-Multiplex networks; multiagent systems; social networks; task allocation; undependable.**

I. Introduction

Most networked distributed systems such as social networks, grid computing, and P2P networks can be viewed as networked multiagent systems (NMASs) in which agents represent the autonomous nodes and interaction relationships represent interconnections among nodes [1-4]. In a NMAS, resources are placed within the network and can be accessed by agents to execute tasks. Therefore, previous methods of task allocation were always implemented based on the resource accessibilities of agents [5-8], which can be called an *agent-oriented task allocation*. Moreover, most existing studies on this subject are based on the assumption that all the links in the network are of the same type, i.e., it is assumed that the underlying network is simplex.

However, in real-world situations multiplex networks are often seen where there are multiple types of links between agents and each type of link may have a different relative bias in communicating different types of resources [9-11]. Therefore, the communication between two agents in multiplex networks should consider the types of links along their communication path; the accessibility for a resource is determined not only by the communication distance but also by the types of links between the agent and the resource. On the other hand, some agents may not provide dependable resources for other agents in different network layers; some types of links may also be undependable for communicating certain types of resources. Therefore, the problem of undependable resource accessibility [8][12] in multiplex networks should also be addressed.

To solve the above problem in multiplex networks, this paper presents a novel *network layer-oriented task allocation* model, where each network layer is composed of the same type of links and the involved agents. Our model extends previous benchmark methods by introducing the factor of the network layer into task allocation; thus the task is first allocated to the network layers that can satisfy the objectives of task allocation, then the final agents are selected from the allocated network layers. In such model, we deal with the following two issues: 1) the task allocation is implemented based on the network layers' resource accessibilities, where a network layer's resource accessibility is related to both the communication distances and link types from such network layer to the resource; and 2) the reputation and reward mechanisms for both agents and network layers in task allocation are presented to deal with the undependability.

II. Problem Description

A. Task Allocation Objective in Undependable Networked Multiagent Systems

One of the main goals of task allocation is to minimize the task execution time [5][7][13-15]. Task execution in NMASs can be described as the agents' operations when accessing the necessary resources distributed at agents in the networks [5-7]. Therefore, to reduce the execution time of a task, one of the key problems is to reduce the time used accessing the resources necessary for the task [1][8][13][15][16], which is mainly determined by the **communication time** among the allocated agents in the network,$\sum_{\forall a_i,a_j \in A_t} C_{ij}$.($A_t$ is the set of agents allocated to task t; C_{ij} is the communication time between a_i and a_j)

On the other hand, in undependable NMASs, if an allocated agent is undependable and cannot provide the desired resources, it will take more time for the other allocated

agents to seek the missing resources. More seriously, if the task cannot obtain the necessary resources a new task allocation may be implemented, which will waste even more time. Therefore, to reduce task execution time in undependable NMASs, we should **guarantee dependable resource access as well as minimize resource access time** [8], and **the objective of task allocation** is to select the agent set A_t that satisfies the following condition:

$$A_t = argmin_{A_t \in A}((\sum_{\forall a_i, a_j \in A_t} C_{ij})/Dep(A_t)) \quad (1)$$

under the constraint that $R_t \subseteq R_{A_t}$, where R_t is the set of resources required by task t, R_{A_t} is the set of all resources owned by A_t, A is the set of all agents in the NMAS, and $Dep(A_t)$ is the probability that A_t can contribute dependable resources for task t.

B. New Problems Caused by Multiplex Networks

Multiplex networks are often seen in real-world scenarios where there are multiple types of links between agents [9-11]. In multiplex networks, each type of link and the involved agents make up a network layer; each network layer may have a different relative bias and dependability in communicating different type of resources [10]. In summary, the new problems of task allocation caused by multiplex networks can be described as follows:

- The communication time between agents in the multiplex network is influenced by the types of links along their communication paths. Therefore, C_{ij} in Equation (1) and the resource negotiation model between agents should be adjusted for multiplex networks.
- Each network layer has different dependability for communicating different resources, and an agent may have a different dependability when it is used in different network layers. Thus, $Dep(A_t)$ in Equation (1) should consider the effects of network layers.

To solve the above problems, in this paper we adopt the following measures:

- We devise a new algorithm for resource negotiation between agents in multiplex networks. Then, the accessibility of a resource is determined not only by the communication distance but also by the link types. Finally, the task allocation is implemented based on the network layers' resource accessibilities.
- We present new definitions for dependability of agents and network layers, and use negotiation reputation and a reward mechanism to deal with the undependable network layers or agents.

III. MODELING MULTIPLEX NETWORKS

A. Structural Characteristics of Multiplex Networks

Definition 1. Network layer. *In a multiplex network, the links of the same type and the involved agents make up a* ***network layer****. Assuming that the links in the multiplex social network $N = \langle A, E\rangle$ are classified into λ different types $1, \ldots, \lambda$, then N can be split to λ network layers, N_x, $1 \leq x \leq \lambda$.*

$$\begin{aligned} N = \{N_x | N_x = \langle A_x, E_x\rangle \wedge A_x \subseteq A \wedge \\ E_x \subseteq E \wedge (\forall e_{xi}, e_{xj} \in E_x \Rightarrow l_{e_{xi}} = l_{e_{xj}})\} \end{aligned} \quad (2)$$

where e_{xi} and e_{xj} are the links in network layer N_x, $l_{e_{xi}}$ and $l_{e_{xj}}$ denote the types of links e_{xi} and e_{xj}.

Definition 2. Associated network layers and agents. *Given a multiplex network, $N = \langle A, E\rangle$; $\forall a_i \in A$, the associated network layers of a_i with d hops are:*

$$AN_{ai}(d) = \{N_x | N_x \in N \wedge h(a_i, N_x) = d\} \quad (3)$$

where $$h(a_i, N_x) = \min_{\forall a_j \in A_x} h(a_i, a_j) \quad (4)$$

and $h(a_i, a_j)$ denotes the hops between a_i and a_j; the number of hops between two adjacent agents is set to 1. Then, the $n^{th}-$order set of associated agents of a_i is:

$$AA_{a_i}(d) = \{A_x | \forall N_x \in AN_{a_i}(d)\} \quad (5)$$

In NMASs, resources are placed at some agents. A resource at one agent can be communicated with and accessed by other agents. Based on the benchmark work in [10], in this paper **the resource communication relevant to link types** is set as follows:

Let there be m types of resources in the multiplex social network $N = \langle A, E\rangle$; each network layer, $\forall N_x \in N$, is associated with a parameter c_{xk} $(1 \leq k \leq m)$ for k-type resources that measures the relative bias speed N_x has in communicating k-type resources. The higher c_{xk} is, the less communication time cost is needed for the k-type resources in N_x.

B. Resource Negotiation in Multiplex Networks

Let the multiplex network be $N = \langle A, E\rangle = \{\langle A_x, E_x\rangle | 1 \leq x \leq \lambda\}$. Now, we design an algorithm for computing the resource negotiation path in multiplex networks, shown as Algorithm 1. We assume that the communication time of two adjacent agents within the same network layer N_x for k-type resources is $1/c_{xk}$. p_{ij}^k denotes the k-type resource negotiation path between a_i and a_j, C_{ij}^k denotes the communication time for a k-type resource access along p_{ij}^k, $\langle a_i, a_j\rangle_x$ denotes the x-type link between a_i and a_j.

Algorithm 1. Calculating the resource negotiation paths for k-type resources among agents in the multiplex network.

For $(i = 1; i \leq |A|; i++)$
  For $(j = 1; j \leq |A|; j++)$
    $\{b = \infty; x^* = 0;$
    For $(x = 1; x \leq \lambda; x++)$
      {**If** there is a x-type link between a_i and a_j,**then**:
        {**If** $c_{xk} < b$, **then**:$\{b = c_{xk}; x^* = x\};\}\}$
    If $b \not\gamma$ 1 00 1 20 1 1 00 025

C. Undependable Situations in Multiplex Networks

In reality, undependable situations may be encountered due to the openness and heterogeneity of NMASs, where agents may take undependable actions [6][12]; moreover, in multiplex networks, each type of link may have a different relative bias in communicating different types of resources [10]. Thus, the communication of some resources on some network links may also be undependable.

Let there be m types of resources in a multiplex social network; then, the resource status of an agent in such a network can be described as $R_a = \sum_{1 \leq k \leq m} n_k r_k$, which denotes that this agent owns resource r_k $(1 \leq k \leq m)$ in the amount of n_k. Now, we can define the undependable agents as follows:

Definition 3. *An agent is undependable if it satisfies one of the following conditions:*

1) It fabricates its resource status information during task allocation, which can be described as follows. Let there be an agent, a_i; the real resources owned by a_i are $R_{a_i} = \sum_{1 \leq k \leq m} n_k r_k$. When the task allocation heuristic inquiries into the resource status of a_i, and the reported resource status of a_i is $MR_{a_i} = \sum_{1 \leq k \leq m} n'_k r_k$, if $n'_k \neq n_k$, a_i is undependable for its k-type resource status information.

2) It does not contribute all its free resources during task execution if the allocated task requires it to do so, which can be described as follows. Let task t require some resources from a_i, which are denoted as $R^t_{a_i} = \sum_{1 \leq k \leq m} n^t_k r_k$. The set of resources that a_i actually contributes to task t is $R^{t'}_{a_i} = \sum_{1 \leq k \leq m} n^{t'}_k r_k$. Let the real resources owned by a_i be $R_{a_i} = \sum_{1 \leq k \leq m} n_k r_k$. If $n^{t'}_k \neq n^t_k \wedge n^{t'}$

determined by the negotiation reputation of N_x, the resource accessibilities of all agents within N_x, and the distribution of agent localities in N_x:

$$\Re N_x(k) = w\_N_x(k) \cdot \sum_{\forall a_i \in A_x} (\Re a_i(k) \cdot \frac{1}{C^k_{ix^*}}) \quad (8)$$

where $C^k_{ix^*}$ denotes the time cost for negotiating k-type resources between agent a_i and a_{x^*}, and a_{x^*} is the center of network layer N_x for k-type resources:

$$a_{x^*} = argmin_{\forall a_i \in A_x} (\sum_{\forall a_j \in (A_x - \{a_i\})} C^k_{ij}) \quad (9)$$

B. Task Allocation Mechanism

Traditional task allocation models are often based on a manager/contractor architecture [6], where the manager is fixed during the allocation process of a task, which is called the *fixed manager manner*. Now, we substantially extend the manager/contractor architectures in [7][8] by presenting a new task allocation architecture where the manager is not fixed during the task allocation process, which is called the *alterable manager manner*; moreover, the managers and contractors are all network layers in our network layer-oriented model. Now we can explain it briefly as follows. At first, a manager is selected for a task by using a centralized heuristic; then such a manager will seek another network layer to act as a contractor to obtain the highest resource accessibility that can satisfy the resource requirements of a task. Next, those two network layers become the already allocated ones. Then, *each network layer in the group of already allocated network layers will act as a manager to seek the next contractor* to obtain the highest resource accessibility that will satisfy the resource requirements of task. Finally, the optimal contractor can be allocated, and the new already allocated network layers will seek the following contractor again. Such a process will repeat until all resources required by the task can be satisfied or all network layers are allocated. Therefore, the alterable manager manner outperforms the traditional fixed manager manner by using the results of the previously allocated network layers when seeking an optimal result.

Definition 8. Distance between two network layers. *Let N be a multiplex network, $N = \langle A, E \rangle$; $N_x, N_y \in N$. The negotiation distance between N_x and N_y for k-type resources is defined as:*

$$D^k_{xy} = \min_{a_i = a_{x^*} \wedge a_j = a_{y^*}} C^k_{ij} \quad (10)$$

where C^k_{ij} is calculated according to Algorithm 1.

Let R_t be the set of resources required by task t, $\overline{R_t}$ be the set of resources for task t that are currently unavailable, and R_{N_x} be the set of resources that are owned by network layer N_x. Then, the resources that N_x may contribute to task t are $\overline{R_t} \cap R_{N_x}$; $\lambda_k(\overline{R_t} \cap R_{N_x})$ denotes the number of k-type resources in $\overline{R_t} \cap R_{N_x}$. Let the manager network layer be N_x and let task t need m_t types of resources; now we can make N_x negotiate with another network layer (e.g., N_y) according to the following negotiation value:

$$VN_y(t) = \sum_{1 \le k \le m_t} (\Re N_y(k) \cdot \lambda_k(\overline{R_t} \cap R_{N_x}) / D^k_{xy}) \quad (11)$$

The manager selects the contractors according to their negotiation values, arranged in descending order.

Theorem 1. *It is assumed that task is t and the negotiation values are correct. Let the manager be N_x and the two contractor candidates be N_y and N_z; $S(N)$ denotes the degree to which the task allocation objective can be satisfied by N. Therefore: $VN_y(t) > VN_z(t) \Rightarrow S(\{N_x\} \cup \{N_y\}) > S(\{N_x\} \cup \{N_z\})$.*

Proof. *According to Equation (1), $S(N)$ is determined by two factors: 1) the negotiation distance between the network layers in N; and 2) the dependability with which N can contribute real resources. In Equation (11), the first part $\Re N_y(k)$ includes the negotiation reputation of a network layer that can measure the dependability of the contractor candidate; the second part $\lambda_k(\overline{R_t} \cap R_{N_x}) / D^k_{xy}$ is inversely proportional to the negotiation distance between the allocated network layers. Therefore, $VN_y(t) > VN_z(t)$ denotes that the value of the negotiation reputation divided by total communication time costs of $\{N_x\} \cup \{N_y\}$ is higher than the one of $\{N_x\} \cup \{N_z\}$, thus we have $S(\{N_x\} \cup \{N_y\}) > S(\{N_x\} \cup \{N_z\})$.* □

Therefore, from Theorem 1, the negotiation value in Equation (11) can be used to satisfy the task allocation objectives in Equation (1). Now, we use the alterable manager manner to implement a network layer-oriented task allocation, as shown as Algorithm 2.

Algorithm 2. Network layer-oriented task allocation.

1) $N_* = \arg\max_{\forall N_x \in N} (\Re N_x(k))$
*/*k-type resources are the ones that task t needs mostly*/*
2) $\overline{R_t} = R_t - R_{N_*}$;$N' = N - \{N_*\}$;$N_t = \{N_*\}$;
   $b1 = 0$; $b2 = 0$; $n = 0$;
3) **If** $\overline{R_t} == \emptyset$, **then**: $\{b1 = 1\}$;
4) **While** (($b1 == 0$ and $b2 == 0$)) **do**:
   4.1) $max = 0$; $b2 = 1$;
   4.2) $\forall N_y \in N'$:
      4.2.1) $max_{temp} = 0$;
      4.2.2) $\forall N_x \in N_t$;
         4.2.2.1) Calculating $VN_y(t)$ according to (11);
         4.2.2.2) **If** $VN_y(t) > max_{temp}$, **then**:
            $\{max_{temp} = VN_y(t)\}$;
            */*Now N_x is the manager*/*
      4.2.3) **If** $max_{temp} > max$, **then**:
         $\{max = \max_{temp}; b2 = 0; N_{temp} = N_y;\}$
   4.3) **If** $(b2 = 0)$, **then**:
      $\{N_t = N_t \cup \{N_{temp}\}$; $\overline{R_t} = \overline{R_t} - R_{Ntemp}$;
      $N' = N' - \{N_{temp}\}$; $n++$; $N_{tn} = N_{temp}$;$\}$
   4.4) **If** $\overline{R_t} == \emptyset$, **then**: $\{b1 = 1\}$;
5) **If** ($b1 == 1$), **then Return** (N_t);
   else Return $(False)$;
6) **End**.

Let there be a set of network layers N. The resource accessibility of N for k-type resources can be defined as $\Re N(k) = \max_{\forall N_x \in N} (\Re N_x(k))$. Now, if the task is t; the

set of network layers allocated by using a network layer-oriented allocation model with alterable manager manner is N_t, $N_t \subseteq N$; and the set of network layers allocated by using a network layer-oriented allocation model with fixed manager manner is N'_t, $N'_t \subseteq N$. Then, we have:

Theorem 2. *Let the multiplex social network be $N = \langle A, E\rangle$, $N = \{N_x | 1 \leq x \leq \lambda\}$, where λ is the number of network layers. If a task t needs k-type resources, we have:$\Re N_t(k) \geq \Re N'_t(k)$.*

Proof sketch. *With the fixed manager manner, in each allocation step only the resource accessibility of the contractor and the communication time cost between the manager and contractor can be optimized. Now, with the alterable manager manner, each agent in the already allocated network layers will act as a manager to seek the contractor with the highest resource accessibility for k-type resources from the viewpoint of the manager, and finally the contractor candidate with the highest negotiation value of all the managers will be selected; thus, the resource accessibility of the contractor and the minimum communication time cost between the contractor and already allocated network layers are optimized. Therefore, we have $\Re N_t(k) \geq \Re N'_t(k)$.* □

Therefore, Theorem 2 proves that our alterable manager manner outperforms the previous fixed manager manner by improving the resource accessibility of allocated network layers.

Theorem 3. *Let the set of allocated network layers using Algorithm 2 be N_t and the first manager be N_*. It is then assumed that there is another set of network layers, N', that includes N_* and can also satisfy all the resources in R_t. Then, we have:*

$$(\forall N' \wedge (N_* \in N') \wedge (N' \subseteq N) \wedge (R_t \subseteq R_{N'})) \Rightarrow \sum_{\forall N_y \in (N_t - \{N_*\})} VN_y(t) \geq \sum_{\forall N_y \in (N' - \{N_*\})} VN_y(t)$$

Proof sketch. *Now we can use reductio ad absurdum to prove Theorem 3. Assume there is a set of network layers N', $N_* \in N' \wedge N' \neq N_t$, that can provide all the required resources for executing task t, and the total negotiation values of $N' - \{N_*\}$ by using the alterable manager manner is $\sum_{\forall N_y \in (N' - \{N_*\})} VN_y(t)$; if $\sum_{\forall N_y \in (N_t - \{N_*\})} VN_y(t) < \sum_{\forall N_y \in (N' - \{N_*\})} VN_y(t)$. There are network layers with lower negotiation values that can provide the required resources in $\overline{R_t}$ and be selected by the already allocated network layers in Algorithm 2, but the higher negotiation-value network layers with the required resources in $\overline{R_t}$ are not selected by the existing network layers. In Algorithm 2, the selection is implemented by Step 4.2 and 4.2.2, which guarantees that the selected network layer in each Step 4.2 has the maximum negotiation value from the currently allocated network layers. Therefore, a situation in which $\sum_{\forall N_y \in (N_t - \{N_*\})} VN_y(t) < \sum_{\forall N_y \in (N' - \{N_*\})} VN_y(t)$ cannot take place in Algorithm 2. To address this issue, we have Theorem 3.* □

From Theorem 3, Algorithm 2 can find the network layers with the maximum negotiation values, thus satisfying the objectives of task allocation in Equation (1) according to Theorem 1.

After the network layers are allocated using Algorithm 2, then the real agents within the allocated network layers will be selected. The selection process is shown as Algorithm 3.

Algorithm 3. Selecting final agents from allocated network layers. */*$N_t = \{N_{tx} | N_{tx} = \langle A_{tx}, E_{tx}\rangle, 1 \leq x \leq n\}$, are the allocated network layers resulted from Algorithm 2 */*

1) $x = 1; b = 0; \overline{R_t} = R_t$;
2) **While**(($x \leq n$) and $b == 0$) **do**:
   2.1) **Set** the tags for all agents in A_{tx} to 0 initially;
   2.2) $a_{x^*} = \arg\max_{\forall a_i \in A_{tx}}(\Re a_i(k))$;
   2.3) **Create** Queue (Q_x); **Insert** Queue (Q_x, a_{x^*});
        Set the tag of a_{x^*} to 1;
   2.4) $A_t = \{a_{x^*}\}$;$\overline{R_t} = \overline{R_t} - R_{ax^*}$;
   2.5) **If** $\overline{R_t} == \emptyset$, **then**: $\{b = 1\}$;
   2.6) **While** ((!EmptyQueue(Q_x))and ($b = 0$)) **do**:
      2.6.1) $aout =$**Out** Queue(Q_x);$R' = \overline{R_t} - R_{aout}$;
      2.6.2) **If** $R' \neq \overline{R_t}$, **then**:$\{\overline{R_t} = \overline{R_t} - R_{aout}$;
             $A_t = A_t \cup \{aout\}\}$;
      2.6.3) **If** $\overline{R_t} == \emptyset$, **then**:$\{b = 1\}$;
      2.6.4) $\forall alocal \in L_{aout}$:
             If the tag of $alocal$ is 0, **then**:
             {**Insert** Queue($Q_x, alocal$);
              Set the tag of $alocal$ to 1};
   /L_{aout} is the set of neighbors of aout in N_{tx}*/*
   2.7) $x++$;
3) **Return** (A_t);
4) **End**.

Theorem 4. *Let the set of allocated agents in N_{tx} using Algorithm 3 be A^*_{tx} and the initiator agent be a_{x^*}; the set of resources lacking from a_{x^*} to implement t is $\overline{R^t_{a_{x^*}}}$. It is then assumed that there is another set of agents in N_{tx}, A'_{tx}, that includes a_{x^*} and can also satisfy the resource requirements of t in N_{tx}; $Com_x(a_i, a_j)$ denotes the communication time between a_i and a_j within N_{tx}. Then, we have:*

$$(\forall A'_{tx} \wedge (A'_{tx} \subseteq A_{tx}) \wedge (\overline{R^t_{a_{x^*}}} \cap R_{A'_{tx}} = \overline{R^t_{a_{x^*}}} \cap R_{A^*_{tx}})) \Rightarrow \sum_{\forall a_i \in (A'_{tx} - \{a_{x^*}\})} Com_x(a_{x^*}, a_i) \geq \sum_{\forall a_i \in (A^*_{tx} - \{a_{x^*}\})} Com_x(a_{x^*}, a_i)$$

Proof. *If Algorithm 3 is used, the set of allocated agents in an allocated network layer N_{tx} is A^*_{tx}, and the total communication costs between a_{x^*} and other agents in $A^*_{tx} - \{a_{x^*}\}$ within N_{tx} are $\sum_{\forall a_i \in (A^*_{tx} - \{a_{x^*}\})} Com_x(a_{x^*}, a_i)$. Now, if there is a set of agents A'_{tx}, $A'_{tx} \neq A^*_{tx}$, which can provide the same set of resources to t as A^*_{tx}, and the total communication cost between a_{x^*} and other agents in $A'_{tx} - \{a_{x^*}\}$ is $\sum_{\forall a_i \in (A'_{tx} - \{a_{x^*}\})} Com_x(a_{x^*}, a_i)$; if $\sum_{\forall a_i \in (A'_{tx} - \{a_{x^*}\})} Com_x(a_{x^*}, a_i) < \sum_{\forall a_i \in (A^*_{tx} - \{a_{x^*}\})} Com_x(a_{x^*}, a_i)$, it shows that there are any agents with a longer communication distance that provide the resources*

in $\overline{R^t_{a_{x^*}}}$, but the nearer agents have but do not provide the resources in $\overline{R^t_{a_{x^*}}}$. Obviously, such a situation cannot take place in Algorithm 3 where a_{x^*} negotiates with other agents from near to far within the network layer as Step 2.6. Therefore, we have Theorem 4. □

Therefore, Algorithm 3 can obtain an initiator agent with the highest resource accessibility and the minimum communication time cost between such initiator agent and other allocated agents within the same network layer.

C. Reward Mechanism

To encourage network layers and agents in multiplex networks to provide dependable resources, now we present the reward mechanism. Each task is associated with a value; if the task can be executed successfully, the allocated network layers and agents will be rewarded with this value, or else they will be punished with such value. Let the associated value for task t be τ_t $(0 \leq \tau_t \leq 1)$, N_t be the network layers allocated to t, and A_t be the set of agents that are really allocated to t. Then, τ_t will be divided into three parts: 1) a reward to the allocated network layers (α); 2) a reward to the allocated agents within the allocated network layers (β); 3) a reward to the involved agents that are out of allocated network layers but provide communication relay services for the resource negotiations between allocated network layers (γ). We can set:$\alpha + \beta + \gamma = 1$;$\gamma \ll \alpha + \beta$. The allocated network layers and agents will be rewarded (or punished) according to their real resource contributions.

Then, the reward mechanism is shown as follows when the task is executed successfully:

$$\begin{aligned} &\forall r_k \in R_t : \forall N_x \in N_t : \\ &w\_N_x(k) = w\_N_x(k) \cdot (1 + \alpha \cdot \tau_t \cdot (|R^t_{N_x}(k)|/|R_t|)) \end{aligned} \tag{12}$$

where $R^t_{N_x}(k)$ is the set of k-type resources that N_x really contributes in the execution of task t:

$$\begin{aligned} &\forall r_k \in R_t : \forall a_i \in A_t : \\ &w\_a_i(k) = w\_a_i(k) \cdot (1 + \beta \cdot \tau_t \cdot (|R^t_{a_i}(k)|/|R_t|)) \end{aligned} \tag{13}$$

where $R^t_{a_i}(k)$ is the set of k-type resources that a_i really contributes to the execution of task t:

$$\begin{aligned} &\forall r_k \in R_t : (\forall a_m \notin \cup_{N_x \in N_t} A_x) \wedge (\exists a_i, a_j \in A_t \wedge a_m in P^k_{ij}) : \\ &w\_a_m(k) = w\_a_m(k) \cdot (1 + \gamma \cdot \tau_t) \end{aligned} \tag{14}$$

Accordingly, the penalty mechanism if the task is executed unsuccessfully is shown as follows:

$$\begin{aligned} &\forall r_k \in R_t : \forall N_x \in N_t : \\ &w\_N_x(k) = w\_N_x(k) \cdot (1 - \alpha \cdot \tau_t \cdot (1 - |R^t_{N_x}(k)|/|R_t|)) \end{aligned} \tag{15}$$

$$\begin{aligned} &\forall r_k \in R_t : \forall a_i \in A_t : \\ &w\_a_i(k) = w\_a_i(k) \cdot (1 - \beta \cdot \tau_t \cdot (1 - |R^t_{a_i}(k)|/|R_t|)) \end{aligned} \tag{16}$$

$$\begin{aligned} &\forall r_k \in R_t : (\forall a_m \notin \cup_{N_x \in N_t} A_x) \wedge (\exists a_i, a_j \in A_t \wedge a_m in P^k_{ij}) : \\ &w\_a_m(k) = w\_a_m(k) \cdot (1 - \gamma \cdot \tau_t) \end{aligned} \tag{17}$$

V. Experimental Validation and Analyses

A. Experimental Settings

To validate the effectiveness of the presented network layer-oriented task allocation model for the undependable multiplex networks, we use the following indexes:

- *Success rate* (sr). The success rate demonstrates the dependability of task allocation:

$$sr = (\sum_{i=1}^{n}(1/\xi_{t_i}))/n \tag{18}$$

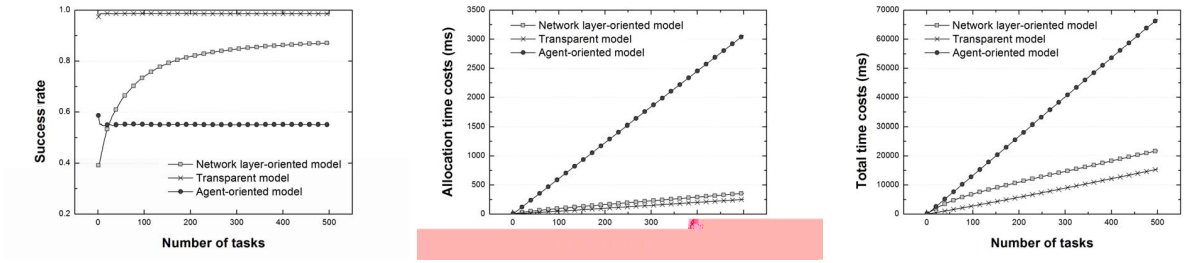
where n is the number of total tasks; ξ_{t_i} indicates that the first $(\xi_{t_i} - 1)$ attempts to allocate task t_i are unsuccessful and only the $\xi_{t_i}{}^{\text{th}}$ allocation of t_i is successful.

- *Time costs*. the total time costs (T), which are the sum of the allocation and execution time costs of all tasks; the allocation time costs of all tasks (T_r); and the average time cost for each task (T_a), which is calculated by dividing T by n. Among these time costs, T_r is a part of T; we consider T_r separately to test the complexity of our proposed network layer-oriented allocation model by comparison to the previous models.

We compare our model with the following two approaches, which act as the benchmarks:

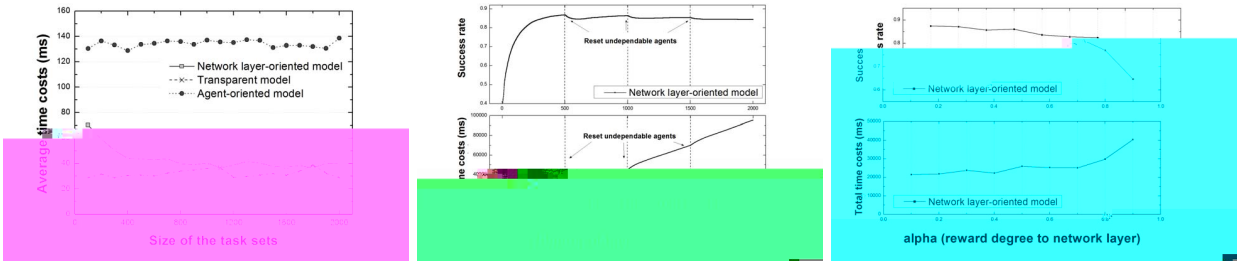
- *Agent-oriented task allocation model*: the task will be allocated to the agents with larger amount of the resources required by the task [1][5-7][13][15]. This model ignores the network layers and does not consider the biases of network layers for communicating different types of resources.
- *Transparent model in which all undependable agents can be detected*: the model can detect all deceptive agents in the network, and it will select the truthful agents to negotiate through the path with the lowest communication cost [8]. This model is not practical in real-world scenarios, but it can be used as a benchmark to evaluate the task allocation performance in undependable situation.

Each experiment comprises 200 runs to obtain the average results. The initial social network is constructed by a random network model, in which 100 agents are included. The connection probability for any two randomly selected agents is set to 0.05. Then, the initial social network is



(a) Success rate of task allocation (b) Allocation time costs of tasks (c) Total time costs of tasks

Figure 1: The comparison of task allocation performances among network layer-oriented model, agent-oriented model, and transparent model



(a) Scalability for size of tasks (b) Robustness for dynamic undependability (c) Effect of reward mechanism

Figure 2: Tests of our network layer-oriented model

achieves nearly 90% when the task number is 500. This shows that our model is good at evolving, i.e., our model can learn and fit the environment gradually in response to the task allocation and execution. Both the successful and unsuccessful allocations of tasks are helpful for the evolution of our model toward a more dependable status. From Fig. 1(a), our model can achieve much better performance based on success rate than the agent-oriented model, and is very close to the transparent model when the number of tasks is large; the main reason is that the reputation and reward mechanisms in our model consider the undependability of both agents and network layers, and thus can effectively keep the dependability of task allocation. It notes that the transparent model can detect all the undependable agents, but its success rate cannot reach 100%; this phenomenon is caused by the network loss rate during resource communication, which may lead to unsuccessful resource access.

Fig. 1(b) shows the results of allocation time costs of the tasks. Similar to Fig. 1(a), our model outperforms the agent-oriented model significantly and performs a little worse than the transparent model. The main reason is the difference in the complexities of the allocation algorithms between our model and the other models. The candidate agents in the agent-oriented allocation model are all the agents embedded in the network, while there are less candidates in our model because the task allocation method is network layer oriented, and in each network layer the number of agents is much smaller. In general, this is an important advantage of network layer-oriented task allocation in multiplex networks. The transparent model is also network layer oriented, and it does not need to check the agents' dependability, thus it has the best performance for reducing allocation time costs.

Fig. 1(c) shows the results of the total time costs for all tasks, where the conclusion similar to Fig. 1(a) and (b) can be made. There are three primary factors accounting for such a result: 1) the allocation algorithm complexity: our model can reduce the allocation complexity much than traditional agent-oriented model; 2) the speed of resource access: the task allocation using our model will select the most suitable network layer to allocate tasks so that the communication cost can be reduced; 3) the dependability of task allocation: the agent-oriented model has low success rate of task allocation (shown as Fig. 1(a)), the total time costs of task allocation will increase manyfold. In contrast, our model can achieve high dependability; thus the total time can remain close to that of the transparent model.

2) Tests on the properties of our model:

From Fig. 1, the effectiveness of our network layer-oriented model has been comprehensively validated by comparison with other two benchmark models. Now, we explore several key properties of our network layer-oriented model: the scalability for the sizes of tasks; the robustness in dynamic undependable situations; and the effect of the reward mechanism in task allocation.

In Fig. 2 (a), we test our model's scalability for the size of task sets. Fig. 2(a) shows the average time of each task when the size of the task sets ranges from 100 to 2000. When the size of the task sets increases, the average time for each task using our network layer-oriented allocation model decreases rapidly and finally tends toward stability (approaching the transparent model). In other words, our model can perform better if there are more tasks arriving in

the system; a potential explanation is that the reputation and reward mechanism has more chances to evolve and thus can achieve higher dependability with a larger task set. Actually, Fig. 1(a) can also be used to make a similar conclusion.

Then, we test our model's robustness in dynamic undependable situations in Fig. 2 (b). The dynamic undependable situations in experiments can be set as follows: the undependable agents in the network will be reset dynamically when the number of tasks allocated is 500, 1000 and 1500; in other words, after each reset, the previously undependable agents may become normal, while some normal agents may become undependable. This dynamics brings a challenge for the reputation and reward mechanism of our model. Fig. 2(b) shows the performance of our model in such a dynamic environment. When the undependable agents are reset dynamically, the success rate of task allocation decreases, obviously, but after the allocation of some tasks, the success rate can be recovered. On the other hand, the total time costs increase much more rapidly for the first few tasks after the undependable agents are reset, but the total rate of increase can still converge to the former state. Such a phenomenon reveals that our model has good robustness in the dynamic environments where agents may change their identities dynamically.

Finally, we test the effect of the reward mechanism in task allocation, shown as Fig. 2(c). We vary the parameter α in Equation (12) which determines the degree of reward to the allocated network layer. From Fig. 2(c), we can find if $\alpha < 0.5$, the performance of our model varies slightly and can keep a good performance. However, if $\alpha \geq 0.5$, and especially when $\alpha \geq 0.8$, the performance of our model decreases. The potential reason is that if the reward to the network layer is too large, the reputations of network layers will not vary sufficiently after the allocation of some tasks; namely, the effect of reputation mechanism will become weak. In conclusion, the degree of reward to the network layer in the reward mechanism of our model should be set to an appropriate value so that dependability can be achieved.

VI. Conclusions and Future Work

To consider the effects of network layers in multiplex networks, this paper presents a novel network-layer oriented task allocation model which is implemented based on the network layers' resource accessibilities; with such model, at first the network layers that can satisfy the objectives of task allocation should be allocated, then the final agents will be selected from the allocated network layers. As shown by the theoretical analyses, the presented model can satisfy the task allocation objective in undependable multiplex networked multiagent systems. Moreover, the experiments show the following: the network layer-oriented task allocation model leads to an improvement in the success rate and execution time of tasks in multiplex networks by comparison with the traditional agent-oriented task allocation model and often performs close to the transparent model, and such model also has good scalability for size of tasks and robustness for dynamic undependability.

In future work, we will explore ways to adapt the model to dynamic multiplex networks. Moreover, we will implement load balancing among network layers and agents in the task allocation.

Acknowledgment

This work was supported by the NSFC (No.61170164), the Program for Distinguished Talents of Six Domains in Jiangsu Province (No.2011-DZ023), and the Funds for Distinguished Young Scholars of Jiangsu Province (BK2012020).

References

[1] Y.Jiang, Z.Huang, "The Rich Get Richer: Preferential Attachment in the Task Allocation of Cooperative Networked Multiagent Systems with Resource Caching," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 42(5): 1040-1052, 2012.

[2] D.Ye, M.Zhang, D.Sutanto, "Self-Adaptation Based Dynamic Coalition Formation in A Distributed Agent Network: A Mechanism and A Brief Survey," *IEEE Transactions on Parallel and Distributed Systems*, 24(5): 1042-1051, 2013.

[3] S. Abdallah,V. Lesser, "Multiagent Reinforcement Learning and Self-Organization in A Network of Agents," *Proceedings of the 6th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-07)*, pp.172-179, Honolulu, Hawaii, May 14-18, 2007.

[4] Y.Jiang, J.Hu, D.Lin, "Decision Making of Networked Multiagent Systems for Interaction Structures," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 41(6): 1107-1121, 2011.

[5] B. An, V. Lesser, K. M. Sim, "Strategic Agents for Multi-Resource Negotiation," *Autonomous Agents and Multi-Agent Systems*, 23(1): 114-153, 2011.

[6] M. M. Weerdt, Y. Zhang, T. Klos, "Multiagent Task Allocation in Social Networks," *Autonomous Agents and Multi-Agent Systems*, 25(1): 46-86, 2012.

[7] Y.Jiang, J.Jiang, "Contextual Resource Negotiation-Based Task Allocation and Load Balancing in Complex Software Systems," *IEEE Transactions on Parallel and Distributed Systems*, 20(5): 641-653, 2009.

[8] Y. Jiang, Y. Zhou, W. Wang, "Task Allocation for Undependable Multiagent Systems in Social Networks," *IEEE Transactions on Parallel and Distributed Systems*, 24(8): 1671-1681, 2013.

[9] J.Gómez-Gardeñes, I.Reinares, A.Arenas, L.M.Floría, "Evolution of Cooperation in Multiplex Networks," *Scientific Reports*, 2: 620, 2012.

[10] O.Yağan, V.Gligor, "Analysis of Complex Contagions in Random Multiplex Networks," *Physical Review E*, Vol.86: 036103, 2012.

[11] C.D.Brummitt, K.M.Lee, K.I.Goh, "Multiplexity-Facilitated Cascades in Networks," *Physical Review E*, Vol. 85: 045102(R), 2012.

[12] H.Ohtsuki, C.Hauert, E.Lieberman, M.A.Nowak, "A Simple Rule for The Evolution of Cooperation on Graphs and Social Networks," *Nature*, vol.441, pp.502-505, 25 May 2006.

[13] J.Liu, X.Jin, Y.Wang, "Agent-based Load Balancing on Homogeneous Minigrids: Macroscopic Modeling and Characterization," *IEEE Transactions on Parallel and Distributed Systems*, 16(7): 586-598, 2005.

[14] O.Shehory, S.Kraus, "Methods for Task Allocation via Agent Coalition Formation," *Artificial Intelligence*, 101(1-2), pp.165-200, 1998.

[15] J.Xu, Albert Y.S.Lam, Victor O.K.Li, "Chemical Reaction Optimization for Task Scheduling in Grid Computing," *IEEE Transactions on Parallel and Distributed Systems*, 22(10): 1624-1631, 2011.

[16] B. Hong; V.K. Prasanna, "Adaptive Allocation of Independent Tasks to Maximize Throughput," *IEEE Transactions on Parallel and Distributed Systems*, 18(10): 1420-1435, 2007.