A Security-aware Load Balancing Algorithm for Structured P2P Systems Based on Ant Colony Optimization

1Wei Mi, 2Chunhong Zhang, 3Xiaofeng Qiu
Beijing University of Posts and Telecommunications, Beijing 100876, China,
{miwei1985, zhangch.bupt.001, qiuxiaofeng}@gmail.com

Abstract

In DHT-based P2P systems, the use of DHT and the heterogeneity of node capacities can cause a load imbalance problem, which may affect the DHT system efficiency. Many solutions have been proposed to tackle the load balancing issue. However, the convergence rate is still low, and they reassign loads among nodes without considering security factor. This paper proposes a security-aware load balancing algorithm based on Ant colony optimization, and designs a candidate nodes discovery mechanism and a target nodes selection mechanism. Two types of agents are cooperating to realize these mechanisms: Search ant and Guide ant. Performance evaluation shows that, compared to the classical algorithms, the load balancing effect of the proposed algorithm is insignificant, while the convergence rate and load balancing security are significantly raised; In addition, this paper analysis how system and algorithm parameters affect the performance of proposed algorithm in depth.

Keywords: P2P, DHT, Load balancing, Ant colony optimization, security-aware

1. Introduction

DHT-based P2P has now become the mainstream application of P2P technology. It is greatly improved in extensibility, fault tolerance, and resources seeking speed. Nowadays there are many research branches on the DHT. Load balancing is one of their research focuses. Excellent balance algorithm can significantly improve the stability and carrying capacity of DHT system.

At present, many solutions have been proposed to tackle the load imbalance [1]–[9]. However, these solutions mainly have the following three deficiencies: 1) the convergence rate is still low. They need more detect messages and higher detect overhead; 2) there are too many harsh assumptions. Some rely on some fixed nodes for collecting load information and generating reassign policy. It requires additional equipments and higher cost, resulting in single point failure; 3) reassign loads among nodes without considering safety factor, resulting in the transfer contents insecurity.

This paper proposes a security-aware load balance algorithm for DHT systems. Based on Ant colony optimization, it designs a candidate nodes discovery mechanism and a target nodes selection mechanism. Such advantages of algorithm are as following:

1) Algorithm relies on all nodes in the system, so it can avoid a single point failure.
2) It proposes an ant-based candidate nodes discovery mechanism. Specifically, volatile pheromones make Search ant sense the change of nodes’ available capacity; Safety factor is also introduced in routing selection as a heuristic factor, which guaranteeing the load transfer' security; The life-span control policy of Search ant makes query overhead controllable.
3) In DHT routing selection, according to the restrictions of objective function, the path selection is determined by pheromones and safety factor. This is a non-direct cooperation way, it can avoid the blind search and improve system convergence rate, which is greatly improved in extensibility.
4) It proposes a target node selection mechanism. It uses a candidate node list that consists of pheromone and safety factor to instruct source node selecting target nodes. It makes a compromise between load balancing effect and security, which improves the effectiveness of the algorithm.
5) Pheromone update is complete with the DHT nodes’ routing table update process. It does not need any additional structures to collect load information and can solve the slow convergence problem, which is caused by no significant differences of pheromone in the initial ant colony optimization.

The rest of this paper is structured as follows. The related work is presented in Section II. Section III describes the security-aware load balancing algorithm. We evaluate performance of the proposed load balancing algorithm in Section IV. Finally, we conclude and present future work in Section V.
2. Related work

In the research of optimizing the load balancing performance for DHT system, ID manipulating solution\(^\text{[7-9]}\) and virtual server solution\(^\text{[1-6,14,15]}\) are now widely used. In order to achieve load balancing, these solutions adjust the size of key space or the number of documents for node through transferring virtual servers or multi-hash, but the emphasis on the transfer node discovery research is different.

Both CAN\(^\text{[1]}\) and Chord\(^\text{[2]}\) had achieved load balancing, and assumed that capacity of all nodes is equal. CFS\(^\text{[3]}\) simplifies load transfer by remove the virtual server, which may lead to the other nodes overloaded and convergence time longer.

Reference\(^\text{[4]}\) describes three load balancing algorithms: one-to-one, one-to-many, many-to-many, based on a virtual server and directory, which are used in static heterogeneous network and are expanded in Reference\(^\text{[5]}\) to the dynamics heterogeneous networks. The assignment of virtual nodes is typically performed by one or more directory nodes, resulting in single-point failure. Reference\(^\text{[6]}\) builds a structure on top of the P2P: k-ary tree, which is responsible for the collection and the release of node information, as well as the transfer strategy of virtual server. The algorithm makes the network structure more complicated.

Reference\(^\text{[7]}\) has proposed the use of the “power of two choices” paradigm to achieve better load balancing. Each object is hashed to \(d \geq 2\) different IDS, and is placed in the least loaded node of the nodes. Reference\(^\text{[9]}\) proposes algorithm for ID space balancing. They assign multiple positions of the ID space to every peer, but choose only one of those virtual nodes to become active at a time. However, this algorithm needs to frequently adjust the node ID, causing higher load regulation overhead.

In this paper, the proposed algorithm is based on Ant colony optimization (ACO). ACO is a novel evolutionary algorithm, which has the characteristics such as positive feedback, distributing computing and the use of a constructive heuristic etc. And these features quite match the demands of network optimization, some ant-based algorithms have been proposed for many application\(^\text{[10-13]}\), such as mobile ad hoc networks routing, topology optimization, distributed QoS routing, resource allocation, etc.

3. A Security-aware Load Balancing Algorithm for DHT Systems

This paper adopts ant colony optimization to design the candidate nodes discovery mechanism and the target nodes selection mechanism. Specially, two types of agents cooperate to realize these mechanisms: Search ant and Guide ant. Search ant simulates the food-seeking behavior of ants that searches for light-nodes on the network. Guide ant is responsible for managing a candidate light-load node list, instructing source node to select target nodes. In this section, some related definition, the data structure of the search and guide ants, the candidate light-load nodes discovery mechanism, the target node selection mechanism and the overall process of load balancing algorithm will be presented.

3.1. Related Definitions

**Definition 1** Node Capacity \((C)\)

The factors that affect the load capacity of the node, such as CPU speed, storage capacity, delay, bandwidth, and so on, are unified as one resource.

\[
C = P(CPU) \cdot w_1 + P(Space) \cdot w_2 + P(Memory) \cdot w_3 + P(IO) \cdot w_4 + P(BandWidth) \cdot w_5
\]

(1)

Where \(w_i\) represents the weight of each factor; \(P(x)\) represents the value of factor \(x\)’s capacity. Different application systems have different weight of each factor.

**Definition 2** Node Load \((L)\)

In DHT network, the load of nodes is the demand for capacity.

\[
L = L(CPU) \cdot w_1 + L(Space) \cdot w_2 + L(Memory) \cdot w_3 + L(IO) \cdot w_4 + L(BandWidth) \cdot w_5
\]

(2)

Where \(w_i\) represents the weight of each factor; \(L(x)\) represents the value of factor \(x\)’s load.

**Definition 3** Utilization Rate \((\mu / \bar{\mu})\)

Node utilization rate \(\mu\) refers to the ratio of load of node \(L\) to its largest carrying capacity \(C\), that is, \(\mu = L/C\). Node utilization rate describes the situation of the current node load.

Adjust load capacity utilization trigger threshold \(LT\): when \(\mu > LT\), node will need to transfer part of their load out, and the overhead of adjust algorithm can be afforded.
System utilization rate refers to the ratio of all the nodes’ load in the system to the largest carrying capacity, which represents the total load situation of network. It can be described as

$$
\pi = \frac{\sum_{i=1}^{N} L(i)}{\sum_{i=1}^{N} C(i)}
$$

(3)

**Definition 4** Load Deviation Rate ($e$)

It can be described as the difference between utilization rate of node and system utilization rate.

$$
e = \mu - \bar{\mu}
$$

(4)

**Definition 5** Node Safety Factor ($S$)

In this paper, service reputation is the safety factor of node. It reflects the accuracy of node’s service description information and the credibility of node’s provided service. For comparison, the value of service reputation is between 0 and 1. How to calculate service reputation value is out of scope.

3.2. Data Structure of the Search and Guide ants

Based on the different tasks performed by the ants, there are two types: Search ant and Guide ant. Search ant simulates the food-seeking behavior of an ant that searches for light-nodes. Guide ant is responsible for managing a candidate light-load node list, instructing source node to select target nodes.

Search ant is generated by the source node whose $\mu > LT$, that find candidate light-load nodes in the DHT. Each search ant has a tabu list to record the visited nodes which prevent repeat visit nodes and a maximal survival time $TTL$. Guide ant generated by candidate light-load node or node whose search ant’s $TTL$ is 0. Guide ant is responsible for managing a candidate light-load node list that consists of pheromone and safety factor.

According to the function of ants, we build data structures for them as shown in Table 1, 2.

<table>
<thead>
<tr>
<th>Table 1. Structure table of search ant</th>
<th>Table 2. Structure table of guide ant</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Ant Id</strong></td>
<td><strong>Restricted condition</strong></td>
</tr>
<tr>
<td>AntID</td>
<td>res(m) s.t</td>
</tr>
<tr>
<td><strong>Ant Id</strong></td>
<td><strong>Source Node</strong></td>
</tr>
<tr>
<td>AntID</td>
<td>FN</td>
</tr>
</tbody>
</table>

3.3. Candidate Light-load Nodes Discovery Mechanism

In DHT systems, the specific circumstance of node’s load is unknown, and nodes are heterogeneity. Adopting ant colony optimization algorithm, we can find candidate light-load nodes in the unknown load distribution system, and choose the suitable one or more nodes to data migration. In this section, we will elaborate the candidate node discovery mechanism from the following aspects: the generation and update of the pheromone, the routing mechanism and life-span control policy of Search ant.

3.3.1. Generation and Updating of Pheromone

Pheromone plays an important role in the candidate node discovery mechanism. It guides the route direction of search ant. Therefore, pheromone must fully reflect the node load. This paper defines the pheromone $ph(i)$ as the available capacity of node i, that is, $ph(i) = C(i) - L(i)$.

How to generate and update pheromone is a key. For faster speed and lower cost, updating of pheromone is complete with DHT node routing table update process. The specific process is:

1. Source node S sends route table update message;
2. On receiving routing table update message, node i reply message with available capacity $ph(i)$;
3. On receiving a reply message, node S records nodes’ information in the routing table entry.

This generation and update mode does not need any additional structures to collect load information. It solves slow convergence problem caused by no significant differences of pheromone in the initial.
3.3.2. Routing Mechanism of Search Ant

Pheromone is the core of the solution based ant colony optimization, which represents some prior knowledge, and its size represents the load balancing effect. Safety factor is also introduced in routing selection as a heuristic factor, which guaranteeing the load transfer security. Therefore, both pheromone and safety factor must be considered in DHT routing. Suppose that node i receives search ant k, node i will select the neighbor node j as next hop by forward probability \( p_k(i, j) \).

\[
p_k(i, j) = \begin{cases} \frac{\sum_{w \in \text{routingTable}(i)} p_k(i, w) \cdot \text{rep}(w) \cdot \text{rep}(j)}{\sum_{w \in \text{routingTable}(i)} p_k(i, w) \cdot \text{rep}(w)}, & \text{if } ph(j) > 0 \& & j \in \text{routingTable}(i) - \text{tabu}(k) \\
\frac{\sum_{w \in \text{routingTable}(i)} p_k(i, w) \cdot \text{rep}(w) \cdot \text{rep}(j)}{\sum_{w \in \text{routingTable}(i)} p_k(i, w) \cdot \text{rep}(w)}, & \text{if } ph(j) < 0 \& & j \in \text{routingTable}(i) - \text{tabu}(k) \\
0, & \text{otherwise} \end{cases}
\]

(5)

Where, \( ph(j) \) represents the phenomenon value of node j; \( rep(j) \) represents the heuristic value of node j; \( \alpha \) and \( \beta \) is the relative important factor of pheromone and heuristic; \( \text{routingTable}(i) \) identifies the routing table entry of node i; \( \text{tabu}(k) \) is the taboo list of search ant k.

3.3.3. Life-span Control Policy

In this algorithm, the survival time of search ant is controlled by life-span control policy. To generate the Search ant, \( TTL \) will be set an initial value and minus one with every hop forward. When a node receives a search ant, the control of its life cycle as follows:

1. If the current node meets the restrictions, it should generate guide ant, update the \( TTL \) of search ant to 0 by force, and end forward search ant.
2. If its all neighbor nodes are in tabu list of search ant, this node should generate guide ant, update the \( TTL \) of search ant to 0 by force, and end forward search ant.
3. If neither the current node meets the constraints, nor all neighbors are in tabu list of search ant, then it should update search ant \( TTL \) to \( TTL = TTL - 1 \); and according to the formula (5), this node chooses neighbor node whose forward probability is largest as the next hop, and forwards search ant.

In the first case, search is ended as long as candidate node is found. In the other case, if \( TTL \) is 0, target transfer nodes should require two or more candidate light-nodes to share the overloaded load.

3.4. Target Node Selection Mechanism

Source node takes the candidate light-load nodes from guide ant as a node list M, and chooses the optimal set of one or more light-load nodes from list M. The metrics should consider two factors: 1) Load deviation rate reflects the overall distribution of load, here may be expressed in the variance of all transfer nodes and source node’ load deviation rate e. 2) Service reputation reflects the credibility of node’s provided service, which guaranteeing the load transfer security.

The objective function (obF) for target node selected and the constraints (s.t.) as follows.

\[
\text{obF} = \frac{\sum_{m = 1}^{M} L(m) - \text{Load}_\text{transfer}(m) \leq C(s) \cdot LT} {A \cdot \text{Load}_\text{transfer}(s) \leq \text{rep}(s) \leq B \cdot \text{rep}(m)}
\]

(6)

\[
\begin{align*}
\forall m \in M \quad L(m) &+ \text{Load}_\text{transfer}(m) \leq C(m) \cdot \overline{M} \\
\sum_{m = 1}^{M} \text{Load}_\text{transfer}(m) &\geq \text{Load}_\text{transfer} \\
\forall m \in M \quad \text{rep}(s) &\leq \text{rep}(m)
\end{align*}
\]

(7)

The selected target nodes (lists) are the candidate nodes who meet the constraints s.t. and the value of \( obF \) is largest. In formula (6)(7), where \( A \) and \( B \) are the weight of load deviation rate and service reputation; \( \text{Load}_\text{transfer} \) is total load which source node transfers out; \( \text{Load}_\text{transfer}(m) \) is the load of node m receives; \( \text{rep}(m) \) is the service reputation value of node m. The constraints are as
follows: the load utilization rate of overloaded node can drop below $LT$, the load utilization rate of light node can’t exceed $\mu$, and the service reputation value of light node is higher than resource node.

### 3.5. Overall Process of Load Balancing Algorithm

Once source node FN’s capacity utilization reaches $LT$, the flow is as follows:

1. Node FN generates $kth (k$ is initialized to 1) Search ant, and sets ant ID, constraints, node FN’s information (address, service reputation etc) and $TTL$;
2. According to the formula (5), node FN chooses the neighbor whose forward probability is $kth$ largest as the next hop, and forwards search ant $k$;
3. On receiving ant $k$, node DN puts its ID, pheromones and service reputation into the data structure of ant $k$. According to $s.t.$, node DN judges whether it is a valid candidate node:
   a. If node DN meets the $s.t.$, it should generate guide ant which return to node FN directly, update the $TTL$ to 0 by force, and end forward search ant $k$.
   b. If node DN does not meet the $s.t.$, and all neighbors are in tabu lists, it should generate guide ant which return to node FN directly, update the $TTL$ to 0 by force, and end forward search ant $k$.
   c. If neither node DN meets the $s.t.$, nor all neighbors are in tabu list of search ant, it should update $TTL$ to $TTL=TTL-1$; If $TTL=0$, node DN generates guide ant which return to node FN directly, otherwise, according to the formula (5), node DN chooses the neighbor whose forward probability is largest as the next hop, and forwards search ant, then go to step (3).
4. On receiving guide ant, source node FN should make all candidate node forms list, and selects target nodes who meet formula (7) and the value of formula (6) is largest, then doing data migration.
5. Source node FN calculates the new load capacity utilization $\mu$, if $\mu>LT$, then $k=k+1$, go to step (1); Otherwise the algorithm is end.

### 4. Performance Evaluation

This section compares load distribution, convergence rate and load balancing security among SANT-algorithm (A Security-aware Load Balancing Algorithm Based on Ant Colony Optimization), O2O-algorithm [6] (One-to-One algorithm) and M2M-algorithm [6] (Many-to-Many algorithm) in Chord context, and evaluates how system and algorithm parameters affect the performance of SANT-algorithm. Parameters used in this section are listed in Table 3 and Table 4.

#### Table 3. Basic experiment parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>node number</td>
<td>$[2^1, 2^{16}]$</td>
</tr>
<tr>
<td>Load</td>
<td>System load</td>
<td>$10^6$</td>
</tr>
<tr>
<td>$\overline{C}$</td>
<td>The mean of nodes’ capacity</td>
<td>$2.5*10^8/N$</td>
</tr>
<tr>
<td>$C$</td>
<td>node’s capacity</td>
<td>$[0.5\overline{C}, 2\overline{C}]$</td>
</tr>
<tr>
<td>$S$</td>
<td>the safety factor of node</td>
<td>$[1, 5]$</td>
</tr>
<tr>
<td>$\mu$</td>
<td>Node utilization rate</td>
<td>$[0, 1]$</td>
</tr>
<tr>
<td>$\overline{\mu}$</td>
<td>System utilization rate</td>
<td>0.4</td>
</tr>
</tbody>
</table>

#### Table 4. Balancing algorithm parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e$</td>
<td>The load deviation rate</td>
<td></td>
</tr>
<tr>
<td>$LT$</td>
<td>Adjust load capacity utilization trigger threshold</td>
<td>$[0.5, 0.8]$</td>
</tr>
<tr>
<td>TTL</td>
<td>maximal survival time</td>
<td>$\log_2N$</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>the relative importance factor of pheromone</td>
<td>$[0, 1]$</td>
</tr>
<tr>
<td>$\beta$</td>
<td>the relative importance factor of heuristic</td>
<td>$[0, 1]$</td>
</tr>
<tr>
<td>dNum</td>
<td>directory nodes number</td>
<td>$\log_2N$</td>
</tr>
</tbody>
</table>

#### 4.1. Performance comparison and analysis with the classical algorithm

SANT and O2O-algorithm are full distributed model, while M2M-algorithm is semi-centralized model. In SANT, once overloaded node’s $\mu$ reaches $LT$, it should execute to search for suitable light-load nodes to data migration; In O2O-algorithm, light-load nodes, whose $\mu<\overline{\mu}$, should random detect node periodically, if the node’s $\mu>LT$, light-load node will do data migration from this node; In M2M-algorithm, the assignment of loads is typically performed by one or more directory nodes, a directory node periodically receives nodes’ load information and executes the load balancing algorithm.

#### 4.1.1. Load Distribution
Load distribution is evaluated with the following considerations. 1) To evaluate the effect of load balancing, we compare the load distribution in SANT, O2O, M2M algorithms and without load balancing solution. 2) Load distribution is also compared among different LT (0.5, 0.6, 0.7, 0.8) in Chord system, whose \(N=2^{10}, \mu=0.4\). 3) The simulation runs under different capacity settings.

![Graph showing load distribution comparison](image)

**Figure 1.** Cumulative Distribution Function of \(\mu\). \(N=2^{10}\)

Fig. 1 shows the empirical CDF of \(\mu\). As shown, maximum \(\mu\) is reduced a lot when using load balancing algorithm, and the values of three algorithms is equal on \(\mu=0.2, 0.4, LT\), which means that the load balancing effect of three algorithms is basically similar. However, minimum \(\mu\) of M2M-algorithm is significantly bigger than the other two algorithms. It means that compared with fully distributed load balancing algorithms, semi-centralized load balancing algorithm is more efficient.

### 4.1.2. Increased Overhead and Security Caused by Load Balancing

For meeting the stringent load balancing requirement, all solutions have to adjust the load through transferring virtual servers, which results in increasing candidate node discovery overhead and data migration overhead a lot. Furthermore, reassign loads among nodes without considering safety factor, resulting in the transfer contents insecurity. Consequently, we will evaluate convergence rate, data migration overhead and security factor increment. Here, convergence rate is evaluated with the detect hops for candidate node discovery per heavy nodes. As shown in Fig. 2, experimental results as follows:

![Graph showing overhead comparison](image)

**Figure 2.** The overhead of load balancing

- The mean of the detection hops per heavy nodes for SANT, O2O and M2M algorithms are distributed in [15,20], [25,45], [110,150] range respectively, indicating that SANT can improve the convergence rate. Because it is a non-direct collaboration way, blind search services can be avoided.
- The data migration overhead, which is the total load transfer from overloaded nodes, is the same among three algorithms, while with the increase of \(LT\) is decreasing.
- Security factor increment per data migration of our SANT algorithm is larger than the other two algorithms obviously. It is because that candidate nodes discovery mechanism introduces safety factor, and target node selection mechanism makes a compromise between load balancing effect and security.
4.2. The Effect of System and algorithm parameters on algorithm performance

This section analyzes how system and algorithm parameters affect the performance of the proposed algorithm in depth. It is evaluated with the following considerations:

- This simulation analysis how the system size and load balancing requirements affect the performance of the SANT algorithm. Here, load balancing requirement is expressed by \( LT \), \( LT \) range \([0.5, 0.8]\), interval value is 0.05; system size \( N \) is in the range of \([2^1, 2^{16}]\).

- Mechanism based on Ant Colony contains a large number of adjustable parameters. However, the values of these parameters are not strictly theoretical guidance. So, it needs to determine the optimal parameter through several experiments. In formula (5), \([\alpha, \beta]\) is important to the SANT algorithm, so, in simulation, the value of \([\alpha, \beta]\) is \([0.4, 0.6], [0.5, 0.5], [0.6, 0.4], [0.8, 0.2]\) respectively.

- Different algorithm parameters \([\alpha, \beta]\) values, with the change of the system parameters \( LT, N \), this paper analyzes the standard deviation of the load deviation rate \( \varepsilon \), the mean of the detection hops per heavy nodes, data migration overhead and security factor increment per data migration.

![Figure 3](image1.png)

**Figure 3.** Simulation results under different system parameter \( LT \) and algorithm parameter \([\alpha, \beta]\)

![Figure 4](image2.png)

**Figure 4.** Simulation results under different system parameter \( N \) and algorithm parameter \([\alpha, \beta]\)

As shown in Fig. 3 and Fig. 4, experimental results are as follows:

- \( LT \) represents the load balancing requirements, greater \( LT \) has lower load balance requirements, so the load balancing effect becomes worse and the load balancing overhead decreases.
• With changing of the system size $N$, load balancing effect and overhead appear to be leveling off. While the fluctuation of detecting overhead is relatively large, because as $N$ increases, the ant's TTL is logarithmic growth. So SANT load balancing mechanism has a good scalability.

• In the candidate node discovery mechanism, $\alpha$, $\beta$ represent the relative importance of available capacity and security factor respectively. With the increase of $\alpha$ (decrease of $\beta$), load balancing effect appears a good trend, the convergence rate becomes faster, and smaller security factor is enhanced.

5. Conclusions

This paper has presented our load balancing algorithm called SANT, which is designed for DHT system. Considering load balancing effect and overhead, security, it designs a candidate nodes discovery mechanism and a target nodes selection mechanism. Two types of agents are cooperating to realize these mechanisms: Search ant and Guide ant. Performance evaluation shows that, compared to the classical algorithms, the load balancing effect of the SANT algorithm is insignificant, while the convergence rate and load balancing security are significantly raised. To address practical issues, we still need to implement SANT in real systems and test its performance in the future.

6. References