Wireless Pers Commun
DOI 10.1007/s11277-017-3943-7

A Delay Aware Super-Peer Selection Algorithm for Gradient Topology Utilizing Learning Automata

Sara Fathipour Deiman[1] · **Ali Mohammad Saghiri**[2] · **Mohammad Reza Meybodi**[2]

Abstract Peer-to-peer network is organized on top of another network as an overlay network. Super peer network is one of the peer-to-peer networks. A super peer, in a super peer based network, is a peer that has more responsibility than other peers have and is responsible for some of the tasks of network management. Since different peers vary in terms of capabilities, selecting a super peer is a challenge in super peer based networks. Gradient topology is of the networks based on super peers. Existing adaptive algorithms, which have been proposed to select super peer in gradient topology, are not aware of delays among the peers. In this paper, the proposed algorithm being aware of the delay among super peers, using learning automata, which is a reinforcement model of learning, selects the new super peers in an adaptive manner. According to the simulation results, the proposed algorithm with respect to the average end-to-end delay in community of super peers, and error in the super peer selection, has better performance than existing algorithms.

Keywords Super peer selection · Learning automata · Gradient topology · Delay

✉ Sara Fathipour Deiman
S.Fathipour.IT@gmail.com

Ali Mohammad Saghiri
a_m_saghiri@aut.ac.ir

Mohammad Reza Meybodi
mmeybodi@aut.ac.ir

[1] Faculty of Computer and Information Technology Engineering, Sama Technical and Vocational Training College Tehran Branch, Islamic Azad University, Tehran, Iran

[2] Soft Computing Laboratory, Computer Engineering and Information Technology Department, Amirkabir University of Technology (Tehran Polytechnic), 424 Hafez Ave., Tehran, Iran

Published online: 21 February 2017

1 Introduction

In Peer-to-peer networks, all peers have equal responsibilities. It means that at the same time a peer will act as a client also as a server [1]. When all the peers have the same responsibility and function, the network performance will be reduced. Any peer-to-peer network in which an attempt is made to instead of giving the same responsibilities to all peers, give more responsibility to the peers with higher ability, is called a super peer based network [2]. Peers with higher abilities, are called super peers [3]. Super peer based networks has many advantages such as Scalability, load balancing, and improving performance [4, 5]. Therefore, one of the challenges posed in peer-to-peer networks, which are super peer based, is the issue of selecting the super peer [6]. There are four ways to select the super peer, such as selecting the super peers in a simple method [7–9], selecting the super peers based on group [5, 10–12], selecting the super peers based on Distributed Hash Table [13–15], adaptive selection of super peers [16–25].

A lot of research has been done on the selection of super peer. Because of the dynamism of these networks, adaptive methods are important. In the adaptive method of selecting super peers, the management algorithm has been continioually tried to select appropriate super peers. In this method, selection of super peers is done according to criteria of peers' utility such as session time, storage space, processing power, bandwidth, capacity of peer, and workload [20].

In [16], the criteria of selecting a super peer are the capacity of the peers. In this algorithm, the ratio of super peers comparing the clients remains constant. The number of super peer is calculated based on the network characteristics. In [17, 18], the criteria of selecting a super peer are the capacity of the peers. In this algorithm, the capacity of super peers must be greater or equal to all clients. In this algorithm, the capacity of peer remains unchanged during the evolution of the network.

In [19], the criteria of selecting a super peer in SG-2 are the capacity of peer and distance among the peers. The SG-2 algorithm creates an overlay network with a small set of super peers in comparison with the size of the network. SG-2 is robust to removing the super peers but generate manay control messages. In [20], Myconet overlay model is presented, which are based on super peers. This network has characteristics such as self-organizing, resiliency and consistency facing with Churn and fast accesses the high levels of capacity utilization. In Myconet network, The capacity of a super peer is the number of peers that can be handled. In this network, super peers can be in different modes depending on the number of peers that are attached to them. In [21], PPT algorithm has been proposed, which is a super peer selection algorithm. In PPT algorithm, the gossip model has been used to exchange peer information in order to select super peer networks with high capacities.

Gradient topology, based on [22, 23] are made in a way that the utility of the peer is of special importance. The criteria of peers' utility are determined according to the particular application. Peers with high utility locate in the core of the topology, while peers with low utility locate in the around the core. Selecting neighborhood in gradient topology is based on the utility of the peer. GT algorithms in gradient topology use a fundamental property threshold to select the super peer, which is provided with one threshold and two thresholds. On a threshold, all peers that their utility is more than the threshold become super peer and others become normal peers. Due to continuous change of the threshold and the role of each node from normal peers to super peers and vice versa, large overhead is imposed on the network. To reduce this overhead, the two separate parts of the threshold selection are used to select the super peer. Peers that their utility is higher than the high threshold become super peers and the peers, which their utility is lower than the low threshold, become normal peers. In order to set an appropriate threshold for selecting super peers, we need overall information from the utility of the peers. However, it is noteworthy that overall information from the utility of the peers and calculating the threshold according to that is too costly. This algorithm to maintain the characteristics of the system is based on the gossip model.

In this paper, a delay aware super-peer selection algorithm based on learning automata will be proposed for gradient topology. Note that existing adaptive algorithms, which have been reported to select super peer in gradient topology, are not aware of delays among the peers. This paper is organized as follows. Section 2, an overview of learning automata that is used as the main learning strategy in the proposed algorithm, Sect. 3 states the problem, the proposed algorithm in Sects. 4, 5. Simulation results and in Sect. 6, the Conclusion is expressed.

2 Overview of the Learning Automata

Learning automata [26–29], are machines that can perform a finite number of actions. An environment evaluates every selected action, the outcome of evaluation is given to the automata in a positive or negative signal and the answer of the environment will affect on the automata to select the next action. The ultimate goal is that automata should to learn to choose the best action from its own actions. The best action is the one that maximizes the probability of receiving a reward from the environment.

Environment can be shown by $E \equiv \{\alpha, \beta, c\}$ in which $\alpha \equiv \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ is the Set of inputs, $\beta \equiv \{\beta_1, \beta_2, \ldots, \beta_m\}$ Set of outputs of the environment and $c \equiv \{c_1, c_2, \ldots, c_r\}$ is the set of possibilities of fines. When β_i has two values, the environment is called the B-model. In this environment, $\beta_i(n) = 1$ is assumed as a negative response or failure and $\beta_i(n) = 0$, as a positive response or success. In an environment of Q, β_i can discretely get one of the limited values in the interval [0,1]. In the S Model, β_i is a random variable between zero and one $\beta_i(n)\ \varepsilon\ [0,1]$.

Learning automata are divided into two groups of fixed and variable structure. In what follows, the automata with variable structure will be introduced. Learning automata with variable structure can be shown by the foursome of $\{\alpha, \beta, p, T\}$ that $\alpha \equiv \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ is a set of automata actions, $\beta \equiv \{\beta_1, \beta_2, \ldots, \beta_m\}$ set of automata inputs, $p \equiv \{p_1, p_2, \ldots, p_r\}$ action probability vector and $p(n + 1) = T[\alpha(n), \beta(n), p(n)]$ is the learning automata. In this type of learning automata, if the action α_i is done in the n-th stage and has received favorable response from the environment, the probability of $P_i(n)$ increases and the other probabilities decreases. However, changes happen in such a way that the sum of $P_i(n)$ always remains constant and equal to one (Fig. 1).

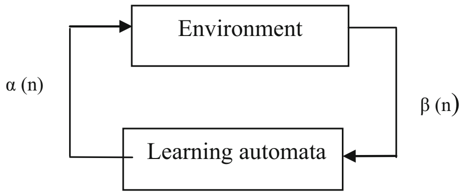

Fig. 1 Relationship between learning automata and environment

In Some applications automata with various number of actions is required [28]. This automaton at instant n selects its action only from non-empty subsets V(n) of actions, which are called active actions. Selection of V(n) subset is performed randomly by an external factor. The activity of the automaton is as follows: To choose an action in time n, first calculates the total probability of active measures K(n), and then calculates the vector P(n) according to the formula (1). Then automaton randomly selects a measure from the set of active measures according to the probability vector P(n) and applies on the environment. If the selected action is α_i.

$$\mathrm{P_i(n) = Prob[\alpha(n) = \alpha_i | V(n)\, is\, set\, of\, active\, actions,}\ \alpha_i \in \mathrm{V(n)] = P_i(n)/K(n)} \tag{1}$$

After receiving the response, probability vector automata P(n), updates its measures based on Formula 2 in case of receiving award and based on Formula 3 in case of receiving fine [28].

$$\begin{aligned} \mathrm{Favorable\, response:}\ \mathrm{P_i(n+1)} &= \mathrm{P_i(n) + a(1 - P_i(n))} \\ \mathrm{P_i(n+1)} &= \mathrm{P_j(n)} + a \cdot \mathrm{p_i(n)} \end{aligned} \tag{2}$$

$$\begin{aligned} \mathrm{Unfavorable\, response:}\ \mathrm{P_i(n+1)} &= \mathrm{(1-b)p_i(n)} \\ \mathrm{P_i(n+1)} &= \mathrm{(b/r - 1) + (1-b)p_j(n)} \end{aligned} \tag{3}$$

Then probability vector automata P(n), using the vector P(n + 1) and in the following day updates the measures.

$$\begin{aligned} \mathrm{P_j(n+1)} &= \mathrm{P_j(n+1) \cdot K(n)\, for\, all\, j,}\ \alpha_j \in \mathrm{V(n)} \\ \mathrm{P_j(n+1)} &= \mathrm{p_j(n)\, for\, all\, j,}\ \alpha_j \notin \mathrm{V(n)} \end{aligned} \tag{4}$$

In the above equations, appropriate and inappropriate responses, a is reward parameters and b is fine parameter. Based on the values of a and b three modes can be considered, When a and b are equal, the algorithm is called L_{RP}, When b is much smaller than a, the algorithm is called $L_{R\varepsilon P}$ and when b is equal to zero the algorithm is called L_{RI}.

Learning algorithm of learning automata, in model $S - L_{RP}$ appears as sub vector of automaton measures probabilities [28]. In automata with r action, if the action α_i is selected in n-th iteration step and the environment's answer to that is $\beta_i(n)$, the vector of automata probabilities will be updated according to the following equation.

$$\begin{aligned} p_i(n+1) &= p_i(n) + a \cdot (1 - \beta_i(n)) \cdot (1 - p_i(n)) - a \cdot \beta_i(n) \cdot p_i(n) \\ p_j(n+1) &= p_j(n) - a \cdot (1 - \beta_i(n)) \cdot p_j(n) + a \cdot \beta_i(n) \cdot \left[\frac{1}{r-1} - p_j(n)\right] - a \cdot (1 - \beta_i(n)) \\ &\quad \cdot p_j(n) \quad \forall j \quad j \neq i \end{aligned} \tag{5}$$

3 Statement of the Problem

Recently gradient topology comparing to other mentioned adaptive methods has been used in many applications [30–33] (Fig. 2). That is why it is very important to pay attention to the issue. The gradient topology for both peers of p and q if peer utility in p is greater than

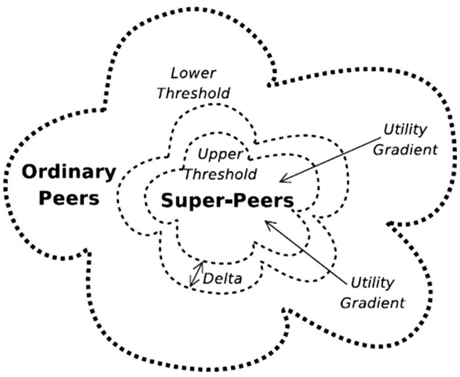

Fig. 2 Selecting the super peer using two thresholds [22]

the utility in q, i.e.: $U(p) \geq U(q)$, then it will be dist[1] $(p, p_0) \leq dist(q, p_0)$. Where dist (x, y) is defined as a measure of distance, i.e. the shortest path between x and y, it is noteworthy that p_0, is a peer in network that has the most utility. In this case, the distance between the peer is defined based on utility [22, 23].

In GT algorithm in gradient topology, the issue is that the super peer selection benefit is only done based on utility and is not aware of the delay between super peers and has no mechanism to calculate it, for selection of new super peers. For this reason in order to select super peers, periodically adjusted periodically calculates the adaptive threshold for super peer selection. This calculation provides a network overhead. The overhead caused by network characteristics gossip and estimation of super peer selection threshold. Each peer to become super peer in each period compares its utility with estimated threshold, which is a very costly. In GT algorithms that selection of super peers are done with two thresholds, the high threshold is shown by t_u and the low threshold is shown by t_l. The distance between the two thresholds is shown by Δ, whose value is obtained from this equation: $\Delta = t_u - t_l$.

The algorithm for calculating adaptive threshold finally will cause the increase or decrease of super peer selection threshold, a change that will affect the distance between the two thresholds, i.e. Δ. If the distance of Δ increases, the error in the selection of appropriate super peer also increases.

By having fixed threshold there will not be any problem bout calculation overhead and estimation of thresholds but the network must be able to learn adaptively, select the peers from delta distance to become super peers that the criteria of delay is considered in it, in a way that their delay be less than the current super peers.

4 The Proposed Algorithm

Our focus is on a peer-to-peer network based on gradient topology with two threshold and using learning automata to select super peer. In this paper, the learning automata are L_{RP} and the model of environment is s. Initially, the network topology and data structure are described, then we will explain the proposed algorithm (Fig. 3).

[1] Distance.

```
*/ Average_P (delay) = Average End to End delay;  Average*_P (delay) = Optimum End to End delay
   P_vector = Set of probability values for r;       θ = The number of iteration as super peer selection/*
while (TRUE)

    {
      if super-peer
          {
            Each Super peer calculate Average End to End delay;
          if Average_P (delay) > Average*_P (delay)
              {
                each super peer Find out neighbors of  super-peer for 1 Hop distance just  in Δ;
                  r = Find out neighbors of  super-peer for 1 Hop distance just in Δ;
                  P_vector = Create probabilities vector (r);
                  Initialize probabilities vector (P_vector);
        /* Select a peer as a super peer according to the action probability vector of learning automata */
                  do
                      {
                        select neighbor from P_vector;
                        compute ß by EQ (6);
                         Update LAP by EQ (5);
                      }while (θ=5)
              }
          }
    }
```

Fig. 3 LAGT.2 algorithm with knowledge of the delay

1. Create a network based on gradient topology with two thresholds. In the first round, the thresholds are estimated adaptively based on [22, 23], after the first round, instead of recalculating the adaptive threshold, the primary adaptive thresholds remain fixed.
2. The utility of each peer has been considered according to its capacity capacity. In addition, for peers P, utility U(p) is equal to C(p), the number of clients that give service to them.
3. It is assumed that the capacity of each peer is unique i.e. $U(p) \neq U(q)$ [22, 23].
4. We consider the relationship between all neighboring symmetric in a way that each peer discriminates between his own link and the link of its neighbors [22, 23].
5. All peers which are in society of super peers, have a learning automata with variable actions of [28] operates with r act of $\{\alpha_1, \alpha_2, \ldots, \alpha_r\}$. r is the numbers of neighboring super peers, which are located at a distance of Δ, are corresponding to the learning automata.
6. p_{vector} vector, is the vector of probability vector of selection probability of learning automata function where p_i is the a_i-th probability of function selection or the i-th probability of neighboring selection as a super peer.
7. Each super peer in its neighboring table keeps the number of attached super peers, the number of connected clients also the probability vector of learning automata.

The steps of the proposed algorithm LAGT.2 is as follows:

Step 1 Each P super peer, in each period, calculates its delay from its neighbors and keeps it in its neighborhood table.
Step 2 Then calculates its average end-to-end delay with its super peers, Average_p (delay).
Step 3 If the amount of Average_P (delay) was more than the optimal end-to-end delay, Average^*_P (delay), in the super peer p:

Step 4 Each super peer detects its (r) neighbors, which are located at a Δ distance and are only a Hop away.

Step 5 Corresponding automata to each super peer, makes the initial value of function selection probability vector equal to $p_i = \frac{1}{r}$.

Step 6 Each super peer's (p) learning automata using p_{vector} vector, randomly selects one of the neighbors (q) and as shown in the following formula, calculates the response of the environment i.e. the ß of the selected neighbor. The amount of dis is equal to (end-to-end delay), the average roundtrip time between the selected super peer and its selected neighbor.

$$ß = \frac{1}{\text{dis}} \tag{6}$$

Step 7 The more the amount of calculated ß of the selected neighbors is bigger than 1/Average*$_P$ (delay), i.e. the more it closer to 1, more reward it gets and its selection probability vector increases.

Step 8 Corresponding automata with super peer, again reviews the probability selecting its neighbors from Δ to become super peer. This step is repeated until one of the neighbors at the previous replication θ, is chosen.

Step 9 The peer, which has been selected in the previous replication θ, is selected as super peer and will be replaced by a neighbor of the sightly super peer that had the maximum delay in the neighborhood list in step 1.

All clients of the previous super peer will connect to the new super peer and the new super peer will connect to all super peers that the previous super peer was connected to.

5 Simulation Results

In order to stimulate the proposed super peer selection algorithm, the PeerSim [34] stimulation is used. The proposed algorithm LAGT.2 is compared with the GT algorithm [22, 23]. Table 1 summarizes the simulation assumptions with their initial values.

The simulation time is equal to 30 time rounds. Each round is equal to a second. Each test is performed for several times and the mean number is calculated.

5.1 Experiment 1

The purpose of this test is to evaluate the proposed algorithm from the perspective of end-to-end delay in super peers. Since the gradient topology is not sensitive to delay and super peers are only selected on the basis of utility, in this study we examine the average of end to end delay in super peers' community in each round, in the two algorithms of LAGT-2 and GT.

According to the simulation results, as can be seen in Fig. 4, the GT algorithm does not have a fixed approach and improvement in average of end-to-end delay in super peers' community. However, in the proposed algorithm, due to this algorithm's awareness of the delay and selecting the appropriate super peer considering the delay, comparing to the GT algorithm, has a far better performance. The vertical axis is in seconds.

Table 1 Simulation parameters

The maximum number of peer in networks	100000		
Maximum number of neighbors of each peer	150	Based on similarity	10–74
		At random	10–76
Utilities	0–1000		
Churn	50–10%		
Average*$_P$ (delay)	2 s		
θ	5		

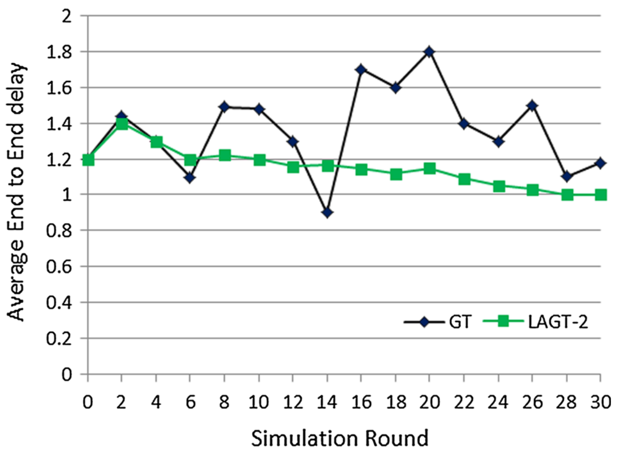

Fig. 4 The average of end-to-end delay in super peers' community

5.2 Experiment 2

In this experiment, the optimization of a function of the average end-to-end delay called Opt in super peers' community by increasing the simulation round is checked. In this experiment, it is assumed that the network has high churn. Optimization of the function of average end-to-end delay is calculated from the following equation.

$$\mathrm{Opt}(\mathrm{t} = \mathrm{current}) = \frac{endtoenddelay(t = 0) - endtoenddelay(t = current)}{endtoenddelay(t = 0)} \quad (7)$$

The proposed LAGT.2 algorithm has better performance than the GT algorithm. In LAGT.2 algorithm, by increasing the number of rounds, the rate for optimization of function of average end-to-end delay in super peers' community increases but in GT algorithm, no constant progression is visible, because selection of super peers in this algorithm is only based on utility. The following diagram is the average of calculated results (Fig. 5).

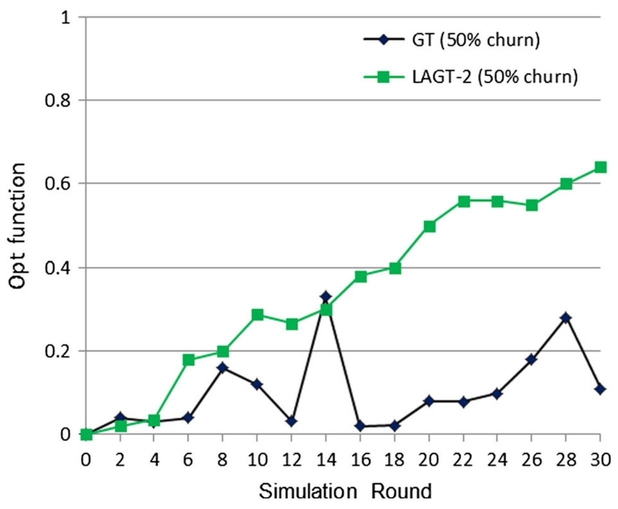

Fig. 5 Opt function of the average end-to-end

5.3 Experiment 3

In order to reduce the change in the role of each peer to super peer and vice versa, two thresholds were used but the tests has indicated that in GT algorithm by increasing the delta distance, the error rate of super peer selection increases (Fig. 6). However, in the proposed algorithm of LAGT-2 the error rate comparing to GT algorithm has been decreased a lot. The reason for error reduction in the proposed algorithm is that the super peers enjoying the learning automata, consider the delay of the peers in Δ distance and according to the formula ß (6) the peer that comparing to the sightly super peer, has less delay more appropriate to be selected. Super peer selection error is obtained based on the delay from the following formula:

$$\mathrm{Err_d} = \frac{\left|\mathbf{D_t} - \mathbf{D_t^*}\right|}{\mathbf{D_t^*}} \tag{8}$$

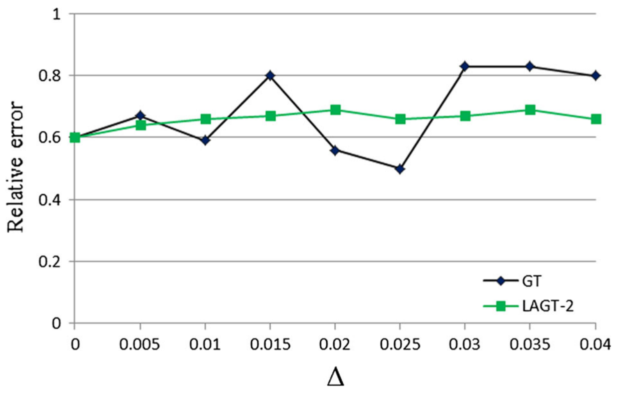

Fig. 6 Super peer select error with increasing the Δ distance

Notably, since the GT algorithm is not aware of the delay, thus increasing or decreasing the Δ distance does not have any effect on super peer selection error based on delay (Table 2).

5.4 Experiment 4

In this experiment, we study the capacity of super peers after removing 10% of super peers in around 10 (Fig. 7). Apart from the round 10, we do not have churn. We examine whether LAGT.2 and GT algorithms reach their maximum super peer capacity. In addition, how many rounds does it take in each algorithm to reach the maximum number of super peers? in GT algorithm, it takes about 14 rounds for the super peers to reach the maximize capacity. In the LAGT-2 algorithm, since the algorithm is aware of the delay but is not aware of the removal of super peers, if the average of end to end delay in super peers' community is more than the appropriate value, algorithm starts to select super peer otherwise does not do anything to select super peer. However, because the super peers will be selected from the community peers, which are the Δ distance, and also because the network topology is based on utility, by selecting super peer in the proposed algorithm, indirectly, the capacity of peer is also considered, but cannot be clearly concluded that the proposed algorithm can reach a maximum capacity of super peers.

5.5 Experiment 5

In this study, we study the number of super peers after removing 10% of super peers in round 10 (Fig. 8). In this experiment, we assume that the network has up to 3500 super peers and in around 10, the slightly percentage of super peers have been removed. Apart

Table 2 Delay parameters

Representing the average delay in super peers' community	$\mathbf{D_t}$
Representing the average optimal delay	$\mathbf{D_t^*}$

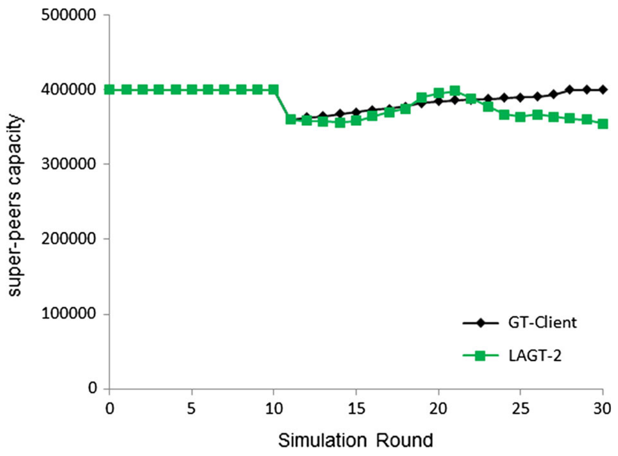

Fig. 7 The super peers' capacity in case of removing some percent of super peer

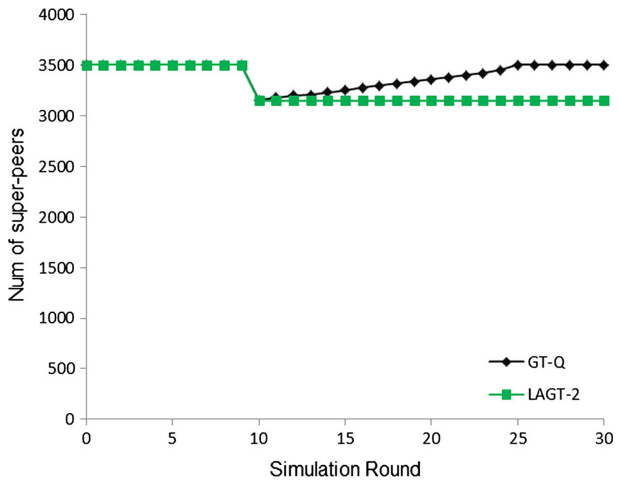

Fig. 8 The number of super peers after removal of some percent of super peers

from the round 10, we do not have any churn. Threshold in GT algorithm is considered based on the appropriate threshold, Q. Does LAGT.2 algorithm reach the maximum number of super peers? In the GT-Q algorithm after removal of 10% of super peers in round 10, it has taken about 14 rounds for the super peers to reach the sightly Max number. However, LAGT-2 algorithm does not reach the maximum number of super peers. Because if a super peer decides to choose a super peer, it is just because of the end-to-end delay and the previous super peer will replace the new selected super peer.

6 Conclusions

In this paper, an adaptive algorithm for super peer selection called LAGT-2 utilizing learning automata is proposed to manage the gradient topology. Algorithm LAGT-2, unlike GT algorithm in gradient topology that are not aware of super peers' delay, is the first algorithm that is aware of super peers' delay. According to the simulation, results show that the proposed algorithm compared to GT, in the average end-to-end delay in super peers' community, and reducing super peer selection error, has better performance.

References

1. Schollmeier, R. (2002). A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In *Proceedings of the first international conference on peer-to-peer computing*. Universitat Munchen, Germany (Vol. 1, pp. 27–29).
2. Yang, B., & Garcia-Molina, H. (2003). Designing a super-peer network. In *Proceedings of the 19th international conference on data engineering*, Bangalore, India. IEEE Computer Society (pp. 49–60).
3. Lua, E. K., Crowcroft, J., Pias, M., Sharma, R., & Lim, S. (2005). A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys and Tutorials, 7,* 72–93.
4. Karger, D. R., & Ruhl, M. (2004). Simple efficient load balancing algorithms for peer-to-peer systems. In *Proceedings of the 16th annual ACM symposium on parallelism in algorithms and architectures*, New York, NY, USA (pp. 36–43).

5. Lo, V., Zhou, D., & Li, J. (2005). Scalable super node selection in peer-to-peer overlay networks. In *Proceedings of the second international workshop on hot topics in peer-to-peer systems*, Washington, DC, USA. IEEE Computer Society (pp. 18–27).
6. Min, S. H., Holliday, J., & Cho, D. S. (2006). Optimal super peer selection for large-scale P2P system. In *International Conference on Hybrid Information Technology*.
7. Liang, J., Kumar, R., & Ross, K. (2006). The KaZaA overlay: A measurement study. In *Computer networks* (pp. 842–858).
8. LimeWire. Accessed on 25th March 2008, How gnutella works. http://wiki.limewire.org/index.php?title=How_Gnutella_Works.
9. Kleis, M., Lua, E. K., & Zhou, X. (2005). Hierarchical peer-to-peer networks using lightweight super peer topologies. In *Proceedings of the 10th IEEE symposium on computers and communications* (pp. 143–148).
10. Nejdl, W., Wolf, B., Qu, C., Decker, S., & Sintek, M. (2002). EDUTELLA: A P2P networking infrastructure based on RDF. In *Proceedings of the 11th international conference on World Wide Web*, ACM (pp. 604–615).
11. Wang, T. I., Tsai, K. H., & Lee, Y. H. (2004). Crown: An efficient and stable distributed resource lookup protocol. In *Proceedings of the 1st international conference on embedded and ubiquitous computing*. Springer, vol. 3207 of Lecture notes in computer science (pp. 1075–1084).
12. Aditya, K., Christopher, G., David, J. M., & George, K. (2015). Optimizing cluster formation in super-peer networks via local incentive design. *Peer-to-Peer Networking and Applications Journal, 8*(1), 1–21.
13. Pirro, G., Talia, D., & Trunfio, P. (2012). A DHT-based semantic overlay network for service discovery. *Future Generation Computer Systems, 28*(4), 689–707.
14. Li, Y., Huang, X., Ma, F., & Zou, F. (2005). Building efficient super-peer overlay network for DHT systems. In *Proceedings of the 4th international conference on grid and cooperative computing* (Vol. 3795, pp. 787–798). Springer.
15. Shi, L., Zhou, J., & Huang, Q. (2013). A Chord-based super-node selection algorithm for load balancing in hybrid P2P networks. In *IEEE/Mechatronic Sciences, Electric Engineering and Computer (MEC)* (pp. 2090–2094).
16. Xiao, L., Zhuang, Z., & Liu, Y. (2005). Dynamic layer management in superpeer architectures. *IEEE Transactions on Parallel and Distributed Systems, 16,* 1078–1091.
17. Montresor, A. (2004). A robust protocol for building superpeer overlay topologies. In *Proceedings of the 4th international conference on peer-to-peer computing* (pp. 202–209). IEEE Computer Society.
18. Gholami, Sh., Meybodi, M. R., & Saghiri, A. M. (2014). A learning automata-based version of SG-1 protocol for super-peer selection in peer-to-peer networks. In *Proceedings of the 10th international conference on computing and information technology (IC2IT2014)* (Vol. 265, pp. 189–201).
19. Jesi, G. P., Montresor, A., & Babaoglu, Ö. (2006). Proximity-aware superpeer overlay topologies. In *Proceedings of the 2nd IEEE international workshop on self-managed networks, systems, and services, vol. 3996 of Lecture notes in computer science* (pp. 43–57).
20. Snyder, P. L., Greenstadt, R., & Valetto, G. (2009). Myconet: A fungi-inspired model for superpeer-based peer-to-peer overlay topologies. In *3th IEEE international conference* (pp. 40–50).
21. Liu, M., Harjula, E., & Ylianttila, M. (2013). An efficient selection algorithm for building a super-peer overlay. *Journal of Internet Services and Applications, 4*(4), 1–12.
22. Sacha, J. (2009). *Exploiting heterogeneity in peer-to-peer systems using gradient topologies*. A thesis submitted to the University of Dublin, Trinity College in fulfillment of the requirements for the degree of Doctor of Philosophy (Computer Science).
23. Biskupski, B., Sacha, J., Dahlem, D., Cunningham, R., Meier, R., Dowling, J., & Haahr, M. (2010). Decentralising a service-oriented architecture. In *Peer-to-peer networking and applications*. The Netherlands, Ireland and Sweden (Vol. 3). Springer.
24. Garbacki, P., Epema, D. H. J., & Steen, M. (2010). The design and evaluation of a self-organizing superpeer network. *Computers IEEE Transactions, 59,* 317–331.
25. Teng, H. Y., Lin, C. N., & Hwang, R. H. (2013). A self-similar super-peer overlay construction scheme for super large-scale P2P applications. *Journal Information Systems Frontiers*, *16*(1), 45–58.
26. Narenda, K. S., & Thathachar, M. (1989). *Learning automata: An introduction*. New York: Prince-Hall.
27. Najim, K., & Poznyak, A. S. (1994). Learning automata: Theory and application. In *Proceeding of the Tarrytown*, Elsevier Science Publishing Ltd, New York.
28. Thathachar, M. A. L., & Bhaskar, R. H. (1987). Learning automata with changing number of actions. *IEEE Transactions on Systems, Man and Cybernetics*, *17*(6), 1095–1100.
29. Narendra, K. S., & Thathachar, M. A. L. (1989). *Learning automata: An introduction*. New York: Prentice Hall.

30. Payberah, A. H., Dowling, J., Rahimain, F., & Haridi, S. (2012). Distributed optimization of P2P live streaming overlays. *Springer Journal*, *94*(8), 621–647.
31. Payberah, A. H., Dowling, J., & Haridi, S. (2011). Glive: The gradient overlay as a market maker for meshbased P2P live streaming. In *Proceedings of the 10th IEEE international symposium on parallel and distributed computing (ISPDC'11)* (pp. 153–162).
32. Payberah, A. H, Dowling, J., Rahimian, F., & Haridi, S. (2010). Gradientv: Market-based P2P live media streaming on the gradient overlay. In *Proceedings of the 10th IFIP international conference on distributed applications and interoperable systems (DAIS'10)* (pp. 212–225).
33. Payberah, A. H, Dowling, J., Rahimian, F., & Haridi, S. (2010). Sepidar: Incentivized market-based P2P livestreaming on the gradient overlay network. In *Proceedings of the IEEE international symposium on multimedia (ISM'10)* (pp. 1–8).
34. Jelasity, M., Montresor, A., Jesi, G. P., & Voulgaris, S. (2009). PeerSim: P2P simulator. http://peersim.sourceforge.net.

Sara Fathipour Deiman received the B.S. and M.S. degrees in computer engineering from Islamic Azad University of Qazvin in 2011 and 2014, respectively. Her research interests include peer-to-peer networks, distributed systems, artificial intelligence, learning automata, reinforcement learning, parallel algorithms, and soft computing.

Ali Mohammad Saghiri received the B.Sc. and M.Sc. degrees in computer engineering in Iran, in 2008 and 2010, respectively. He is currently the Ph.D. student of computer engineering in AmirKabir University of Technology, Tehran, Iran. His research interests include peer-to-peer networks, distributed systems, artificial intelligence, learning automata, reinforcement learning, parallel algorithms, and soft computing.

Mohammad Reza Meybodi received the B.S. and M.S. degrees in Economics from Shahid Beheshti University in Iran, in 1973 and 1977, respectively. He also received the M.S. and Ph.D. degree from Oklahoma University, USA, in 1980 and 1983, respectively in Computer Science. Currently, he is a full professor in Computer Engineering Department, Amirkabir University of Technology, Tehran, Iran. Prior to current position, he worked from 1983 to 1985 as an assistant professor at Western Michigan University, and from 1985 to 1991 as an associate professor at Ohio University, USA. His research interests include wireless networks, fault tolerant systems, learning systems, parallel algorithms, soft-computing and software development.