A Smart Topology Construction Method for Anti-tracking Network Based on the Neural Network

Changbo Tian[1,2], YongZheng Zhang[1,2](✉), Tao Yin[1,2], Yupeng Tuo[1,2], and Ruihai Ge[1,2]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
{tianchangbo,zhangyongzheng,yintao,tuoyupeng,geruihai}@iie.ac.cn
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China

Abstract. Anti-tracking network is the effective method to protect the network users' privacy confronted with the increasingly rampant network monitoring and network tracing. But the architecture of the current anti-tracking network is easy to be attacked, traced and undermined. In this paper, We propose smart topology construction method (STon) to provide the self-management and self-optimization of topology for anti-tracking network. We firstly deploy the neural network on each node of the anti-tracking network. Each node can collect its local network state and calculate the network state parameters by the neural network to decide the link state with other nodes. At last, each node optimizes its local topology according to the link state. With the collaboration of all nodes in the network, the network can achieve the self-management and self-optimization of its own topology. The experimental results showes that STon has a better robustness, communication efficiency and anti-tracking performance than the current popular P2P structures.

Keywords: Anti-tracking network · Topology · Neural network · Cyber security

1 Introduction

1.1 Background and Motivation

Along with the popularization and development of network, the Internet has entered into every aspects of our lives. And the Internet has evolved into a global platform for social networking, finance, education, communication, health care and so on. Large amount of personal information is transferred in the network,

Supported by the national natural science foundation of China under grant No. U1736218.

and most of them has huge economic benefits. So, the easy reach of the Internet has posed a serious threat to the online privacy of network users [1].

Even though the end-to-end encryption technology has been developed very mature, but it can only protect the data content of communications against the eavesdropping of the adversaries. The significant information about the identity of the sender and the receiver or the network addresses of the source and the destination are still in danger [2]. The adversaries can easily monitor or eavesdrop the network users' online behavior, and aggregate the data of users' communication to make a big profit.

In order to protect the online privacy of network users, anonymous communication (AC) network emerges as the times require [3]. However, most of the current AC networks focus on the anonymity of network users' identities at the application layer. From the network layer, when confronting with the network tracing and monitoring, they have not too many advantages in tracking-resistance. So, we propose the anti-tracking network which is used to mitigate the network tracing and network monitoring in the network layer. The anti-tracking network is built on the P2P network and the weakness of the P2P network, such as the cut point and key point problem, the problem of topology management and so on [4], have a big impact on the performance of anti-tracking network. Motivated by this, we devote into the research of smart topology construction method to build a smart and robust anti-tracking network.

1.2 Limitation of Prior Art

Anti-tracking network is used to fight against the network tracing and network monitoring. Because the anti-tracking network is built on P2P network, there are some disadvantages which seriously affect the performance of anti-tracking network [5–7].

– Cut point and key point problem: Cut points and key points are the critical threats to the P2P-based anti-tracking networks. The disconnection of cut points will split the network into different blocks. The key points are the core points to transfer messages and they are always the bottleneck of message transmission. So, the attack to the cut points and key points is the common means to undermine the anti-tracking network. If so, the anti-tracking network can not provide good performance to the network users.
– The monitor of C&C channel: As we know, the traditional anti-tracking network needs the controller to construct, manage and optimize the network topology. The frequent communication between the anti-tracking network and the controller may lead to the exposure of the controller's identity [8,9]. Once the controller is traced, the anti-tracking network will be taken over or fall into an unavailable situation.
– The fixed transmission path: Most of the anti-tracking network has the fixed topology structure and fixed message transmission path. This provides tremendous convenience for the adversary to trace the traffic flow. And, the adversary can easily probe the topology structure and destroy the anti-tracking network.

1.3 Proposed Approach

To solve the problems discussed above, we propose a smart topology construction method for anti-tracking network based on neural network, called STon. The basic principle of STon is that we deploy the neural network algorithm on each node of the anti-tracking network. Each node collects its local network state periodically to generate the network state parameters. Then, each node executes the neural network algorithm with the parameters to decide the connection or disconnection with other nodes. Each node only has the ability to adjust its local network topology structure. But, according to the collaboration of all nodes in the anti-tracking network, the network has the ability to adjust and optimize its topology structure by itself. With STon, the anti-tracking network doesn't need the controller to manage or optimize the topology structure any more, it achieves the self-management and self-optimization of topology structure.

To make a straightforward sense of our proposal, we conclude STon with three steps:

– Deployment of neural network in each node.
– Collection of network state parameters.
– Calculate the link state and adjust the topology.

1.4 Contributions

We make three key contributions in this paper as follows.

– We propose a smart topology construction method (STon) for anti-tracking network. STon achieves the self-management and self-optimization of network topology and effectively avoid the emergence of cut points and key points. STon makes the anti-tracking network more robust.
– We propose a parameter collection algorithm (PCA) to collect the local network state and generate the corresponding parameters for the deployed neural network. PCA can effectively collect the network state even if the network topology is dynamic.
– We propose connection judgement algorithm (CJA) which calculates the link state with network state parameters. CJA achieves that each node can optimize its local topology according to its network state. With the collaboration of all nodes in the network, anti-tracking network has the ability to optimize its own topology and always keeps in a stable and robust situation by itself without the management of the controller.

2 The Overview of STon

The main principle of STon is that the nodes in anti-tracking network have the ability to change its link state dynamically according to the network state for the purpose that the network can manage and optimize its topology intelligently and automatically. So, the network does not change its topology arbitrarily, but changes it to a more stable and robust structure. In this way, the maintenance expense and potential risks [10,11] of P2P network would be curtailed greatly. And the anti-tracking performance and robustness of the network would be improved highly. Before the detail discussion of STon, we need to introduce two important concepts as follows.

- **Candidate node.** Candidate nodes are defined as the nodes which can be choosed to calculate the link state by the current node. We denote the neighboring nodes collection of the current node as N, and denote the current node's neighbor' neighbor collection as S. Then the candidate nodes collection $C = \{N, S\}$.
- **Node stable situation.** For the current node u and one of its candidate nodes v, node u calculate with the parameters of node v and get the result. If the result is 1 and $v \in N$, or the result is 0 and $v \in S$, then the current node is in *node stable situation*. Node stable situation means the link state of the current node has been the same with the adviced link state by the neural network, and needs no optimization.

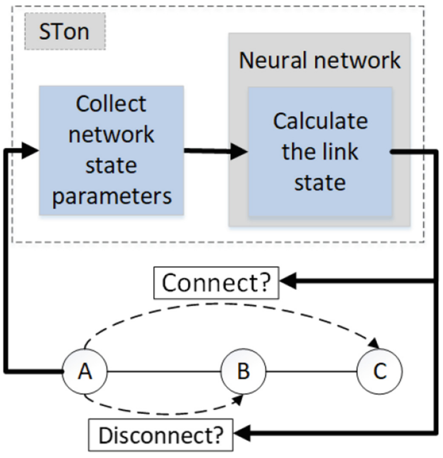


Fig. 1. The overview of STon.

Each node has to deploy the neural network, collect the network state parameters and calculate the link state with candidate nodes. As showed in Fig. 1, there is a simple topology that node A links to node B and node B links to node C. If node A is not in the *stable situation*, the node A needs to collect the network state parameters of node B and node C, then calculates link state with them. In Fig. 1, the link state between node A with other two nodes is subject to the calculation result of the neural network deployed in each node in advance. At last, with the calculation result, the node A updates its local topology.

So, STon has big advantages in the management and optimization of the topology for anti-tracking network. Anti-tracking network with STon will not need the controller's management in case of the traceback of C&C channel of the controller. From the aspect of robustness of network, the topology can change according to the changes of network state. With the parameters of network state, STon prefers to update its topology to a more stable and efficient structure in case of the emergence of cut points or key points. From the aspect of anti-tracking, the dynamic change of topology according to the network state will change the message transmission path which makes the network monitoring and traffic tracing more difficult.

3 Methodology

3.1 Parameter Collection Algorithm

Parameter Selection. How each node adjust its local topology is subject to the parameters calculated by the STon. In order to make sure the network optimizes its topology towards a more robust and anti-tracking structure, we comply with the following rules to select the network state parameters.

- **Robustness.** P2P-based anti-tracking network is an open and distributed platform. Each node can connect or disconnect with the network freely. This process may result in the emergence of cut points and key points which are the potential threats to the network. So, we need to select the parameters conductive to avoid the cut points and key points.
- **Anti-tracking.** STon has the ability to optimize its topology by itself. So, without the communication to the controller, C&C channel is hard to be monitored. Inspired by the infeasibility of monitoring the global Internet, cross-domain communication [12] is proposed to improve the anti-tracking performance. So, we also consider the cross-domain between two neighbouring nodes.
- **Invulnerability.** Invulnerability is the precondition of the availability of anti-tracking network. When some nodes, even the cut points or the key points, are removed, the network can still recover its topology to a robust and stable structure. This power is what the anti-tracking network needs. We also select the parameters which make sure the network has rapid resilience.

In view of the above, we classify the parameters into two categories showed as follows.

- The parameter of network topology. This kind of parameters can be counted directly according to the network topology information and used for the optimization of the network topology.
  - *C_Domain:* The domain of current node.
  - *D_Domain:* The domain of candidate node.
  - *C_Degree:* The degree of current node.

  - *D_Degree:* The degree of candidate node.
  - *C_CandidateAmount:* The candidate node number of current node.
  - *C_NeighborsAvgDegree:* The average degree of all neighboring nodes of the current node.
  - *SharingNode:* The sharing node number between the current node and the candidate node. If the two nodes are neighboring nodes, the parameter is 0.
  - *DifferenceOfDegree:* The difference of the degree between the current node and the candidate node.
- The parameter of network traffic. This kind of parameters is generated by the traffic information and used for the optimization of traffic load balance. In unit time, each node counts the amount of message transfered by it and calculates the average value to generate the parameters.
  - *C_MessageAvg:* The average transmission quantity of the current node in unit time.
  - *D_MessageAvg:* The average transmission quantity of the candidate node in unit time.
  - *C_NeighborsAvgMessage:* The average transmission quantity of all neighboring nodes of the current node in unit time.
  - *DifferenceOfAvgMessage:* The difference of average transmission quantity between the current node and the candidate node.

The original parameters need to be normalized and limited in the interval [0, 1] for the further calculation of the neural network.

C_Domain and *D_Domain* denote the domain of the current node and the candidate node. We firstly number each domain from $1 \sim n$ and n denotes the total number of all domains. Then the two parameters can be normalized by the Eq. (1) in which i denotes the domain number. Except the two parameters mentioned above, the other parameters have the value of *0* or any positive integer. So, we normalize the other parameters with Eq. (2) in which x denotes the parameter value.

$$f(i) = \frac{i}{n} \tag{1}$$

$$f(x) = \frac{1}{1+x} \tag{2}$$

Parameter Collection Algorithm. Each node uses PCA to collect the network state information and generates the corresponding parameters. For the parameter of network topology, only when the local topology of the current node changes, the current node updates this kind of parameters. For the parameter of network traffic, each node counts the amount of the message transfered by it in each unit time and computes the average value as this kind of parameters. Because the parameter of traffic information changes as the network traffic changes, we need recalculate the parameter of traffic information to get the latest network traffic information in each unit time. Algorithm 1 gives the pseudocode of PCA. In Algorithm 1, we can only collect the basic parameters directively generated by the network topology and network traffic. The other parameters, such as *C_CandidateAmount*, *C_NeighborsAvgDegree*, *SharingNode*, *DifferenceOfDegree*, *C_NeighborsAvgMessage*, *DifferenceOfAvgMessage*, need further calculations of the basic parameters from the current node and its candidate nodes.

Algorithm 1. Parameter Collection Algorithm

Input: v: current node. *totalmessages*: the total amount of transfered messages.
Output: P: Parameter collection

```
1: function UpdateTopologyPara($v$)  ▷ The function of collecting the parameter of network topology.
2:     $domain \leftarrow GetDomain(i)$  ▷ Get the parameter of domain.
3:     $degree \leftarrow GetDegree(i)$  ▷ Get the parameter of degree.
4:     $P \leftarrow domain, degree$
5: end function
6:
7: function PCA($v, totalmessages$)
8:     while True do
9:         if $TopologyChange(v)$ then  ▷ Check whether the topology has been changed. If $True$, optimize the parameters of network topology.
10:            UpdateTopologyPara($v$)
11:        end if
12:        if $UnitTime(v)$ then  ▷ Check whether it's time to update the parameters of network traffic.
13:            $avgmessage \leftarrow totalmessages/totaltimes$  ▷ Calculate the average value of total amount of transfered messages.
14:            $P \leftarrow avgmessage$
15:            $totaltimes++$  ▷ Accumulate the number of unit times.
16:        end if
17:    end while
18: end function
```

3.2 Connection Judgement Algorithm

With CJA, the anti-tracking-network can achieve the self-management and self-optimization of topology. In detail, each node asks for the parameters of all its candidate nodes, and executes CJA with its candidate node one by one to calculate the link state. At last, this node updates its local topology according to the calculated link state.

The Structure of the Neural Network. Firstly, we introduce the neural network deployed in each node. We use two RBMs to construct a two-layer neural network. The first RBM is Gaussian-Bernoulli RBM (GBRBM) [13], because the input of the first RBM is real number distributed in the interval $[0, 1]$, but the output value is 0 or 1. The second RBM is Bernoulli-Bernoulli RBM (BBRBM) [13], both of its input and output are 0 or 1. The structure of the proposed neural network is showed in Fig. 2. The input of GBRBM is the parameters collected by PCA, and the output of GBRBM is as the input of BBRBM to calculate the link state.

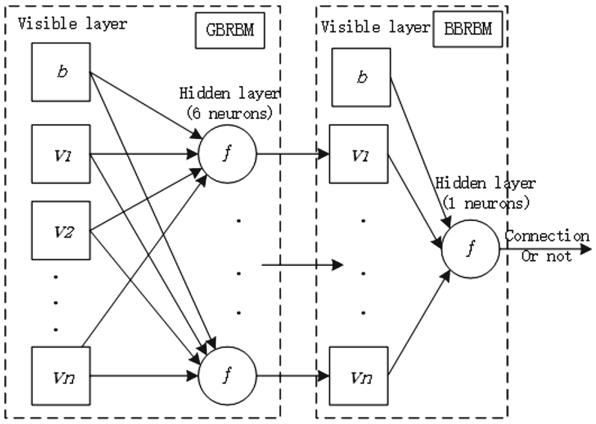


Fig. 2. The structure of neural network.

Parameter Training of Neural Network. In the parameter training of RBM, we usually use Contrastive Diversity (CD) [14] to train the weights between the nodes in different layers and the bias of each layer.

In order to improve the accuracy and efficiency of parameter training, we use genetic algorithm (GA) to optimize the weights of the neural network. The fitness function of GA is defined based on the distribution of nodes, domains and the traffic load of each node. The fitness function is showed in Eq. (3). α, β, γ are the coefficients. d, d_{avg} separately denote the degree of current node and the average degree of its neighboring nodes. num_i denotes the number of nodes in the same domain with the current node. m, m_{avg} separately denote the amount of messages transfered by the current node and average amount of messages transfered by all its candidate nodes.

$$F = \alpha \left(\frac{1}{1 + |d - d_{avg}|} \right) + \beta \left(\frac{1}{1 + num_i} \right) + \gamma \left(\frac{1}{1 + |m - m_{avg}|} \right) \qquad (3)$$

Crossover and mutation are the important steps to optimize the computational process of GA. Firstly, we need to calculate the crossover and mutation rates according to the evolution situation. In the optimization process of weights, in order to keep both the items with high fitness value and the diversity of the population in GA, we use Eq. (4) to calculate the crossover and mutation rates.

In Eq. (4), λ_1, λ_2 are the constants in the interval $(0, 1)$. f_{max} denotes the highest fitness in the population of GA. f_{avg} denotes the average fitness of the whole population.

$$p = \begin{cases} \lambda_1 \frac{f_{max}-f}{f_{max}-f_{avg}} & f \geq f_{avg} \\ 1 - \lambda_2 \frac{f_{max}-f_{avg}}{f_{max}-f} & f < f_{avg} \end{cases} \tag{4}$$

If we use Eq. (4) to calculate the crossover rate, f denotes the higher fitness of the two items in crossover calculation. If we use Eq. (4) to calculate the mutation rate, f denotes the fitness of the item in mutation calculation. So, with Eq. (4), when the fitness is too high, the crossover and mutation rate are low so that GA can avoid the premature convergence. When, the fitness is too low, the crossover and mutation rate are high so that GA can expand the item range to search the best solution.

In crossover calculation, we take two different crossover methods to generate two different items. One item prefers a balanced crossover of its parents, but another item prefers crossover with its parent has high fitness. We use g_i and h_i separately denote the genetic value of two items in crossover calculation, and s_i denotes the genetic value of a new item generated by the crossover calculation of g_i and h_i. The crossover calculation is showed in Eq. (5) in which p_c denotes the crossover rate. If we calculate the balanced crossover, we use Eq. (6) to replace the parameters Δ_i and Θ_i in Eq. (5). If we calculate the crossover inclined to the parent with higher fitness, we use Eq. (7) to replace the parameters Δ_i and Θ_i in Eq. (5).

$$s_i = p_c \Delta_i + (1 - p_c)\Theta_i \tag{5}$$

$$\begin{aligned} \Delta_i &= g_i \\ \Theta_i &= h_i \end{aligned} \tag{6}$$

$$\begin{aligned} \Delta_i &= \begin{cases} g_i & f_i \geq f_j \\ max(p_c g_i + (1 - p_c)h_i, g_i) & f_i < f_j \end{cases} \\ \Theta_i &= \begin{cases} g_j & f_j \geq f_i \\ max(p_c g_j + (1 - p_c)h_i, g_j) & f_j < f_i \end{cases} \end{aligned} \tag{7}$$

In mutation calculation, we randomly choose one value from the interval $[x_1, x_2]$ with the probability p_m to replace the genetic value of a item. The upper and lower bounds of the above interval are defined as Eq. (8) in which x_{min} and x_{max} separately denote the minimum and maximum weight of the neural network to keep the mutation calculation in a reasonable range. When the fitness of the item is approximate to the best fitness of the population, the interval of mutation will be small; on the contrary, when the fitness of the item is too low, the interval of mutation will be large. In this way, we not only avoid the impact of mutation to the items with high fitness, but also expand the range of mutation items in case of the premature convergence.

$$\begin{aligned} x_1 &= (1 + \frac{f}{f_{max}})x_{min} \\ x_2 &= (1 - \frac{f}{f_{max}})x_{max} \end{aligned} \tag{8}$$

GA is used to search for the approximate optimal weights for the neural network. When the GA converges, we can use the parameters generated by GA to train the neural network for higher efficiency and fast convergence. The training method of RBM has been discussed detailly in [14], and needs no further elaboration.

Connection Judgement Algorithm. CJA is used to calculate the link state according to the network state. CJA is a two-layer neural network based on RBM. Each node uses CJA to calculate the network state parameters to decide the link state with its candidate nodes. And then, the node updates its local topology according to the calculated link state.

The output of CJA is 0 or 1. If the output is 0, the node will break the connection with the candidate node if they are neighboring nodes. If the output is 1, the node will ask for connection to the candidate node and send its network state parameters to it. If the candidate node receives the parameter sent by the node who asks for the connection, it will also execute the CJA with the received parameters to make its own decision. Only when the output of the candidate node's CJA is also 1, the two nodes will build the connection. Or, the candidate node will refuse the connection. The pseudocode of CJA is showed as Algorithm 2.

4 Experiment and Analysis

We propose STon to achieve the self-management and self-optimization of anti-tracking network's topology. The anti-tracking network with STon has better performance in robustness, load balance and anti-tracking performance. We compare STon with the two popular P2P topologies, ZeroAccess [15] and TDL-4 [16], and analyze the advantages of STon.

ZeroAccess is based on a layered and unstructured P2P topology and its structure is familiar with Gnutella network [17]. So, we use the open source data of Gnutella topology from Stanford university [18] to simulate the ZeroAccess. TDL-4 is structured P2P topology based on Kademlia protocol [19]. We also use the open source date of Kademlia topology from Illinois university [20] to simulate the TDL-4. We use Igraph to simulate the above three P2P topology structures with 6000 nodes.

4.1 Evaluation of Robustness

To evaluate the robustness [21], we firstly give a evaluation formula (9) to quantify the evaluation results.

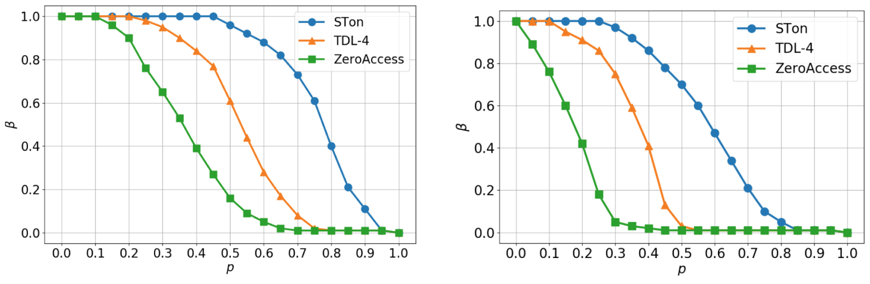
$$\beta_p = \frac{The\ node\ number\ of\ MCS(G(p))}{The\ node\ number\ of\ G} \tag{9}$$

In formula (9), $G(p)$ denotes the subgraph after remove p percent of the nodes from the original network. $MSC(G(p))$ denotes the maximum connected subgraph of $G(p)$. In the experiment, we use two different ways to remove nodes from the network to compare the robustness.

Algorithm 2. Connection Judgement Algorithm

```
Input: $u$:current node. $C$: candidate nodes collection. $N$: neighboring nodes collection.
 1: function CalculateParameters($u, C, N$)    ▷ The function of calculating the parameter of network topology.
 2:     for $v$ in $C$ do
 3:         $parameter = GetParameter(v)$ ▷ Get the parameters of candidate node $v$.
 4:         $connection = RBM(parameter)$    ▷ Calculate the link state.
 5:         if $connection = 0$ then
 6:             if $v$ in $N$ then
 7:                 $Breaklink(v)$    ▷ Break the connection with node $v$ if they are neighboring nodes.
 8:                 $Update(u)$    ▷ Update the network state parameter and candidate nodes collection of current node.
 9:             end if
10:         else if $connection = 1$ then
11:             $parameter_u = GetParameter(u)$  ▷ Get the parameter of current node $u$.
12:             $result = RequestLink(v, parameter_u)$    ▷ Current node ask for connection with the candidate node $v$.
13:             if $result = 1$ then ▷ when the calculation result of candidate node $u$ is 1, it means the candidate node also agrees with the connection with current node.
14:                 $BuildConnection(v)$  ▷ Build connection between current node and candidate node.
15:                 $Update(u)$
16:             end if
17:         end if
18:     end for
19: end function
20:
21: function ListenConnectionRequest($v$)    ▷ The function of listening the request of connection.
22:     while GetConnectionRequest() do    ▷ If there is a request of connection, the current node need to calculate and judge the connection with it.
23:         $parameter_v = GetParameter(v)$
24:         $connection_v = RBM(parameter_v)$
25:         if $connection_v = 0$ then
26:             $RefuseLink(v)$  ▷ If the calculatoin result is 0, current node refuses to build connection with it.
27:         else if $connection_v = 1$ then
28:             $BuildConnection(v)$
29:             $Update(u)$
30:         end if
31:     end while
32: end function
```

– **Random-p removal.** In each round, we remove p percent of nodes from the network randomly.
– **Top-p removal.** In each round, we remove p percent of nodes with highest degree from the network.



(a) Random-p Removal (b) Top-p Removal

Fig. 3. The influence of node removal on the robustness of networks

From the experiment results in Fig. 3, no matter in which node removal experiment, STon has the best robustness. Because when some nodes disconnect the network, the network topology will be changed and also may be broken into different blocks. Without the management of the controller, the TDL-4 and ZeroAccess can not management their topologies by themselves, so the topologies of the two P2P networks will be more worse along with the more nodes disconnect from the network. But, STon has the ability of self-management of its topology. STon can perceive the changes of topology and optimize it according to the network state to a stable situation by the calculation of network state parameters. From the two experiments, we can see that the top-p removal has the greater damage to the network, because the nodes with highest degree always have a more important position in network communication. So, TDL-4 and ZeroAccess perform worse in top-p removal. But for STon, the performance of the two experiments decreases not too much. This is because the self-optimization of topology always keeps STon in a stable situation as the network state changes.

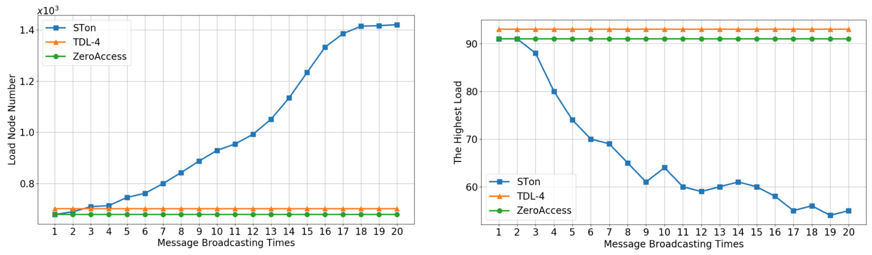
4.2 Evaluation of Load Balance

Network load balance is an important evaluation index of quality of service for anti-tracking network [22]. We evaluate the network load of the three topology structures with the following index under the same network size and the same message transmission volume.

- **Load node number:** The number of nodes take part in the message transmission in the network.
- **The highest load:** In each round of message transmission, the amount of transferred message by the node with highest load in the network.

The index of *load node number* is higher, then in the same amount of transferred message, more nodes take part in the message transmission. In this way, the transferred message will be shared to more nodes in case that a few nodes take on the more traffic load. The index of *highest load* is to measure the network load through the worst case.

We set up 20 rounds for message broadcasting and the TTL of the message is 4. After each round, we count the corresponding index of network load to evaluate the load balance performance of the three topology structures.



(a) The changes of load node number according to the continuous broadcasting (b) The changes of the highest load according to the continuous broadcasting

Fig. 4. The performance of network load balance comparision experiment

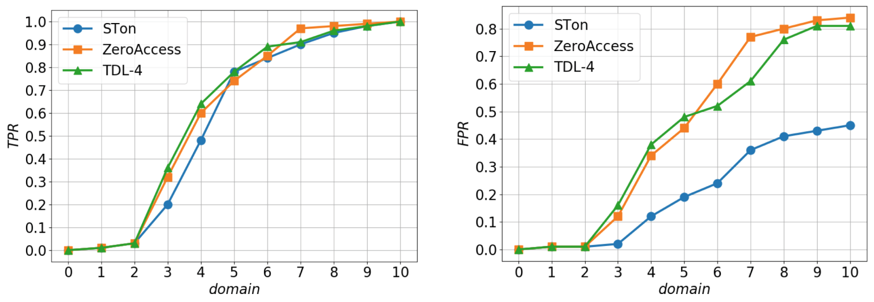
As illustrated in Fig. 4, TDL-4 and ZeroAccess have no ability of self-optimization. So, if the topology structure has no changes, the network traffic load also has no changes. But, STon has an obvious changes in the index of *load node number* and *the highest load*. This means STon can optimize its topology as the network traffic changes. The ability of self-management and self-optimization of topology will provide huge convenience and advantages to ensure the efficient communication and strong tracking-resistance for anti-tracking network.

4.3 Evaluation of Anti-tracking

Anti-tracking is the priority for anti-tracking network. The monitoring range of the adversary has a direct impact on the intensity of anti-tracking performance. So, we use the number of traced nodes by the adversary to measure the anti-tracking performance. Firstly, we need introduce two indexes for anti-tracking evaluation as follows.

– **TPR: True Positive Rate.** TPR denotes the proportion of the traced target nodes in all target nodes. TPR is used to evaluate the anti-tracking ability of the network. The less target nodes are traced, the stronger anti-tracking performance the network has.
– **FPR: False Positive Rate.** FPR denotes the proportion of the traced target nodes in all the traced nodes. FPR is used to evaluate the difficulty of the adversary to trace the target nodes. Because the adversary can trace a lot of nodes in the network, but may be hard to trace the target nodes.

We randomly choose 50% nodes as target nodes from the network to send message to a specific node. Assume that the adversary can only monitor the traffic flow to trace the target nodes. For convenience, we assign all the nodes into 10 domains equally. We assume the adversary can monitor one or more domains to trace the target nodes. By increasing the monitoring domains, we evaluate the anti-tracking performance of the three topology structures with the index of TPR and FPR.



(a) The changes of TPR along with the increase of monitoring domains

(b) The changes of FPR along with the increase of monitoring domains

Fig. 5. The performance of anti-tracking comparision experiment

In Fig. 5, the values of X-axis are the number of domains monitored by the adversary. In Fig. 5(a), along with the increase of monitoring domains, the number of traced target nodes increases exponentially and the performance of the three topology structures is almost the same. This is because the anti-tracking performance is subject to the monitoring range of the network by the adversary. Just think that if the adversary can monitor the whole the Internet, there will be no chance to escape from the network tracing and network monitoring. But in Fig. 5(b), the FPR of STon increases slowly. From the anti-tracking experiment, we can conclude that STon improve the anti-tracking performance by the means of dynamic changes of network topology and transmission path. In this way, the adversay will trace too many unrelated nodes. So, STon makes the adversary to trace the target nodes more difficult.

5 Conclusion and Future Work

In this paper, we propose an smart topology construction method for anti-tracking network based on the neural networks, called STon. STon achieves the self-management and self-optimization of network topology which improve the robustness, communication efficiency and anti-tracking performance of the network. Firstly, we use genetic algorithm to optimize the weights of the neural network and help the neural network converge fastly. After deployed the neural network on each node of the network, each node has the ability to calculate the network state parameters to evaluate the link state. With the calculation result of the neural network, each node can optimize its local topology. With the collaboration of all nodes in the network, the network has the ability of self-management and self-optimization of its topology. This kind of network has stronger robustness, communication efficiency and anti-tracking performance than the current popular network topology structures.

But, there are still some weakness in our work. Firstly, we need to train the neural network previously to make sure the algorithm can suit to the network state in practice. Once the network state has a drastic change, and the network state may be totally different with the training data. Then, the performance of the neural network may be reduced greatly, even lose the efficacy. In the future, we will take further research on the online weight learning and rapid adaption to new network state of the neural network.

Acknowledgements. We thank the anonymous reviewers for their insightful comments. This research was supported in part by the national natural science foundation of China under grant No. U1736218.

References

1. Shirazi, F., Simeonovski, M., Asghar, M.R., et al.: A survey on routing in anonymous communication protocols. ACM Comput. Surv. (CSUR) **51**(3), 51 (2018)
2. Ren, J., Wu, J.: Survey on anonymous communications in computer networks. Comput. Commun. **33**(4), 420–431 (2010)
3. Dixon, L., Ristenpart, T., Shrimpton, T.: Network traffic obfuscation and automated internet censorship. IEEE Secur. Priv. **14**(6), 43–53 (2016)
4. Kang, S.: Research on anonymous network topology analysis. In: 2015 International Conference on Automation, Mechanical Control and Computational Engineering. Atlantis Press (2015)
5. Feinerman, O., Haeupler, B., Korman, A.: Breathe before speaking: efficient information dissemination despite noisy, limited and anonymous communication. Distrib. Comput. **30**(5), 339–355 (2017)
6. Zang, W., Zhang, P., Wang, X., et al.: Detecting sybil nodes in anonymous communication systems. Procedia Comput. Sci. **17**, 861–869 (2013)
7. Mittal, P., Borisov, N.: Information leaks in structured peer-to-peer anonymous communication systems. ACM Trans. Inf. Syst. Secur. (TISSEC) **15**(1), 5 (2012)

8. Caballero, J., Poosankam, P., Kreibich, C., et al.: Dispatcher: enabling active botnet infiltration using automatic protocol reverse-engineering. In: Proceedings of the 16th ACM Conference on Computer and Communications Security, pp. 621–634. ACM (2009)
9. Cho, C.Y., Caballero, J., Grier, C., et al.: Insights from the inside: a view of botnet management from infiltration. In: Usenix Conference on Large-scale Exploits & Emergent Threats: Botnets (2010)
10. Danezis, G.: Designing and attacking anonymous communication systems. University of Cambridge, Computer Laboratory (2004)
11. Kotzias, P., Matic, S., Caballero, J.: CARONTE: detecting location leaks for deanonymizing tor hidden services. In: ACM SIGSAC Conference on Computer & Communications Security. ACM (2015)
12. Yin, T., Zhang, Y., Li, J.: AppBot: a novel P2P botnet architecture resistant to graph-based tracking. In: 2016 IEEE Trustcom/BigDataSE/ISPA, pp. 615-622. IEEE (2016)
13. Yamashita, T., Tanaka, M., Yoshida, E., et al.: To be Bernoulli or to be Gaussian, for a restricted Boltzmann machine. In: International Conference on Pattern Recognition, pp. 1520–1525. IEEE Computer Society (2014)
14. Hinton, G.E.: Training Products of Experts by Minimizing Contrastive Divergence. MIT Press, Cambridge (2002)
15. Kerkers, M., Santanna, J.J., Sperotto, A.: Characterisation of the Kelihos.B Botnet. In: Sperotto, A., Doyen, G., Latré, S., Charalambides, M., Stiller, B. (eds.) AIMS 2014. LNCS, vol. 8508, pp. 79–91. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43862-6_11
16. Golovanov, S., Soumenkov, I.: TDL4 top bot. Kaspersky Lab Analysis (2011)
17. Ripeanu, M.: Peer-to-peer architecture case study: Gnutella network. In: Proceedings of the First International Conference on Peer-to-Peer Computing, pp. 99–100. IEEE (2001)
18. Leskovec, J.: Stanford Large Network Dataset Collection (2014) [OL]. http://snap.stanford.edu/data/index.html
19. Maymounkov, P., Mazières, D.: Kademlia: a peer-to-peer information system based on the XOR metric. Revised Papers from the First International Workshop on Peer-to-Peer Systems (2002)
20. Godfrey, B.: Repository of Availability Traces (2015) [OL]. http://pbg.cs.illinois.edu/availability/
21. Scott, D.M., Novak, D.C., Aultman-Hall, L., et al.: Network robustness index: a new method for identifying critical links and evaluating the performance of transportation networks. J. Transp. Geogr. **14**(3), 215–227 (2006)
22. Zeng, Z., Veeravalli, B.: Design and performance evaluation of queue-and-rate-adjustment dynamic load balancing policies for distributed networks. IEEE Trans. Comput. **55**(11), 1410–1422 (2006)