International Journal of Future Generation Communication and Networking
Vol. 9, No. 7 (2016), pp. 197-204
http://dx.doi.org/10.14257/ijfgcn.2016.9.7.19

A Protocol Model of S^3 Computing Designed for Learning Community Platform of College Teachers

Liang Jia[1] and Liuhong Yan[2 *]

1 School of Information Science & Engineering, Chang Zhou University, China
2 Zhou Youguang School of Languages and Cultures, Chang Zhou University, China
** E-mail: Sanctifier@vip.qq.com*

Abstract

S^3Computing requires distributed computing performed by social network platform has high scalability and security. Protocol models meeting the requirements of S^3Computing not only ensure the correctness and robustness of distributed computing, but also reduce risks introduced by involvement of nodes with low reputation in computing. These models safeguard the data collections and computations performed on platform of teacher's learning community for social researches. This paper constructs a protocol model entitled $\mathcal{LC}$ which adapts platform of teacher's learning community and meets the requirements of S^3Computing. This protocol model is the key step of implementing distributed computations on learning community platform.

***Keywords**: Learning community, social network, distributed computing, security, privacy*

1. Introduction

As investments in education and scientific research are continuously growing on a global scale, teaching and research conditions are greatly improved especially for college teachers in developing countries. Hence, college teachers in these countries can perform researches close to international academic level. They need to keep pace with international academic circle by improving their competences in researches and enhancing the connections with other researchers, especially members of their research teams. As a social interactive form across research fields which aims to improve competences of participants in research, learning community of college teachers provides a safe and convenient environment for research discussions across fields and countries, and a platform for performing social researches. Hence, sociability, scalability and security of procedure of data collection and processing on the platform should be guaranteed. These three features are defined as S^3 problem in [1] and computation for solving S^3 problem is termed as S^3 computing.

Sociability of S^3 computing refers to fact that participants of computations involve users of social network. Scalability means the spacial, computational and information complexities won't grow rapidly after user number exploded. Security consists of accuracy and privacy. Accuracy means computational robustness is maintainable when low reputation users participate in computation, and privacy refers to fact that inputs of standard reputation users are very hard to be obtained by analyzing computational results, and the final computational result is very hard to distorted by altering intermediate results.

Protocol models proposed in [2-4] involve partial requirements of S^3 computing, but scalabilities of these models are limited. [12] describes an improved security strategy but it requires distributed computation can only be performed by nodes trust each other. Scalability is also mentioned in [13], but error tolerances of its protocol model are limited.

Although the model proposed in [16] has a better scalability than the one in [13], it requires the agent nodes are assigned with unique identities which decrease privacy.

The rest of this paper is organized as follows. Section 2 contains two sections. The qualified definitions of scalability, accuracy and privacy are made in subsection 2.1 and existence of protocol satisfying conditions of S^3 computing is proved in subsection 2.2. Section 3 consists of four subsections. Subsection 3.1 introduces general scheme of protocol model $\mathcal{LC}$, subsections 3.2 to 3.3 describe the design details of three phrases of $\mathcal{LC}$. The conclusion is drawn in section 4.

2. Protocol Model Satisfying S^3 Computing

This section first introduces the definitions of scalability, accuracy and privacy in S^3 computing, and then gives the proof of existence of protocol model meeting requirements of S^3 computing.

2.1. Definitions Related to S^3 Computing

Node P represents the user authenticated by platform, i.e., the college teacher whose personal and college identities are authenticated. There is a one-one map between teachers and nodes. Nodes have two basic authorities: (1) communicating with arbitrary nodes and the communicational information cannot be maliciously peeked by other nodes. (2) marking an arbitrary node. Marking refers to the description attached to public profile of a node who committed malicious actions. The descriptions can only be made by nodes who communicated with the marked node. If description is proved to be true, then reputation of marked node will be decreased and description will be attached to its public profile for a period of time; if description is proved to be false, then reputation of node made the description will be decreased and this node will be marked by system. This strategy of mutual checking among nodes and restricted intervention made by system is proved to be effective in various social networks as on-line games in [5], recommendation systems in [6] and spam filtering in [7].

Distance d_{ms} in multiset is the number of elements only occurring in one multiset of two. For instance, multisets $S_1 = \{\{v_1\ v_1\ v_3\ \}\}$ and $S_2 = \{\{v_1\ v_2\ \}\}$ can be built based on set $\{v_1\ v_2\ v_3\ \}$ and $\mathrm{d}_{ms}(S_1, S_2\ ) = 3$. This is because v_3 occurs in S_1 1 time, v_2 occurs in S_2 1 time, v_1 occurs in S_1 and S_2 for 2 times and 1 time respectively, and the smaller number 1 is taken. If the occurrence times are represented by vectors, then $\boldsymbol{V}_1 = [2 \quad 0 \quad 1]^{\mathrm{T}}$, $\boldsymbol{V}_2 = [1 \quad 1 \quad 0]^{\mathrm{T}}$ and $\mathrm{d}_{ms} = \|\boldsymbol{V}_1 - \boldsymbol{V}_2\| = \left\|[1 \quad -1 \quad 1]^{\mathrm{T}}\right\| = 1 + |-1| + 1 = 3$.

S^3 candidate is a quadruple $(f, \mathbb{V}, \mathbb{U}, d)$ in which $\mathbb{V}$ is the set containing possible input values, f maps the inputs of n nodes to metric space $(\mathbb{U}, d)$, i.e., $f\colon \mathbb{V}^* \to \mathbb{U}$ and $f(v_1, v_2, \ldots, v_n) = f(v_{\sigma(1)}, v_{\sigma(2)}, \ldots, v_{\sigma(n)})$ holds for any permutation σ of v_1 to v_n. Similar with d_{ms}, d is employed to compute distance between two outputs of f in space $(\mathbb{U}, d)$, i.e., $d(f(v_1, v_2, \ldots, v_n),\ f v_1', v_2',, \ldots, v_n')) \in \mathbb{U}$.

g-Scalability refers to the case in which the spacial, computational and information complexities of distributed computing performed based on protocol model are $\mathcal{O}(g(n)\, poly(\lg(n)))$ where $g\colon \mathbb{N} \to \mathbb{N}$ for each node.

g-Accuracy refers to the case in which the maximal distance between output O_p involving all nodes with standard reputation and the actual result $f(v_1, v_2, \ldots, v_n)$ in $(\mathbb{U}, d)$ meets the following condition:

$$\frac{1}{\Delta(n)} \max_{\text{standard } p} d(f(v_1, v_2, \ldots, v_n), O_p) = \mathcal{O}(\frac{1}{g(n)})$$

Where v_i represents the input of ith node and $\Delta(n)$ is the maximal distance among all distances corresponding to computational results involving n nodes in $(\mathbb{U}, d)$, i.e., :

$$\Delta(n) = \max_{\substack{(x_1 \cdots x_n) \\ (y_1 \cdots y_n)}} d(f(x_1, x_2, \dots, x_n), f(y_1, y_2, \dots, y_n))$$

Accuracy refers to the case in which when coalition $\mathcal{B}$ of low reputation nodes infers the input of standard node P through the output O_1 generated by a distributed computing of input $I = (v_1, v_2, \dots, v_n)$ where $v_1, v_2, \dots, v_n$ are not all the same, there is an output O_2 corresponding to input I' which satisfies conditions that $O_1 = O_2$ holds for $\mathcal{B}$ and $I \neq I'$ holds for P. The probability to infer input of P through O_2 is $\frac{1}{n^\alpha}$ where $\alpha > 1$.

When a protocol model is of g-Scalability, g-Accuracy and accuracy, it is said that the model satisfies S^3 computing.

2.2. Existence of Protocol Model Satisfying S^3 Computing

For convenient reference, the proof of existence of protocol model satisfying S^3 computing is repeated here based on [1]. Assuming $C = (f, \mathbb{V}, \mathbb{U}, d)$ is a S^3 candidate and satisfies conditions that $\mathbb{V}$ is finite, $\Delta(n) = \Omega(n)$, and $\mathrm{d_{ms}}(\mathrm{f(x)}, \mathrm{f(y)}) \leq \mathrm{kd(x, y)}$ where k is a constant. If there is an algorithm $\mathcal{A}$ which locally performs the computation required by f according to the compact representation of multiset of inputs with complexity of $\mathcal{O}(g(n)\, poly(\lg(n)))$ and there is a protocol model $\mathcal{D}$ satisfying S^3 computing, then a protocol model $\mathcal{T}$ which performs the computation required by f exists.

Protocol model $\mathcal{T}$ can be constructed as follows. Converting the input v_p of each node P to compact representation of multiset $\{\{v_p\}\}$. The compact representation ms_p of multisets from all nodes is constructed based on model $\mathcal{T}$. For each node, the output O_p is generated by locally running algorithm $\mathcal{A}$ with input ms_p, i.e., $O_p = f(ms_p)$.

Because complexities of protocol model $\mathcal{D}$ and algorithm $\mathcal{A}$ both are $\mathcal{O}(g(n)\, poly(\lg(n)))$, $\mathcal{T}$ also has this complexity which proves $\mathcal{T}$ is of g-Scalability. When $\mathcal{D}$ is assumed to be of h-Accuracy, h-Accuracy of $\mathcal{T}$ is given by the following:

$$\begin{aligned}
\frac{1}{\Delta(n)} \max_{\text{standard } p} d(f(v_1, v_2, \dots, v_n), O_p) &= \frac{1}{\Delta(n)} \max_{\text{standard } p} d(f(v_1, v_2, \dots, v_n), f(ms_p)) \\
&\leq k \frac{1}{\Delta(n)} \max_{\text{standard } p} d_{ms}(\{\{v_1, v_2, \dots, v_n\}\}, ms_p) \\
&\leq 2k \frac{n}{\Delta(n)} \frac{1}{\Delta_{ms}(n)} \max_{\text{standard } p} d(f(v_1, v_2, \dots, v_n), f(ms_p)) \\
&= 2k \frac{n}{\Delta(n)} \mathcal{O}(\frac{1}{h(n)})
\end{aligned}$$

Namely,

$$\frac{1}{\Delta(n)} \max_{\text{standard } p} d(f(v_1, v_2, \dots, v_n), O_p) = \mathcal{O}(1)\, \mathcal{O}(\frac{1}{h(n)})$$

Privacy of $\mathcal{T}$ is guaranteed by $\mathcal{D}$. Therefore, protocol model satisfying S^3 computing exists. The following section constructs protocol model $\mathcal{LC}$ which adapts learning community of college teachers and satisfies S^3 computing.

3. Protocol Model $\mathcal{LC}$

This section describes the construction of protocol model $\mathcal{LC}$ in details. The general scheme is sketched in subsection 3.1. Subs**ections 3.2 to 3.3 respectively describes the three phases of performing distributed computing based on $\mathcal{LC}$.**

3.1. General Scheme of $\mathcal{LC}$

As depicted in Fig. 1, $\mathcal{LC}$ consists of three phases, namely, Phase 1: construct groups and proxies; Phase 2: generate local aggregation and Phase 3: process aggregation with token.

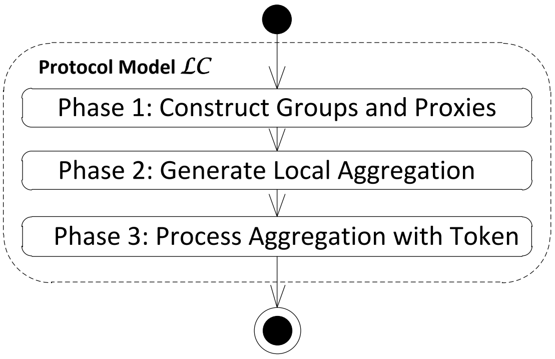
Protocol Model $\mathcal{LC}$
Phase 1: Construct Groups and Proxies
Phase 2: Generate Local Aggregation
Phase 3: Process Aggregation with Token

Figure 1. Three Phases of $\mathcal{LC}$

3.2. Phase 1 of $\mathcal{LC}$: Construct Groups and Proxies

According to [10-11], nodes of low reputation will be randomly distributed among all groups if all n nodes are randomly distributed among groups of size $\sqrt{n}$. For $\mathcal{LC}$, the distribution of nodes is based on learning communities. If the distributed computing does not involve data restricted in some learning communities, then all nodes are distributed as mentioned above; if it does, then the distribution has to be made based on communities. For instance, if there are n_i nodes in the ith community, then n_i nodes will be distributed to group of size $\sqrt{n_i}$ which only contains nodes from ith community. Because Phase 3 of $\mathcal{LC}$ requires an unique group to initiate, this group is constructed based on the reputation levels of nodes, i.e., before the distribution starts, choose $\sqrt{n}$ nodes in the decreasing order of reputation, then the rest $n - \sqrt{n}$ nodes are distributed as mentioned above. This unique group is called group of reputation. The whole procedure is shown in Fig. 2.

Groups generated by distribution form a closed and ordered ring. For an arbitrary node P in any group, P can send messages to all nodes in group containing P or $\mathcal{L}$ nodes in each of next $\mathcal{K}$ groups counting from group of P in the ring. All $\mathcal{K} \cdot \mathcal{L}$ nodes out of group of P to which P can send messages are called proxies of P. The priory factors of proxy selection with respect to P are communication frequencies and reputations. For any node of standard reputation in the next $\mathcal{K}$ groups, if its communication frequencies involving P is $\mathcal{F}$ and its reputation is $\mathcal{R}$, then the value $\mathcal{R}^{\alpha}\ln(\mathcal{F})$ where $\mathcal{F}, \mathcal{R}, \alpha > 1$ determines whether it will be chosen as a proxy of P. Nodes in a specific group among $\mathcal{K}$ groups are ordered in values of $\mathcal{R}^{\alpha}\ln(\mathcal{F})$, and the first $\mathcal{L}$ nodes are chosen as proxies of P in this group. Since $\mathcal{F}$ may have a very large value, $\ln(\mathcal{F})$ reduces its impact. To raise the impact of $\mathcal{R}$, it has a term $\mathcal{R}^{\alpha}$.

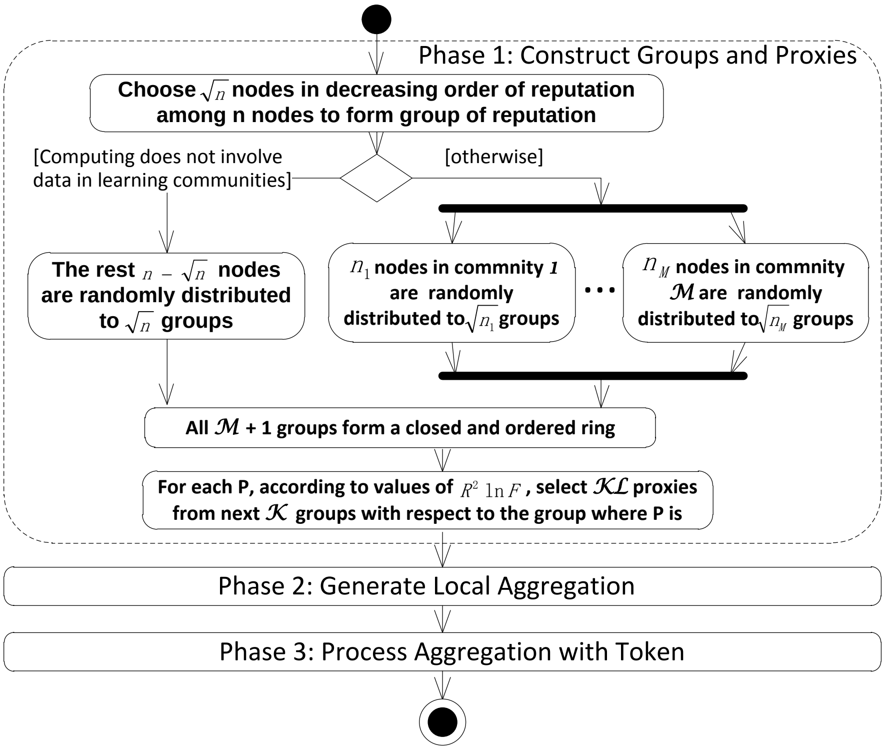
Phase 1: Construct Groups and Proxies
Choose $\sqrt{n}$ nodes in decreasing order of reputation among n nodes to form group of reputation
[Computing does not involve data in learning communities]
[otherwise]
The rest $n - \sqrt{n}$ nodes are randomly distributed to $\sqrt{n}$ groups
n_1 nodes in commnity $\mathcal{1}$ are randomly distributed to $\sqrt{n_1}$ groups
• • •
n_M nodes in commnity $\mathcal{M}$ are randomly distributed to $\sqrt{n_M}$ groups
All $\mathcal{M}$ + 1 groups form a closed and ordered ring
For each P, according to values of $R^2 \ln F$, select $\mathcal{KL}$ proxies from next $\mathcal{K}$ groups with respect to the group where P is
Phase 2: Generate Local Aggregation
Phase 3: Process Aggregation with Token

Figure 2. Phase 1: Construct Groups and Proxies

3.3. Phase 2 of $\mathcal{LC}$: Generate Local Aggregation

When phase 1 ends, for each node P in each group, $\frac{\mathcal{K}\cdot\mathcal{L}-1}{2}$ inputs and $\frac{\mathcal{K}\cdot\mathcal{L}-1}{2}$ inverse inputs are generated. All $\mathcal{K}\cdot\mathcal{L}-1$ generated inputs and one true input of P are randomly sent to $\mathcal{K}\cdot\mathcal{L}$ proxies. [17] describes a similar strategy. The probability of receiving the true input is $\frac{1}{\mathcal{K}\cdot\mathcal{L}}$. When a message is received, proxy first checks whether the received message is valid, i.e., whether $v_i \in V$ holds. If it holds, then aggregate the input, otherwise alarm is triggered. Aggregation is defined to be operator $\oplus$ in space $(\mathbb{U}, \oplus)$ which satisfies the following condition.

$$d(v_1 \oplus v_2,\, v'_1 \oplus v'_2) \le d(v_1,\, v'_1) + d(v_2,\, v'_2)$$

After aggregation, the proxy perform the following computation to check whether the aggregation is valid.

$$d(v_1 \oplus \ldots \oplus v_k,\, v'_1 \oplus \ldots \oplus v'_k) \le d(v_1,\, v'_1) + \cdots + d(v_k,\, v'_k) \le k \cdot \delta_V$$

Where δ_V represents the maximal distance in $\mathbb{V}$ and $k \in \mathbb{N}$. The above formula estimates whether the aggregation belong to space $(\mathbb{U}, d)$, i.e., whether $v_1 \oplus \ldots \oplus v_k \in \mathbb{V}$. The property of operator $\oplus$ implies $\mathbb{V} \subseteq \mathbb{U}$ holds. If the above formula holds for aggregation, then aggregation will be copied and sent to every node in the group; otherwise, alarm is triggered. If the generated input and corresponding reverse input are sent to the same node, then the aggregation of two inputs is null. When the dispatching ends, each node aggregates all received aggregation to generate local aggregation. Fig. 3 describes the whole procedure of Phase 2. Unlike Phase 3 initiated by group of reputation, all nodes in the ring participate in Phase 2.

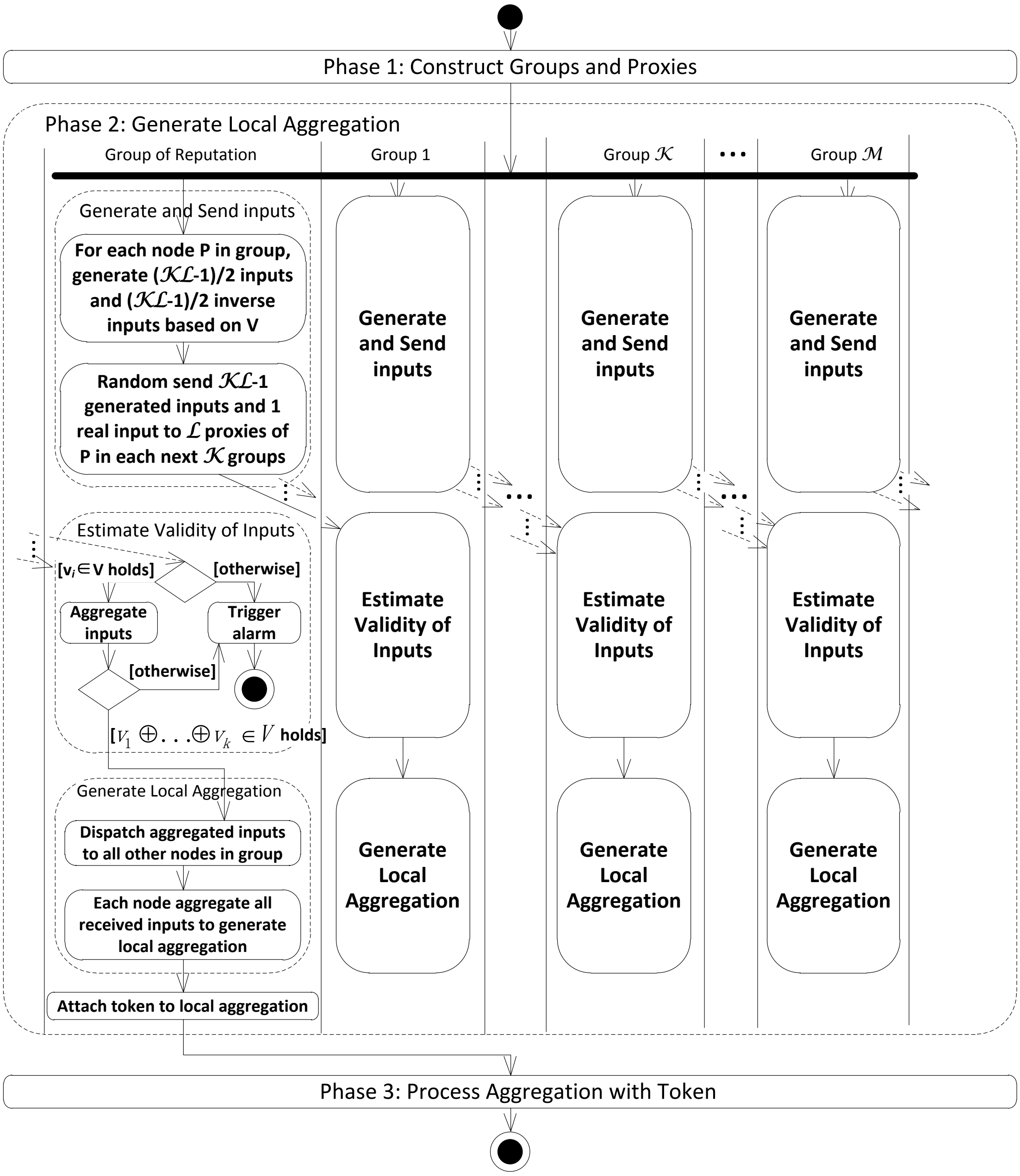
Phase 1: Construct Groups and Proxies
Phase 2: Generate Local Aggregation
Group of Reputation
Group 1
Group $\mathcal{K}$
• • •
Group $\mathcal{M}$
Generate and Send inputs
For each node P in group, generate ($\mathcal{KL}$-1)/2 inputs and ($\mathcal{KL}$-1)/2 inverse inputs based on V
Random send $\mathcal{KL}$-1 generated inputs and 1 real input to $\mathcal{L}$ proxies of P in each next $\mathcal{K}$ groups
Estimate Validity of Inputs
[v_i∈V holds]
[otherwise]
Aggregate inputs
Trigger alarm
[otherwise]
[$v_1 \oplus ... \oplus v_k \in V$ holds]
Generate Local Aggregation
Dispatch aggregated inputs to all other nodes in group
Each node aggregate all received inputs to generate local aggregation
Attach token to local aggregation
Generate and Send inputs
Estimate Validity of Inputs
Generate Local Aggregation
Phase 3: Process Aggregation with Token

Figure 3. Phase 2: Generate Local Aggregation

3.4. Phase 3 of $\mathcal{LC}$: Process Aggregation with Token

The group of reputation generated in Phase 1 starts Phase 3. When Phase 2 terminates, real inputs of all nods are preserved in local aggregations. Phase 3 ensures each node finally obtains all n real inputs by processing aggregations attached by tokens and the computation of f involving all real inputs is locally performed for each node.

Because there are no nodes of low reputation in group of reputation, the local aggregations of nodes in the group all are same. Each node then attaches its local aggregation by a unique token. Then aggregation of token is dispatched to corresponding $\mathcal{K} \cdot \mathcal{L}$ proxies. For each proxy received aggregation of token, if the token is received for the first time, then the proxy aggregate local aggregation with received aggregation and send the resulting aggregation to its $\mathcal{L}$ proxies in neighboring group; if the token is found to have been received, then the proxy checks whether the token is exactly same as before, if same, then broadcasts received aggregation to any other nodes in its group; if not, alarm will be triggered. This procedure is called "Process Aggregation with Token" in Fig. 4. The procedure repeats until each node has obtained inputs of all other $n-1$ nodes.

During all three phases mentioned above, once alarm is triggered, it means some aggregation is maliciously changed by nodes of low reputation and the correctness of final computational result is not guaranteed. Therefore, the whole computation aborts if any alarm is raised. System will find all low reputation nodes involved in the maliciously-changed aggregation and mark all found nodes.

According to [1], protocol model $\mathcal{LC}$ is of $\sqrt{}$-Scalability, $\sqrt{}$-Accuracy and accuracy which means $\mathcal{LC}$ satisfies S^3 computing.

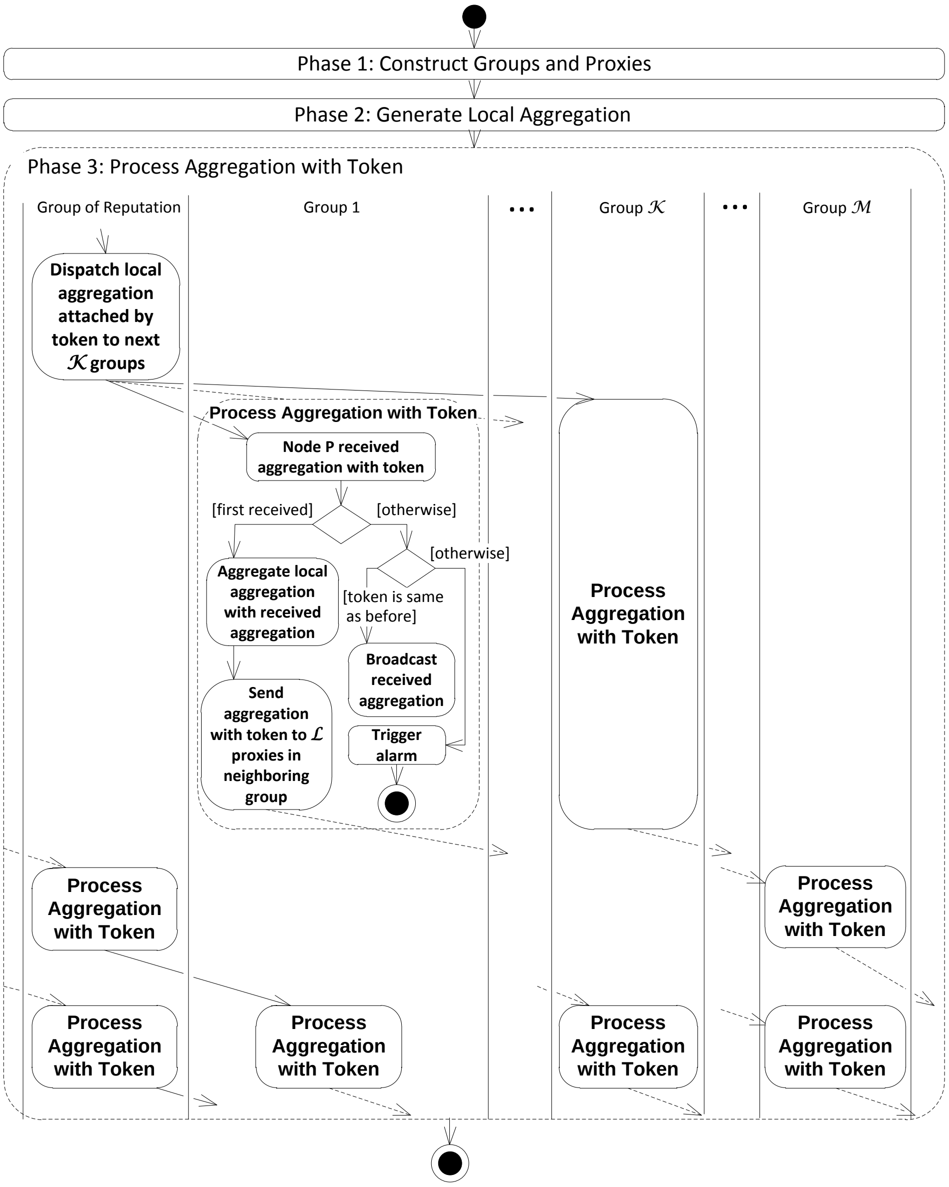
Phase 1: Construct Groups and Proxies
Phase 2: Generate Local Aggregation
Phase 3: Process Aggregation with Token
Group of Reputation
Group 1
• • •
Group $\mathcal{K}$
• • •
Group $\mathcal{M}$
Dispatch local aggregation attached by token to next $\mathcal{K}$ groups
Process Aggregation with Token
Node P received aggregation with token
[first received]
[otherwise]
[otherwise]
Aggregate local aggregation with received aggregation
[token is same as before]
Broadcast received aggregation
Send aggregation with token to $\mathcal{L}$ proxies in neighboring group
Trigger alarm
Process Aggregation with Token
Process Aggregation with Token
Process Aggregation with Token
Process Aggregation with Token
Process Aggregation with Token
Process Aggregation with Token
Process Aggregation with Token

Figure 4. Phase 3: Process Aggregation with Token

4. Conclusion

This paper constructs protocol model $\mathcal{LC}$ which adapts learning community platform of college teachers and satisfies S^3 computing. Through the proposed protocol model, correctness and robustness of distributed computation are guaranteed which makes the social researches performed on learning community platform possible.

Acknowledgment

This research is financially supported by the educational project of Chang Zhou University (Grant No. 2015XSJY06).

References

[1] Giurgiua, R. Guerraouia, K. Huguenina and A. M. Kermarrecb, “Computing in social networks”, Inform. And Computation, vol. 234, **(2014)**, pp. 3-16.

[2] J. Benaloh, “Secret sharing homomorphisms: keeping shares of a secret”, Proceedings of the 6th Annu. Int. Conf. Advances in Cryptology, London, UK, **(1986)**, pp. 251-260.

[3] R. Rivest, A. Shamir, Y. Tauman, “How to share a secret”, Commun. of ACM, vol. 22, no. 11, **(1979)**, pp. 612–613.

[4] A. Yao, “Protocols for secure computations”, Proceedings of the 23rd IEEE Annu. Symp. on Found. of Comput. Sci., Washington, DC, USA, **(1982)**, pp. 160-164.

[5] P. Kabus, W. W. Terpstra, M. Cilia and A. Buchmann, “Addressing cheating in distributed MMOGs”, Proceedings of the 4th Ann. Workshop on Network and Syst. Support for Games, New York, USA, **(2005)**, pp. 1-6.

[6] N. Tran, B. Min and L. J. Li, “Subramanian, Sybil-resilient online content voting”, Proceedings of the 6th USENIX Symp. on Networked Syst. Design and Implementation, Boston, USA, **(2009)**, pp. 15-28.

[7] M. Sirivianos, K. Kim and X. Yang, “SocialFilter: introducing social trust to collaborative spam mitigation”, Proc. of the 30th IEEE Int. Conf. Comput. Commun., Shanghai, China, **(2011)**, pp. 2300-2308.

[8] Z. Galil and M. Yung, “Partitioned encryption and achieving simultaneity by partitioning”, Inf. Process. Lett., vol. 26, no. 2, **(1987)**, pp. 81–88.

[9] I. Gupta, K. Birman, P. Linga, A. Demers and R. V. R. Kelips, “Building an efficient and stable P2P DHT through increased memory and back-ground overhead”, Proceedings of the Second Int. Workshop on Peer-to-Peer Syst., Berkeley, CA, USA , **(2003)**, pp. 160–169.

[10] L. H. Vu, K. Aberer, S. Buchegger and A. Datta, “Enabling secure secret sharing indistributed online social networks”, Proc. of the 25th Annu. Comput. Security Applic. Conf., Honolulu, Hawaii, USA, **(2009)**, pp. 419–428.

[11] D. Angluin, J. Aspnes, Z. Diamadi, M. J. Fischer and R. Peralta, “Computation in networks of passively mobile finite-state sensors”, Distrib. Comput., vol. 4, no.18, **(2006)**, pp. 235–253.

[12] R. Guerraoui, E. Ruppert, “Names Trump Malice: tiny mobile agents can tolerate Byzantine failures”, Proceedings of the 36th Int. Colloq. on Automata, Languages and Programming, Rhodes, Greece, **(2009)**, pp. 484-495.

[13] C. Delporte-Gallet, H. Fauconnier, R. Guerraoui and E. Ruppert, “Secretive birds: privacy in population protocols”, Proceedings of the 11th Int. Conf. Principles of Distributed Syst., Guadeloupe, French West Indies, **(2007)**, pp. 329–342.