ICIC Express Letters
Part B: Applications
Volume **8**, Number **6**, June **2017**

ICIC International ©2017 ISSN 2185-2766
pp. 937–943

TO ACHIEVE OPTIMAL TRUSTWORTHY AGREEMENT IN THE UNRELIABLE MOBILE CLOUD COMPUTING ENVIRONMENT

Shu-Ching Wang, Shih-Chi Tseng and Kuo-Qin Yan*

Chaoyang University of Technology
No. 168, Jifeng East Road, Wufeng District, Taichung 41349, Taiwan
{ scwang; s10314903 }@cyut.edu.tw; *Corresponding author: kqyan@cyut.edu.tw

Received December 2016; accepted March 2017

Abstract. *Network bandwidth and hardware technology are developing rapidly, resulting in the vigorous development of the Internet. Nowadays, Mobile Cloud Computing (MCC) is widely accepted as a concept that can significantly improve the user experience when accessing mobile services. MCC increases the number of user applications on the Internet. Research must be focused on how distributed systems can provide better reliability and fluency. The agreement problem is fundamental to fault-tolerant distributed systems. Reach agreement on a same value in a distributed system, even if certain components in distributed system were failed; the protocols are required so that systems still can be executed fault-freely. In this study, the Trusted Timely Computing Base (TTCB) is used when the message is transmitted. The Trustworthy Agreement Protocol (TAP) for MCC that we proposed can solve the agreement problem with a minimal number of rounds of message exchanges and tolerates a maximal number of allowable malicious and dormant faulty transmission media in the Cluster-based MCC (CMCC). The TAP attempts to solve the agreement problem and makes all faulty-free nodes in the topology of MCC achieve stable results without any influence from faulty components.*

Keywords: Distributed agreement problem, Mobile cloud computing, Fault tolerant, Trusted timely computing base

1. Introduction.

MCC at its simplest refers to an infrastructure where both the data storage and the data processing happen outside of the mobile device [4]. As MCC has become increasingly popular, network topology has trended toward wireless connectivity, thus providing enhanced support for MCC [6]. This technological trend has greatly encouraged distributed system design and support for cloud nodes [2]. The "cluster" has attracted significant attention recently because it requires less infrastructure, it can be deployed quickly, and it can automatically adapt to changes in topology. Therefore, the structure of a cluster can suit military communication, emergency disaster rescue operations, and law enforcement [2], and be used to the cloud-computing technology of MCC [2].

In order to enhance the capability of fault tolerance and ensure network security, it is necessary to provide a stable mobile cloud service environment. However, the reliability of cloud nodes is one of the most important aspects in MCC [5]. In order to provide a reliable CMCC, a mechanism to allow a set of cloud nodes to agree on an agreement value is required.

The Byzantine Agreement (BA) problem introduced by Lamport et al. [3] is one of the fundamental problems in distributed computing. The definition of the problem is to make the fault-free nodes in an n-node distributed system reach agreement. The source node chooses an initial value to start with, and communicates to each other by exchanging messages. A group of nodes is referred to make an agreement if it satisfies the following conditions [3].

(Agreement): All fault-free nodes agree on a common value.

(Validity): If the initial value of fault-free source node n_s is v_s, then all fault-free nodes shall agree on the value v_s.

In an agreement problem, many cases are based on the assumption of node failure in a fail-safe network [3]. Based on this assumption, a Transmission Medium (TM) fault is treated as a node fault, whatever the fault-freeness of an innocent node is, so an innocent node does not involve agreement [9].

Actually, the symptom of a faulty TM can be classified into two types: dormant (such as crash, stuck-at, or delay) and malicious. A dormant faulty TM always can be identified by the receiver if the transmitted message was encoded appropriately (i.e., by NRZ-code, Manchester code [7]) before transmission. On the other hand, the malicious faulty TMs are unpredictable. In this study, the agreement problem is revisited to enlarge the fault tolerant capability by allowing both dormant and malicious faulty TMs to exist in the system simultaneously.

Therefore, in this study, the agreement problem is revisited with the assumption of TM failure on dormant and malicious faults in the CMCC while message transmissions may be disturbed by some faults, break down, stuck-at, noise or an intruder. The proposed protocol TAP, can tolerate d dormant faulty TMs and m malicious faulty TMs simultaneously existing in the CMCC to reach agreement. In addition, the protocol requires only two rounds of message exchange.

The rest of this paper is organized as follows. The related work is shown in Section 2. The proposed protocol TAP will be illustrated in Section 3. In Section 4, an example of executing the proposed protocol is given. Section 5 is responsible for proving the fault-freeness and complexity of TAP. Finally, Section 6 gives conclusions of this research.

2. Related Work.

The design and development of the trustworthy agreement protocol have several requirements that must be considered. Therefore, the structure of CMCC and the security technology will be discussed in this section.

The MCC would also be based on the basic cloud computing concepts. MCC combining the mobile devices and cloud computing to create a new infrastructure, MCC can perform the heavy lifting of computing-intensive tasks and storing massive amounts of data [2]. Currently, the cluster cloud is a more practical kind of cloud computing. A cluster of multiple cloud nodes in a cluster cooperates to achieve some objectives [2]. CMCC consists of a set of loosely or tightly connected cloud nodes that work together so that, in many respects, they can be viewed as a single system. The components of a cloud cluster are usually connected to each other through fast LANs with each cloud node. All cloud nodes of CMCC are usually deployed to offer improved performance and availability compared to a single computer, while typically are much more cost-effective than single computers of comparable speed or availability. However, in CMCC, from the aspects of mobile computing and cloud computing, mobile cloud computing is a combination of both technologies, the development of distributed, grid and centralized algorithms, as well as prospects for broad application [5]. The CMCC is shown in Figure 1.

In a CMCC, the nodes are interconnected. Reach agreement on a same value in a distributed system, even if certain TMs in distributed system were failed (inner damage or outer intruder); hence, the protocols are required so that systems still can be executed fault-freely. In this study the TTCB is used when the message is transmitted [8]. There are two characteristics of TTCB: security and synchronization. The TTCB system structure is shown in Figure 2. Payload System is composed of useful software in the Host and Payload Network offers the ways that contact with each Host [8]. Because the TTCB is secure, nodes can receive the same result through the TTCB.

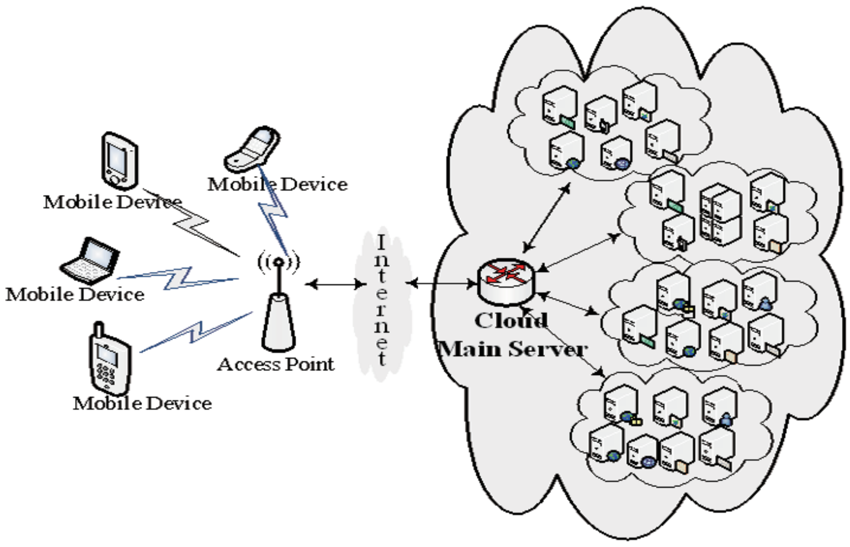


Figure 1. Cluster-based MCC [5]

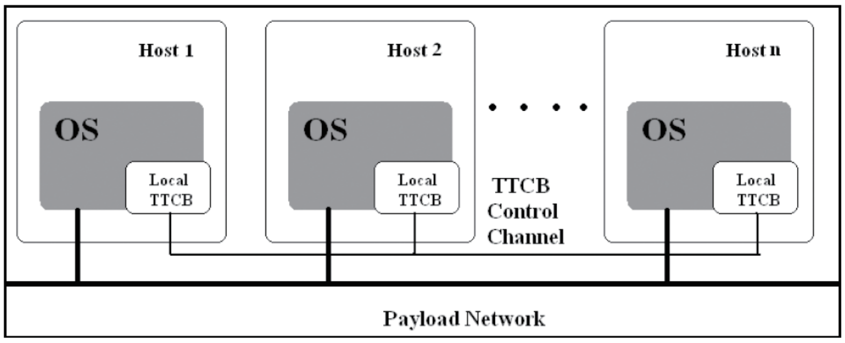


Figure 2. TTCB system structure [8]

In this study, all nodes of a distributed system can reach agreement even if some TMs are faulty to interfere from some noise hijacker or arbitrary behavior. Then, the fault-tolerance capacity of the system is enhanced.

3. **The Proposed Protocol.** This study proposes a new protocol TAP to solve the agreement problem of faulty TMs that may send wrong messages to influence the system to achieve agreement in a CMCC. The proposed protocol TAP consists of two phases, the *message exchange phase* and *decision making phase*. Moreover, TAP only needs two rounds of message exchanges to solve the agreement problem.

In the first round of the *message exchange phase*, the source node n_s multicasts its initial value v_s through TMs by TTCB. And then, each node stores the received value in root of message-gathering tree (mg-tree) [9]. The mg-tree is a tree structure which is used to store the received messages. In the second round, each node n_i acts as the sender, sending the value v_s (received from source node n_s) to other nodes by TTCB. However, the receiver can always detect the message(s) through dormant faulty components if the protocol TAP appropriately encodes a transmitted message by using Manchester code [7]. Hence, if the messages pass through any dormant faulty TMs, then λ will be stored in mg-tree of receiver. In *decision making phase*, in order to reduce the influence from the faulty components, a majority value is taken from all nodes in the same cluster to set the majority value at level 2. Finally, the agreement among all nodes will be achieved. The proposed protocol TAP is presented in Figure 3.

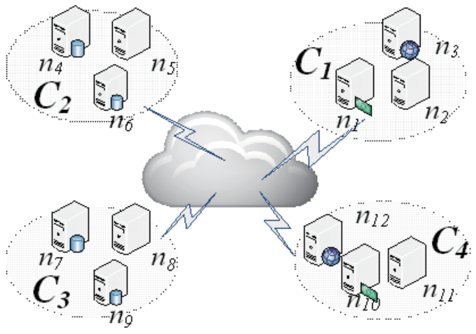
TAP protocol (for node n_s with initial value v_s)	
Message Exchange Phase	
Round 1:	The source node sends its value (v_s) to other nodes by TTCB; each receiver node obtains the value and stores the received value in the root of its mg-tree. If the cluster-disjoint path from source node to destination cluster passes through any dormant faulty transmission media, then λ is stored.
Round 2:	Each node transmits the values at the root in its mg-tree to each cluster's nodes by TTCB. If the cluster-disjoint path from source node to destination cluster passes through any dormant faulty transmission media, then λ is stored. Each receiver node takes a majority on its received messages and stores the majority value in the corresponding vertices at level 2 of its mg-tree.
Decision Making Phase	
Step 1: (1) Take the majority value of V_i in mg-tree. (2) Each λ value is ignored and does not join to the majority. (3) If the majority value does not exist, then set the # (it will be ignored and does not join to the majority in Step 2 at level 2). else set the majority value $\{0, 1\}$ at level 2. **Step 2:** (1) Take a majority value of mg-tree, each λ value is ignored and does not join to the majority. (2) If the majority value does not exist and v_{ki} of V_i is equal to v_i in its mg-tree at level 2, then set $DEC_i = \text{NOT}(v_i)$, // rule 1 else set $DEC_i = v_s$ if the majority value is v_s in the mg-tree. // rule 2 Otherwise set $DEC_i = \phi$. // rule 3	

FIGURE 3. The proposed protocol TAP

4. **An Example of TAP Executed.** An example of executed TAP is shown in Figure 4. Figure 4(a) shows 12 nodes, 4 clusters in a CMCC. Cluster C_1 includes the nodes n_1, n_2 and n_3. Cluster C_2 includes the nodes n_4, n_5 and n_6. Cluster C_3 includes the nodes n_7, n_8 and n_9. Cluster C_4 includes the nodes n_{10}, n_{11} and n_{12}. The initial value of the source node n_s (n_3) is 1. Figure 4(b) describes the values v_s received by each node by TCCB in the first round of *message exchange phase*. The received values are changed maliciously in n_4, n_5 and n_6 due to the fact that the TMs are malicious faults. The received values of n_{10} and n_{12} are interfered within dormant faulty TMs, and λ will be set to n_{10} and n_{12}.

In the end of *message exchange phase*, each node sends the value received in the first round by TTCB to all nodes as shown in Figure 4(c). Since the TMs between C_1 and C_2 are malicious, the values stored in n_i's mg-tree ($1 \leq i \leq 6$) are changed maliciously. The TMs between n_1 and n_{10}, n_3 and n_{12} are dormant, the values λs are stored in n_i's mg-tree ($i = 1$ to 3 and 10 to 12). Similarly, the TMs between n_4 and n_8, n_5 and n_7 are dormant, and the values λs stored in n_i's mg-tree ($i = 4$ to 9) are also interfered within dormant faults. The TMs between n_7 and n_{11}, n_8 and n_{10} are dormant faults, and the values λs stored in n_i's mg-tree ($i = 7$ to 12) are interfered within dormant faults.

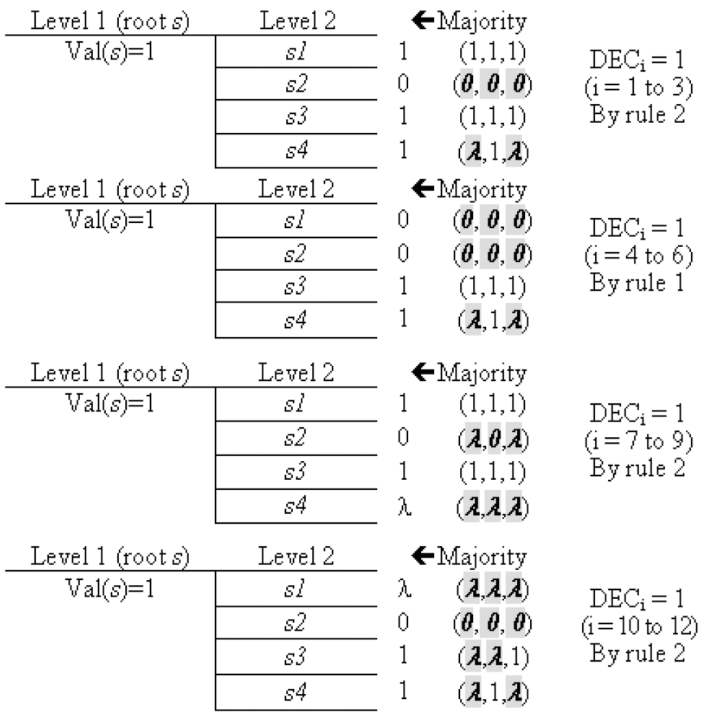
The *message exchange phase* has completed after two rounds by TAP. In order to reduce the fault-free values of the TMs that were interfered within dormant or malicious faults, each node takes majority in Step 1 of *decision making phase*. By the first of Step 1, each node takes the majority on the values received from a cluster, and stores its majority values in mg-tree at level 2. Then each node takes the majority value from the mg-tree



(a) A 4-cluster MCC with dormant and malicious faulty TMs

	Level 1 (Root s)
Node n_1	1
Node n_2	1
Node n_3	1
Node n_4	***0***
Node n_5	***0***
Node n_6	***0***
Node n_7	1
Node n_8	1
Node n_9	1
Node n_{10}	$\boldsymbol{\lambda}$
Node n_{11}	1
Node n_{12}	$\boldsymbol{\lambda}$

(b) The first round of message exchange



Level 1 (root *s*)	Level 2	←Majority		
Val(*s*)=1	*s1*	1	(1,1,1)	DEC$_i$ = 1 (i = 1 to 3) By rule 2
	s2	0	(***0***, ***0***, ***0***)	
	s3	1	(1,1,1)	
	s4	1	($\boldsymbol{\lambda}$,1,$\boldsymbol{\lambda}$)	

Level 1 (root *s*)	Level 2	←Majority		
Val(*s*)=1	*s1*	0	(***0***, ***0***, ***0***)	DEC$_i$ = 1 (i = 4 to 6) By rule 1
	s2	0	(***0***, ***0***, ***0***)	
	s3	1	(1,1,1)	
	s4	1	($\boldsymbol{\lambda}$,1,$\boldsymbol{\lambda}$)	

Level 1 (root *s*)	Level 2	←Majority		
Val(*s*)=1	*s1*	1	(1,1,1)	DEC$_i$ = 1 (i = 7 to 9) By rule 2
	s2	0	($\boldsymbol{\lambda}$,***0***,$\boldsymbol{\lambda}$)	
	s3	1	(1,1,1)	
	s4	λ	($\boldsymbol{\lambda}$,$\boldsymbol{\lambda}$,$\boldsymbol{\lambda}$)	

Level 1 (root *s*)	Level 2	←Majority		
Val(*s*)=1	*s1*	λ	($\boldsymbol{\lambda}$,$\boldsymbol{\lambda}$,$\boldsymbol{\lambda}$)	DEC$_i$ = 1 (i = 10 to 12) By rule 2
	s2	0	(***0***, ***0***, ***0***)	
	s3	1	($\boldsymbol{\lambda}$,$\boldsymbol{\lambda}$,1)	
	s4	1	($\boldsymbol{\lambda}$,1,$\boldsymbol{\lambda}$)	

(c) The mg-tree of each node in the second round of message exchange

FIGURE 4. An example of TAP execution

at level 2 in the end of Step 2. Let DEC_i be defined as the value chosen by node n_i to be agreed on with other nodes, each node agrees on a common agreement shown in Figure 4.

5. The Fault-Freeness and Complexity of TAP.

To cope with the CMCC on dual failure modes, the proposed protocol TAP can tolerate d dormant and m malicious faulty TMs existing simultaneously in an n nodes and C clusters of MCC to agree on a common agreement. The lemmas and theorems are used to prove the fault-freeness of TAP.

Lemma 5.1. *Let the initial value of the sender node n_i be v_i. By using TAP, the destination cluster's nodes can receive the value v_i from the sender node n_i by TTCB if $m_{xy} \le \lceil(\mathrm{TM}_{xy} - d_{xy})/2\rceil - 1$ and $d_{xy} \le \mathrm{TM}_{xy} - 1$, where m_{xy} is the total number of allowable malicious faulty TMs between Cluster C_x and C_y, d_{xy} is the total number of allowable dormant faulty TMs between Cluster C_x and C_y, and TM_{xy} is the total number of TMs between Cluster C_x and C_y.*

Proof: By using TAP, the sender node can transmit its value to the destination cluster's nodes through TM_{xy} cluster-disjoint paths. According to the assumption of $m_{xy} \le \lceil(\mathrm{TM}_{xy} - d_{xy})/2\rceil - 1$ and $d_{xy} \le \mathrm{TM}_{xy} - 1$, the nodes in the destination cluster, in the worst case, can get $\mathrm{TM}_{xy} - d_{xy}$ values from the sender node. Since $m_{xy} \le \lceil(\mathrm{TM}_{xy} - d_{xy})/2\rceil - 1$ and $d_{xy} \le \mathrm{TM}_{xy} - 1$, the majority can be taken on these $\mathrm{TM}_{xy} - d_{xy}$ values and let each of the nodes in the destination cluster get the value v_s.

Lemma 5.2. *The decision value DEC_i = majority value.*

Proof: Lemma 5.2 is proven by the definition of the agreement problem.

Theorem 5.1. *Protocol TAP is valid.*

Proof: According to Lemmas 5.1 and 5.2, the validity of TAP is confirmed.

Theorem 5.2. *Protocol TAP can make each fault-free node agree on a common agreement.*

Proof: If a node agrees on value Z (where Z $= v_i = v_s$, and $1 \le i \le n$ by Lemma 5.2), all nodes should agree on value Z.

Theorem 5.3. *The amount of information exchange by TAP is* $\mathrm{O}(n)$.

Proof: In the first round, every node receives one initial value from the source node. In the end of message exchange phase, n values are received from the other $(n-1)$ nodes in the network; hence, the total number of message exchange is $1 + (n-1) = n$. The result implies that the complexity of information exchange is $\mathrm{O}(n)$.

Theorem 5.4. *One round of message exchange cannot solve the agreement problem.*

Proof: [Part A]. Message exchange is necessary. A node cannot derive whether or not a disagreeable value exists in other nodes without message exchanging. Therefore, agreement problem cannot be implemented.

[Part B]. One round of message exchange is not enough to solve agreement problem. If node n_i of Cluster C_x is connected with node n_m of Cluster C_y by faulty TM, node n_i may not know the initial value in node n_m by using only one round of message exchange. Hence, it is possible to reach an agreement by using one round of message exchange.

Theorem 5.5. *If the total number of faulty TMs is f and $f > \lceil c/2\rceil - 1$, where $m_{xy} \le \lceil(\mathrm{TM}_{xy} - d_{xy})/2\rceil - 1$, it is also impossible to reach an agreement.*

Proof: When $f > \lceil c/2\rceil - 1$, then each cluster has at least one TM connecting with other cluster. It is possible that there are more malicious faulty TMs than fault-free TMs between Cluster C_x and C_y. Whatever the number of rounds of message exchange, these nodes between Cluster C_x and C_y will always be confused by the messages transferred through those malicious faulty TMs. Similarly, all of TMs between Cluster C_x and C_y might be dormant fault if possible, then regardless of the number of rounds of message exchange, these nodes between cluster C_x and C_y will not receive any non-λ's. However, the decision making by these nodes may conflict with other nodes, agreement problem is solved impossibly, and Theorem 5.5 is proven.

Theorem 5.6. *TAP can tolerate the maximum number of faulty TMs in a CMCC.*

Proof: The theorem has been proven by Theorems 5.2, 5.3, 5.4 and 5.5.

6. Conclusions.

The agreement problem is a fundamental problem in the distributed environment [3]. The problem has been studied by various kinds of network model in the past [9]. According to previous studies, the network topology plays an important role in this problem [9]. Therefore, in this study, the agreement problem in CMCC is revisited. The trustworthy agreement problem is redefined by TAP protocol within TTCB in a CMCC and can achieve a common value with two rounds of message exchanges.

That is, the TAP has the following features.

- The TAP can solve the agreement problem in a CMCC.
- The TAP allows the design of reliable communication using the Trusted Timely Computing Base (TTCB).
- The TAP can solve the agreement problem by the minimum number of rounds of message exchanges.
- The TAP increases the fault tolerance capability by allowing for malicious faulty TMs.

Merely considering component faults in the agreement problem is insufficient for the highly reliable distributed system of a CMCC. A related closely problem is called the Fault Diagnosis Agreement (FDA) problem [1]. The objective of solving the FDA problem is to make each fault-free node can detect or locate the common set of faulty components in the distributed system. Therefore, solving the FDA problem for the highly reliable distributed system underlying topology of CMCC is included in our future work.

REFERENCES

[1] M. L. Chiang, S. C. Wang and L. Y. Tseng, An early fault diagnosis agreement under hybrid fault mode, *Expert Systems with Applications*, vol.36, no.3, pp.5039-5050, 2009.

[2] A. Khan, M. Othman, S. Madani and S. Khan, A survey of mobile cloud computing application models, *IEEE Communications Surveys & Tutorials*, vol.16, no.1, pp.393-413, 2014.

[3] L. Lamport, R. Shostak and M. Pease, The Byzantine general problem, *ACM Trans. Programming Languages and Systems*, vol.4, no.3, pp.382-401, 1982.

[4] V. Lawson, V. Kumar and L. Ramaswamy, Mobile cloud enabled sensor services: Opportunities, challenges and approaches, *Proc. of IEEE International Conference on Mobile Services*, pp.292-297, 2015.

[5] G. Mukesh and S. Sukhwinder, Mobile cloud computing, *International Journal of Enhanced Research in Science Technology & Engineering*, vol.3, no.4, pp.517-521, 2014.

[6] F. Niroshinie, W. L. Seng and R. Wenny, Mobile cloud computing: A survy, *Future Generation Computer Systems*, vol.29, no.1, pp.84-106, 2013.

[7] C. A. Sunshine, *Computer Network Architectures and Protocols*, Springer Science & Business Media, 2013.

[8] P. Veríssimo and A. Casimiro, The timely computing base model and architecture, *IEEE Trans. Computers*, vol.51, no.8, pp.916-930, 2002.

[9] S. C. Wang, K. Q. Yan, C. L. Ho and S. S. Wang, The optimal generalized Byzantine agreement in cluster-based wireless sensor networks, *Computer Standards & Interfaces*, vol.36, no.5, pp.821-830, 2014.