Weak Consistency Model in Distributed Systems Using Hierarchical Colored Petri Net

Mortaza Abbaszadeh[1*], Saeed Saeedvand[2]

[1] Departement of Computer Science, Ilkhchi Branch, Islamic Azad University, Ilkhchi, Iran.
[2] Computer Engineering Department, Faculty of Electrical and Computer Engineering, University of Tabriz, Tabriz, Iran.

*Corresponding author. Email: abbaszadeh@iauil.ac.ir

Manuscript submitted January 10, 2017; accepted April 25, 2017.
doi: 10.17706/jcp.13.2.236-243

Abstract: With regard to recent developments and wide application of distributed systems, keeping consistency of data has been considered as a serious challenge in these systems. Colored Petri Net (CPN) has high capacity in terms of modeling various algorithms and proving them mathematically. Also proving the presented models has great importance. The importance of keeping consistency at distributed systems at different levels always has been known. Therefore in this research, for first time a hierarchical model for weak consistency along with UTC global time in CPN tools has been presented. The presented model is proved and implemented by using a simulator presented in the CPN tools. In this study, it has been shown that how our method modeled and coded by ML language for distributed systems so that an acceptable level of the weak consistency at distributed systems is obtained.

Key words: Weak consistency, colored petri net, distributed systems, CPN tools.

1. Introduction

In distributed systems, services are considered as a structure and base in various computer systems. We know that, with regard to distributed systems, transferring transactions among systems with various distances is time-consuming in a distributed system. Therefore, keeping consistency in the databases and the order of transactions execution in each distributed execution of the processing are important issues investigated and studied. Although there are some constraints and challenges in distributed systems, they should be consistent. The most ideal mode for distributed systems is data consistency. In reading and writing operations of all data, all distributed systems must be considered similarly (Strict Consistency)[1], but executing strict consistency is impossible due to data delay in transferring and sending items. Therefore, various methods have been presented to keep the data consistency in distributed systems such as causal consistency [2]-[5], which previously we provided a model for causal consistency at distributed systems using hierarchical coloured petri net [6]. In this algorithm, operations executed in distributed systems are followed with a special order such as sequential consistency model [7], weak consistency model [8], FIFO or PRAM [7]. Weak consistency model is one of the most important algorithms among mentioned algorithms [9]. Weak consistency algorithm is taken into account as a model for consistency of database. This consistency method presents the concept of database synchronization and a set of reading and writing operations by locating distributed processes in a uniform execution unit. Since there are various requirements in consistency in distributed systems, weak consistency provides an appropriate level of consistency. It should be mentioned that latency must be taken into account with consistency [10].

Petri nets [11]-[13] are graphical tools to formally describe concurrent activities and to analyze them in terms of mathematical proof. Petri nets have capability of advance simulation for modeling algorithms by using programming written into ML artificial intelligence language [14]. Hence, in this study, a hierarchical model has been presented for modeling weak consistency operations, and then it has been proved and implemented in CPN-TOOLS modeling environment. In a research, Gregor and his colleagues presented a distributed model for B-tree algorithm with weak consistency [15]. In the presented model, indexing data structure has been considered to obtain it by using B-tree method, so they presented a model for distributed and parallel execution in cloud computing. In their presented algorithm, weak consistency has been used to update data in decentralized B-tree structure. In this research, it has been shown that many distributed processors can be managed with trivial repetition of internal nodes by using weak consistency. Zhenhai and his colleague presented weak consistency of wavelet estimator for p-mixing errors [16]. In this study, estimation of semi-parametric regression model continuously used in statistical models has been modeled. Weak consistency has been used in p-mixing position to obtain estimations. In another research, Peter Bailis and his colleagues considered data management [17]. In this study, the features of data consistency have been investigated in storing distributed data, and weak consistency has been taken into account in this study.

The structure of this study is as follows. In the first section, colored petri net has been investigated and studied. In the second section, weak consistency has been reviewed by presenting a simple example. The model presented for weak consistency is investigated in the third section. In the fourth section, the pages of distributed systems have been explained. Processors synchronization is explained in the fifth section. In the sixth section, analyzing state space is demonstrated. Finally and in the seventh section, conclusion is presented.

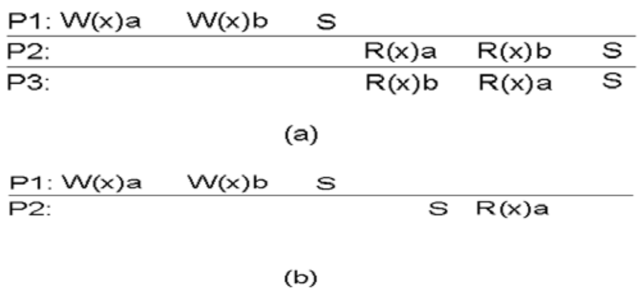
2. Colored Petri Net

Petri net is a graphical tool to formally describe activities in concurrent systems and to analyze them [11]-[13]. Petri net involves four elements; place, transition, arc and token. These elements show various and possible positions of the system. Transitions are formed in the shape of rectangle, and they show system's activities. A transition is active when its input places have tokens. When a mighty transition fires, a token is reduced from input place, and a token is added to output place. A transition is selected randomly. Directional arcs connect the places to transitions and vice-versa. Colored Petri Nets have been developed from classic petri nets, and they have capability of programming with ML language. ML is a programming language for artificial intelligence. Its combination with colored petri net modeling has created much capability to provide recursive functions and various commands in the edge of model. Multi-set marking and operators can be executed through using colored petri net [18]. CPN is one of the best developed tools for modeling and verification of models [18]-[20]. Generally, CPN model is a formal model, and it is considered as a language of mathematic definition in terms of syntax and semantics.

3. Weak Consistency Model

Weak consistency model [8], [21] is one of the strongest consistency models used in executing parallel and programming. This model can be executed in all models that are weaker than sequential consistency. In weak consistency, writings are synchronized by a processor in all processes and other processors by considering global time like UTC, and they are observed by a process and in each system for reading. In weak consistency model, when the value of a variable changes in a process, it is not necessary to inform all. In strict consistency model, change is necessary, but in our model, synchronization is considered to apply changes when it is necessary. This model locates a set of reading and writing operations in a uniform execution unit, and provides a level of consistency.

For example, in part (a), P1 continuously performs writes, and then synchronization is performed. In fact, the last value of x is b. since data synchronization is not required in other processes, they have not synchronized, and they may observe the written value of x variable in reading data. In section (b), since P1 has firstly performed synchronization after writing, and then P2 has performed synchronization, the latest changes (b value) are observed. The read value is a, and weak consistency has been contradicted. A model has been presented in distributed systems for weak consistency, and hypotheses have been considered in CPN. According to these hypotheses, distributed data and operations are executed. Operations are considered with a local ID and a global ID to determine the order of executing commands.



```
P1: W(x)a    W(x)b    S
P2:                        R(x)a   R(x)b   S
P3:                        R(x)b   R(x)a   S

                    (a)

P1: W(x)a    W(x)b    S
P2:                              S   R(x)a

                    (b)
```

Fig. 1. (a) An example for weak mode, and (b) an example for contradicting weak mode.

4. The Presented Model for Weak Consistency

In the presented model, three processes have been designed to demonstrate the related operations in weak consistency (P1, P2, P3). Since the model is hierarchical, a top page has been designed to create a connection between them.

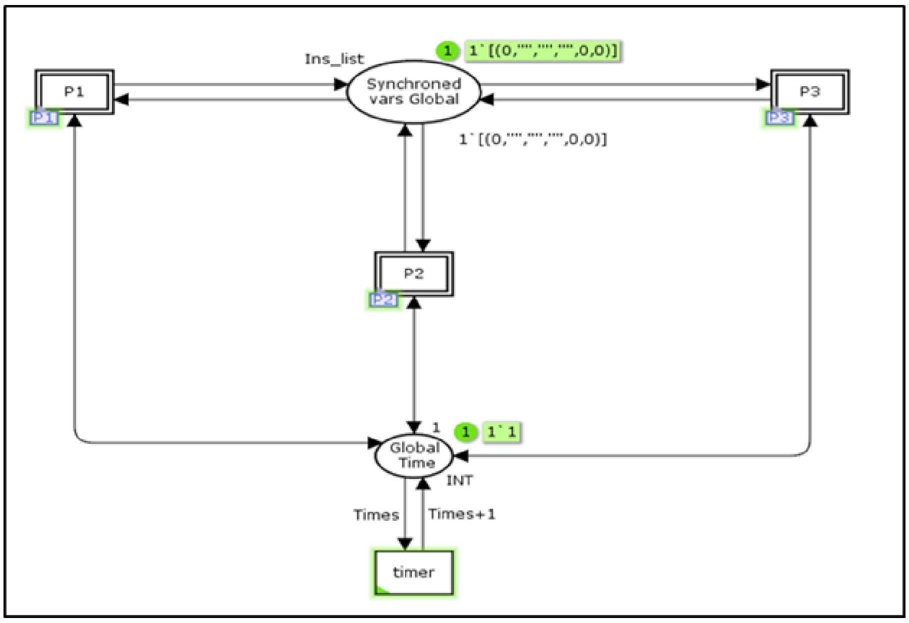


Fig. 2. High level of presented hierarchical model.

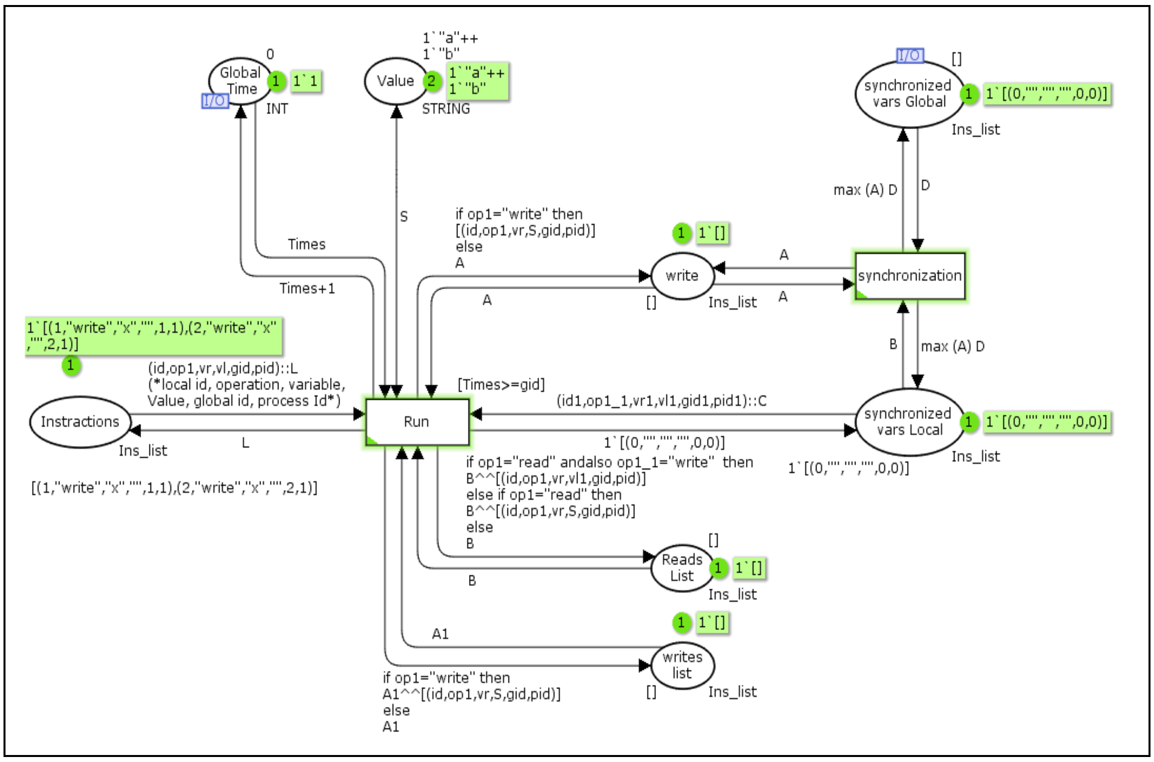
As it can be observed in Fig. 2, hierarchical transitions of P1, P2 and /p3 have been created to show three distributed systems. This transition is connected to a place of "global time" (to determine global time). It should be mentioned that this place is controlled by a timer, and its value increases. Designed processes are connected to a place called "synchronized vars global" in addition to "global time". All synchronized global variables are located in that place.

5. Pages Related to Distributed Processors

Since the model has been hierarchically designed, these three processes are similar, and they are separated by an identification number. In fact, the differences of pages lie in the place of instructions. Each process has its own commands, so in the presented model, just one of the pages showing a distributed system will be explained (Fig. 3).

As it can be observed in Fig. 3, there is a transition named run, and it is considered as the basis of model. Input data are obtained from processes according to input edges, commands and functions. Before detecting the operations of "read" and "write", the following procedures are considered.

- If the operation is related to write, then it binds one of the values of "a" or "b" showing the value by using "value" place, and adds a list of products to the third field. Then, it writes it in "fire" place (that is, it fires it). This cursor has been considered as a local memory for processes. "run" transition inserts all write operations in the place of "writes list", and reads can use it in next steps. If the input command is "read", then various states will occur.
- If a process reads data from other processes, it firstly refers to "synchronized vars global" place. If there are data in that place, then it reads data (it can be manually determined that whether reading has been performed in the form of synchronization, or local data written by the process can be read).
- If a process reads synchronized data, there will be a state in which all writes are located in "synchronized vars global" place by all processes. Before synchronizing them, the items that have not been previously placed in "read list" (or they have not been read) can be read. In the next state, "read process" command can read data from local memory, and it can read data written by other unsorted processes. "a" or "b" values can be changed or added. In Fig. 3, commands dedicated to each process have not been demonstrated.



Global Time
0
1`1
I/O
INT
Value
1`"a"++ 1`"b"
STRING
synchronized vars Global
I/O
[]
1`[(0,"","","",0,0)]
Ins_list
max (A) D
D
S
Times
Times+1
if op1="write" then [(id,op1,vr,S,gid,pid)] else A
1`[]
write
A
[]
Ins_list
A
A
synchronization
B
max (A) D
1`[(1,"write","x","",1,1),(2,"write","x","",2,1)]
(id,op1,vr,vl,gid,pid)::L (*local id, operation, variable, Value, global id, process Id*)
[Times>=gid]
(id1,op1_1,vr1,vl1,gid1,pid1)::C
synchronized vars Local
1`[(0,"","","",0,0)]
Ins_list
Instractions
Ins_list
L
Run
1`[(0,"","","",0,0)]
1`[(0,"","","",0,0)]
[(1,"write","x","",1,1),(2,"write","x","",2,1)]
if op1="read" andalso op1_1="write" then B^^[(id,op1,vr,vl1,gid,pid)] else if op1="read" then B^^[(id,op1,vr,S,gid,pid)] else B
[]
Reads List
1`[]
Ins_list
B
A1
1`[]
writes list
[]
Ins_list
if op1="write" then A1^^[(id,op1,vr,S,gid,pid)] else A1

Fig. 3. Designed CPN model for one of distributed systems.

In Fig. 4, it has been shown that the example of Fig. 1 has been presented in instructions place related to processes, and a list of products has been considered.

- The first field is considered as ID with local commands orders in the process
- The second field is considered as an operation obtaining two values of "read" and "write".
- The third field is considered as a variable (for example, x has been considered here).
- The fourth field is a place for value of the related variable (here, it can receive the values of "a" or "b" from "value" place).

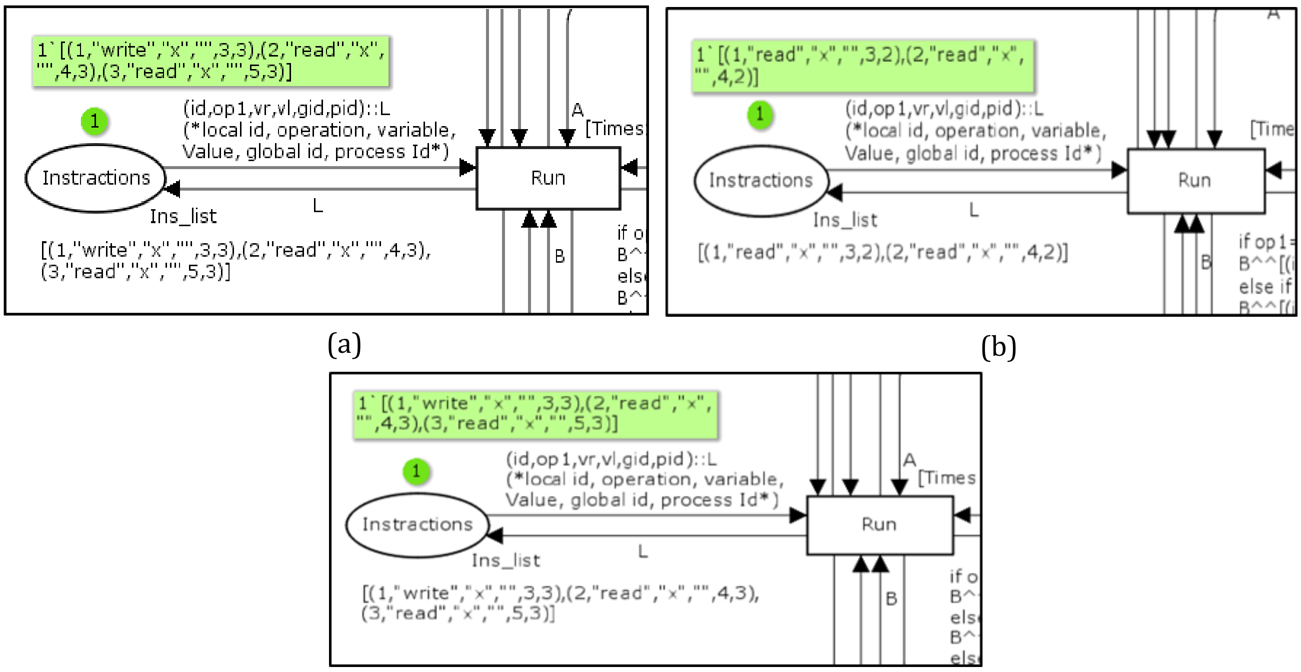

1`[(1,"write","x","",3,3),(2,"read","x","",4,3),(3,"read","x","",5,3)]
(id,op1,vr,vl,gid,pid)::L
(*local id, operation, variable, Value, global id, process Id*)
Instractions
Run
Ins_list
L
A
B
[(1,"write","x","",3,3),(2,"read","x","",4,3),(3,"read","x","",5,3)]
(a)
1`[(1,"read","x","",3,2),(2,"read","x","",4,2)]
[(1,"read","x","",3,2),(2,"read","x","",4,2)]
(b)
(c)

Fig. 4. The section related to the processes located in each distributed system and their initial markings. (Three different distributed systems as a, b, c labels).

6. Synchronization

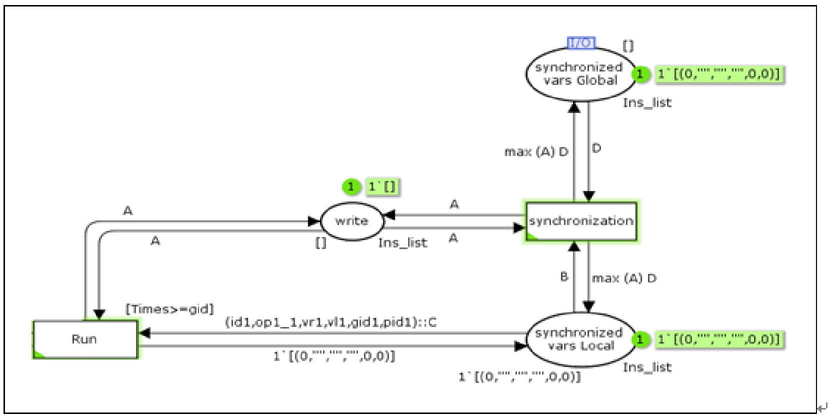


Fig. 5. Synchronization part of the model.

In this model, a transition named "synchronization" has been created and considered, and synchronization of other processes data is performed. When a process performs synchronization, it fires all its "writes" to "synchronized vars global" place. These operations are performed according to predetermined time sequence (TS), or they are updated on the basis of that sequence (global). Also, transition performs synchronization operations of its own data. In this operation, a place named "synchronized vars global" is considered in the model. After firing this transition, all writes performed by other processes are placed in this place according to global time sequence. The main transition of run uses it when it executes read commands (Fig. 5).

6.1. Describing Functions, Color Sets and Variables

In this model, a recursive function called main has been written into ML language. This function receives two lists as an input, and adds it to the list as a product on the basis of global ID. In this function, the edge of transition drawn from "synchronization" to "synchronized vars global" is considered.

```
fun max ((a,b,c,d,e,f)::L1) []= [(a,b,c,d,e,f)]|
max [] ((a,b,c,d,e,f)::L2) = [(a,b,c,d,e,f)]|
max ((a,b,c,d,e,f)::L1) ((a1,b1,c1,d1,e1,f1)::L2) =
if e>=e1 then
[(a,b,c,d,e,f)] else
[(a1,b1,c1,d1,e1,f1);]
```

In fact, the task of max function is to sort the commands to execute each one. In the model, a set of closet, type of int string, has been defined on the basis of defined requirements. Also, a list of products has been defined for places, and they have been shown in the following sections.

```
colset ID=INT;
colset OP=STRING;
colset VR=STRING;
colset VL=STRING;
colset GID=INT;
colset PID=INT;
colset INS=product ID*OP*VR*VL*GID*PID;
colset Ins_list=list INS;
var id,id1:ID;
var op1,op1_1:OP;
var vr,vr1:VR;
var vl,vl1:VL;
var gid,gid1:GID;
var L,A,A1,B,C,D:Ins_list;
var S:STRING;
var Times:INT;
var pid,pid1:PID;
```

It should be mentioned that all closets have been defined to determine data type of values, and to create a product. They are vars used on the edges.

7. State Space Analysis

After implementing the model to prove it, state space of the model was tested by CPN TOOLS software. The following summarized results are obtained:

Statistics

```
------------------------------------------------------------------------
   State Space
        Nodes:    30768
        Arcs:     135196
        Secs:     300
        Status: Partial
   Scc Graph
        Nodes:    30768
        Arcs:     112309
        Secs:     1
 Boundedness Properties
```

```
-----------------------------------------------------------------------
   Best Upper Multi-set Bounds
       P1'Instractions 1     1`[]++ 1`[(1,"write","x","",1,1),(2,"write","x","",2,1)]++1`[(2,"write","x","",2,1)]
       P1'Reads_List 1        1`[]
       P1'Synchronized_vars_Local 1    1`[(0,"","","",0,0)]++
1`[(1,"write","x","a",1,1)]++
1`[(1,"write","x","a",3,3)]++
1`[(1,"write","x","b",1,1)]++
1`[(1,"write","x","b",3,3)]++
1`[(2,"write","x","a",2,1)]++
1`[(2,"write","x","b",2,1)]
       P1'Value 1              1`"a"++              1`"b"
-----------------------------------------------------------------------
   Dead Markings
       11676 [30768, 30767, 30766, 30765, 30764, ...]
   Dead Transition Instances
       None
   Live Transition Instances
       None
  Fairness Properties
-----------------------------------------------------------------------
    No infinite occurrence sequences.
```

According to the obtained results, state space of the model involves 30768 nodes and 135196 edges. State space of CPN is a display mode of bottleneck. In this model, it has been shown that bottleneck has not occurred in nodes. It is obvious, all commands have been appropriately and completely executed, and it conveys the appropriate performance of the model. It should be mentioned that automatic analysis of the system modes is performed by CPN TOOLS simulator and on the basis of computational tree log (CLT) [22].

8. Conclusion

CPN TOOLS simulator is a strong simulator for formal modeling of distributed and non-deterministic systems. In this research, a new hierarchical model has been presented in the CPN-Tools modeling environment for the first time. In this model, weak consistency has been presented in distributed systems to keep the consistency of data among several processes of the distributed systems. The presented model has created a state by using the ML functions, and the weak consistency has been surely followed by all processes. In this model, the states are not rejected at all. Finally, state space analysis is performed by using the CLT part and the CPN-Tools simulator. It has been shown that the model lacks any bottleneck, so the model has been mathematically proved.

References

[1] Herlihy, M., & Wing, J. (1987). Axioms for concurrent objects. *Proceedings of the 14th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages* (pp. 13-26).

[2] Mustaque, A., James, E., Prince, K., & Phillip, W. (1995). Causal memory: Definitions, implementation, and programming. *Distributed Computing, 9,* 37-49.

[3] Peter, B., Joseph, M., & Hellerstein, I. (2013). Bolt-on causal consistency. *Proceedings of the annual Symposium on Cloud Computing.*

[4] Jiaqing, D., Amitabha, R., & Willy, Z. (2013). Orbe: Scalable causal consistency using dependency matrices and physical clocks. *Proceedings of the 4th annual Symposium on Cloud Computing.*

[5] Bailis, P., Fekete, A., & Ghodsi, A. (2012). The potential dangers of causal consistency and an explicit solution. *Proceedings of the 3rd ACM Symposium on Cloud Computing.*

[6] Saeedvand, S., Abbaszadeh, M., & Ansaroudi, F. (2015). Modelling causal consistency for distributed systems using hierarchical coloured petri net. *Indian Journal of Science and Technology, 8(35).*

[7] Lamport, L. (1979). How to make a multiprocessor computer that correctly executes multiprocess programs. *IEEE Trans Comput,* 690-691.

[8] Dubois, M., Scheurich, C., & Briggs, F. (1986). Memory access buffering in multiprocessors. *Proceedings of the 13th Annual International Symposium on Computer Architecture* (pp. 434-442).

[9] Alglave, J. (2016). Simulation and invariance for weak consistency. *Rival X (ed) Static Analysis, 9837.* Berlin: Springer.

[10] Abadi, D. (2012). Consistency tradeoffs in modern distributed database system design: CAP is only part of the story. *Computer, 45(2),* 37-42.

[11] Jensen, K. (1992). Coloured petri nets basic concepts, analysis methods and practical use. *Basic Concepts of Monographs in Theoretical Computer Science Springer*.

[12] Jensen, K. (1997). Coloured petri nets basic concepts, analysis methods and practical use. *Practical Use of Monographs in Theoretical Computer Science.*

[13] Jensen, K. (1994). Coloured petri nets. basic concepts, analysis methods and practical use. *Analysis Methods of Monographs in Theoretical Computer Science.*

[14] Paulson, L. (1996). *ML for the Working Programmer* (2nd ed.). NY, USA: Cambridge University Press.

[15] Gregor, V., & Bochmann, S. (2013). Distributed b-tree with weak consistency. *Networked Systems, cture Notes in Computer Science 7853,* 159-174.

[16] Zhen, C. (2010). Weak consistency of wavelet estimator for p-mixing errors. *Scientia Magna, 6(4),* 70-74.

[17] Bailis, P., Venkataraman, S., Franklin, M., & Hellerstein, J. (2013). Berkeley ISU PBS at work: Advancing data management with consistency metric. *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data* (pp. 1113-1116).

[18] Aalst, W., & Stahl, C. (2011). Modeling business processes: A petri net-oriented approach. *Information Systems.*

[19] CPN Tools. CPN Tools Download. Retrieved from http://cpntools.org/download

[20] Pashazadeh, S., & Saeedvand, S. (2014). Modelling of walking humanoid robot with capability of floor detection and dynamic balancing using colored petri net. *International Journal in Foundations of Computer Science & Technology (IJFCST), 4 (2).*

[21] Sarita, V., & Adve, M. (1990). Weak ordering-A new definition. *Proceedings of the 17th Annual International Symposium on Computer Architecture.*

[22] Baier, C., & Katoen, J-P. (2008). *Principles of Model Checking.*

Mortaza Abbaszadeh received his BSc degree in computer software engineering from Shabestar IAU, he received his M.Sc. degree in computer software engineering in Faculty of Electrical and Computer Engineering in University of Qazvin IAU in Iran. He is working as a lecturer in Islamic Azad University of Iran since 2004. His research interest includes: robotic, artificial intelligence and web mining.

Saeed Saeedvand received his BSc degree in computer software engineering from Islamic Azad University in 2011, he received his M.Sc. in electrical and computer engineering in Department, University of Tabriz in 2014 and currently, he is Ph.D. student at University of Tabriz. He is working as a lecturer in University of Tabriz and Islamic Azad University of Iran. Also he is working in humanoid robotic teams at University of Tabriz and Islamic Azad University as team leader. He worked on humanoid adult-size and kid-size robots since 2009. His research interest includes artificial intelligence, robotic, modeling, control and cloud computing.