UDC 004.5:519.876.2

M. Iwaniak, W. Khadzhynov
Technical University of Koszalin
Department of Electronics & Informatics
ul. Śniadeckich 2, 75-453 Koszalin, Polska
hadginov@ie.tu.koszalin.pl

Distributed Transactions Modeling with the Use of Petri Nets

The attempt of using ordinary Petri Net to model and study Three-Phase Commit protocol (3PC) is presented. A brief overview of Petri Nets is introduced. The nature of typical and distributed transactions are explained. 3PC protocol actions are described. The Petri Net of 3PCprotocol followed by reachability analysis and study of the net properties is presented.

***Key words**: Petri Net, distributed transactions, 3PC, Three-Phase Commit protocol.*

1. Introduction

The theory and the practice of distributed database is a very complex issue. During many years of studies the standard that would be helpful in projecting and implementation of systems based on distributed database was not developed. There is especially a lack of operative standard for heterogonous distributed database.

The theory of Petri Net is used in modeling and analyzing parallel processes. The structure and function of Petri Net could described in algebraic form and processed with the use of numerical methods.

The solution allowing projecting, modeling and verifying processes in distributed database with the use of Petri Net is searched. Such a net might serve for supervising the flow of data in distributed database nodes or for generating configurations that allow the realization of projected processes.

This work presents the attempt of usage Petri Net for showing the function of 3PC protocol. We try to answer the question: whether the ordinary Petri Net is an adequate tool for modeling such a complex process as a commitment of distributed transaction is.

2. Petri Nets

The mathematical theory of Petri Nets was created by Carl Adam Petri. It has wide field of usage in analysis and modeling.

Most commonly described is The Ordinary Petri Network, also named as network of class position/transition. It's graphical representation is bipartite graph which contains two types of nodes connected by arcs. Nodes of the net are:

— places (positions) — represented by circles;

— transitions () — represented by rectangles.

Petri Net can be expressed as the triplet $N = (P, T, D)$ where:

— P is a collection of places $|P| = m$;

— T is a collection of transitions $|T| = n$;

— D is incidence matrix of dimension $m \times n$. This matrix describes relations between collections of places and transitions;

— D^- is pre-incidence matrix of dimension $m \times n$, it contains elements $d_{ij}^- = w(i, j)$ which describe the weight of transition j input arc, that is the arc directly connecting place i with transition j ($P \times T \rightarrow N$);

— D^+ is post-incidence matrix of dimension $m \times n$, it contains elements $d_{ij}^+ = w(i, j)$ which describe the weight of transition j out arc, that is the arc directly connecting transition j with place i with ($P \times T \rightarrow N$);

— $D = [D^+ - D^-]$ is incidence matrix of dimension $m \times n$, this matrix is created by subtracting pre-incidence matrix from post-incidence matrix, it contains elements $d_{ij} = d_{ij}^+ - d_{ij}^-$. These elements come from subtracting weights of input arc from weights of output arcs.

2.1. Marking of net and firing of transitions

Marked Petri Net can be described by four-tuple $PN = (P, T, D, M_0)$, where M_0: $P \rightarrow \{0,1,2,..\}$ is an initial marking of the net that defines the token distribution in the network places.

In Petri Net with given initial marking can occur dynamic events. If the input places of given transition have an amount of tokens equal to input arc weight then the transition can be fired. After firing of the transition from all input places the tokens are taken away and new tokens are inserted into the output places. The amounts of tokens taken away from each input place and inserted into output places are equal to weights of given input and output arcs. The process is presented on Fig. 1.

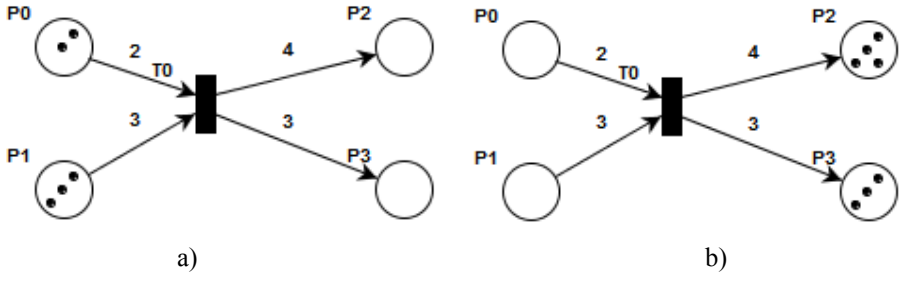

Fig. 1. Transition: a) transition ready to be fired; b) transition after firing

Depending on the initial marking the reachability set for given Petri Net can differ. The sets of reachable markings are created and presented with usage of a directed graph

called a reachability tree. A reachability tree represents one by another both firing of transitions and marking changes. Each sequence of firings and marking changes is called the execution of the net.

A reachability tree contains such elements as:

— the root — representing initial marking,

— nodes — representing each reachable marking,

— arcs — representing fired transition,

— leaves — representing final or repeated markings.

A reachability tree can be prepared by analysis of following transitions firings. This analysis can be simplified by algebraic properties of Petri Net. Each column of the incidence matrix describes a change in marking after a firing of given transition. Each marking can be computed with usage of the following formula

$$M_k = M_{k-1} + e[t_j]D,$$

where $k = 1,2,3,\ldots$ and $e[t_j]$ a firing vector containing a digit 1 in position corresponding j transition.

2.2. Properties of Petri Net

Boundness — the net is bounded, when for each place we can determine a finite number of tokens appearing in given place. This means that collection of possible markings is finite and can be searched.

Safeness is a detailed property of boundness. The place is safe if for any given marking the number of tokens in place is 0 or 1. If this condition is valid for every place in the net then the net is safe. This property is important when Petri Net is projected for hardware implementation, where place will be replaced by transistor capable of having 0 or 1 value.

Conflictuality — a conflict of transition firing can occur when for given marking more than one transitions is enabled. It can also happen if firing of one transition disables other already enabled transition [2].

Conservativness — the net is conservative if sum of tokens in every marking is constant. This can be proven only for simple nets.

2.3. Timed Petri Net

In the Timed Petri Net we can define some time delay. If given transition gets ready (required number of tokens for each place is preset), the transition firing will happen only after the delay. For two timed transitions sharing the same input place a specific situation can occur. If one gets fired it can take away the shared tokens and the another might not be able to be fired.

2.4. Coloured Petri Net

Coloured Petri Net is a graph containing places, transitions and arcs in which the token stores value of given type. For tokens of given type specific colors are assigned.

Places of Petri Net can store tokens of many colors. Given place however must allow the specific color to be stored in it. Colours separately are assigned to arcs weight. For example transition can be enabled after number of each tokens colours are equal to colours weights defined for the arc.

3. Processing transactions in database

The digital collection of properly organized data is called database. The stored data usually model real objects. Database Management System in short DBMS is responsible for accuracy, safety and effectiveness of the access to stored data. The structure of database is too complex to work with it without the usage of DBMS. Typically DBMS is responsible for providing data access service. Through local operating system the DBMS cares of safe data storing and mediates in all operations executed by users.

Stored data are fully useful only if they are valid and consistent. Each data change takes a whole database into entirely new database state. Many data changes can happen in one unit of time and some of them are erroneous. The transaction mechanism cares for validness and consistence of an actual database state.

A transaction is an operation that correctly executed guaranties database consistence. Transactions can consist of many operations of reading and writing the data. The changes made by the transaction must be done as a whole or none.

The ACID properties are the ones that guaranty proper transactions processing. These properties are:

— Atomicity — states that each transaction must be executed as whole or not at all;

— Consistency — states that after the transaction is finished the database system will remain consistent. The changes made by the transaction will not be stored in database if during the commitment occurs an error like integrity bounds violation or any other that guaranties a data validation;

— Isolation — level of isolation determines the possibility of mutual data reading by transactions proceeded concurrently. By the selected level of isolation we determine types of possible anomalies that may occur during transactions execution;

— Durability — states that database system can startup after failure and provide consistent, not damaged and actual data stored in bounds of committed transactions.

DBMS registers all stages of transaction processing in a transaction log. During database recovery only committed transactions are recovered, because only those can guaranty receiving the consistent state of database.

In a single database during processing of transactions many anomalies can occur, like blocking, conflicts or deadlocks. A single DMBS upon its configuration and capabilities tries to avoid or removes occurring anomalies.

4. Transactions processing in distributed database

Distributed database is a collection of many logically bounded databases connected via computer network. Variety of distributed database types is summarized in [1].

Short time transactions are advised by DBMS. From application or user points of view execution of transaction in distributed system should not be different from traditional system.

Standalone database has implemented program called transaction manager shortly TM, that cares for proper database state management and for communication with clients. In distributed environment the additional communication is happening between TMs. One of TMs serves a role of Coordinator, the others take a role of Cohorts. A way of communication between Coordinator and Cohorts are defined by distributed commit protocol. The classical example of such protocol is the one named Two-Phase Commit. This protocol has two phases: phase of voting and phase of committing. The voting phase when Coordinator sends question to its Cohors, asking if they are ready to commit distributed transaction. Returning communicates are called votes. These votes can be: vote-commit or vote-abort. After collecting votes Coordinator makes its decision. If there was any vote for aborting or any of Cohorts did not respond at all, the coordinator will make the decision of global abort of distributed transaction. If all of the votes were positive its decision will be global commit.

Making of the decision only when all of the votes are for commit is the main way for assure atomicity into distributed database.

4.1. Three Phase Commit protocol for distributed transaction

On Fig. 2 the algorithm of 3PC protocol is presented. Fig. 2 was prepared upon [1]. The circles represent states of Coordinator and Cohort processes. The rectangles represent operations of logging received messages and made decisions into system log. Arrows represent the flow of messages and control. Algorithm looks as follows.

1. The coordinator creates log entry with information <begin_commit> and sends <prepare> message to Cohorts. Coordinator now waits for Chorts votes. Coordinator state is <WAIT>.

2. The cohorts decide if they are ready to commit transaction. They store their decision into log file and send <vote-commit> or <vote-abort> to Coordinator. Varing of the decision new state of Cohort process can be READY or ABORT.

3. The coordinator checks received messages from all cohorts:

a) If there was any vote-abort or some cohort did send his vote, the Coordinator makes the decision of aborting distributed transaction. Coordinator writes his decision into log file and sends <global-abort> message. Coordinator state is <ABORT>;

b) If all cohorts have responded with <vote-commit> message then Coordinator makes decision to begin second phase of commit. Coordinator writes <prepare-to-commit> into log file. It sends <prepare-to-commit> message to Cohorts. Coordinator state is <PRE-COMMIT>.

4. The cohorts write received message into the log. In case of <global-abort> Cohorts processes turn into <ABORT> state and acknowledge is send to Coordinator. In case of <prepare-to-commit> Cohorts processes turn into <PRE-COMMIT> state and sends <ready-to-commit> message to Coordinator.

5. After collecting Cohorts <ready-to-commit> messages Coordinator writes <commit> into log. It sends <global-commit> message to all Cohorts. Coordinator state is now <COMMIT>.

6. The cohorts write received message into log and send back acknowledge message. Cohorts are now in <COMMIT> state.

7. The coordinator writes <end_of_transaction> into log file.

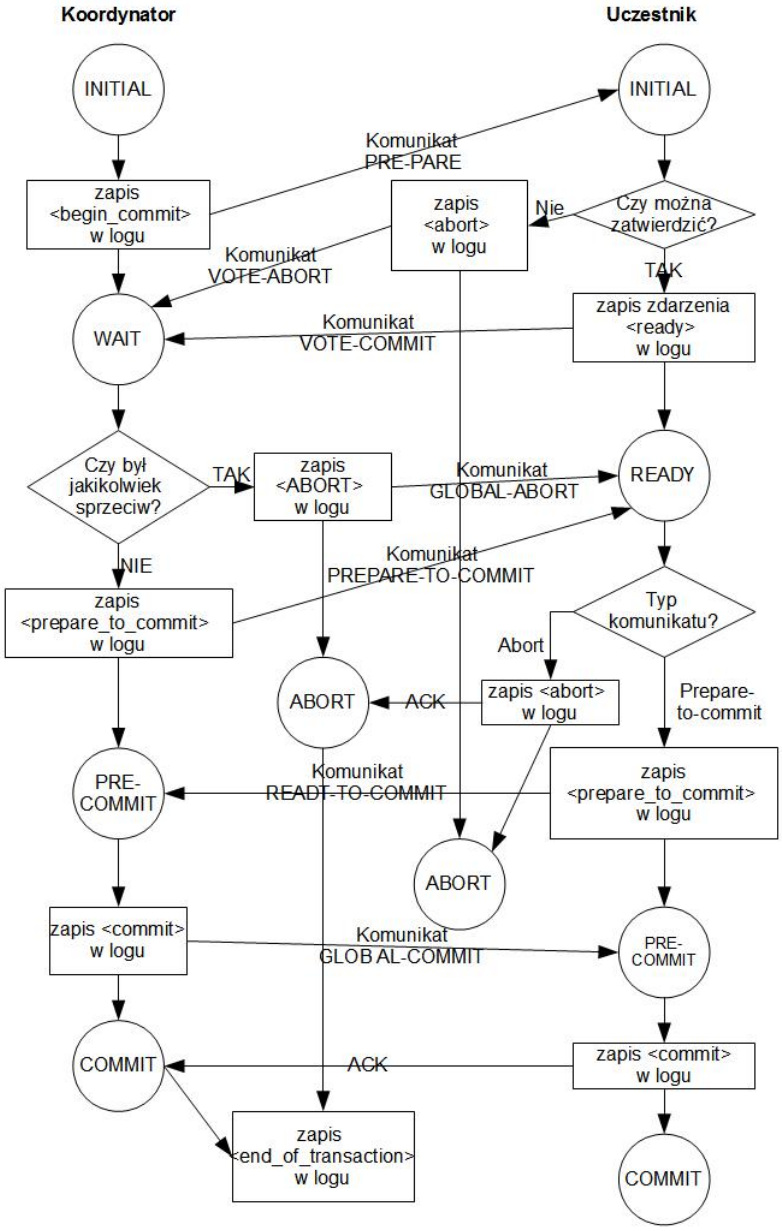

Fig. 2. Three Phase Commit protocol algorithm with one participant

5. The ordinary Petri Net model of 3PC protocol

Described in point 4.1 the 3PC protocol was introduced as Petri Net model on Fig. 3. The example was prepared for Coordinator and one Cohort. Places of Coordinator (Tab. 1) are aligned to left edge of figure. Places of Cohort (Tab. 2) are aligned to right edge of figure.

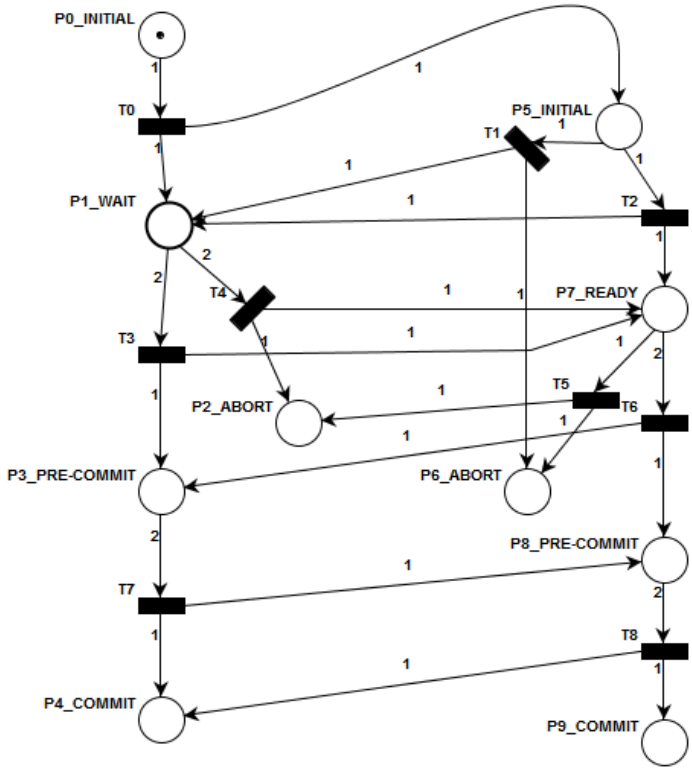

Fig. 3. Petri Net of 3PC protocol algorithm

Table 1. Places and transitions of Coordinator

Places		Transitions	
*P*0	INITIAL	*T*0	1) write to log (begin_commit) 2) message to Cohorts (prepare)
*P*1	WAIT	*T*4	1) write to log (abort) 2) message to Cohorts (global-abort)
*P*2	ABORT	*T*3	1) write to log (prepare-to-commit) 2) message to Cohorts (prepare-to-commit)
*P*3	PRE-COMMIT	*T*7	1) write to log (commit) 2) message to Cohorts (global-commit)
*P*4	COMMIT	–	–

Table 2. Places and transitions of Cohort

Places		Transitions	
*P*5	INITIAL	*T*1	1) write to log (abort) 2) message to Coordinator (vote-abort)
*P*6	ABORT	*T*2	1) write to log (ready) 2) message to Coordinator (vote-commit)
*P*7	READY	*T*5	1) write to log (abort) 2) potwierdzenie do Coordinator
*P*8	PRE-COMMIT	*T*6	1) write to log (prepare-to-commit) 2) message to Coordinator (ready-to-commit)
*P*9	COMMIT	*T*8	1) write to log (commit) 2) potwierdzenie do Coordinator

5.1. Incidence Matrixes for Petri Net

Rows of Tab. 3 contains places P_0, P_1, P_2, P_3, P_4 which represent the states of Co-ordinator and places P_5, P_6, P_7, P_8, P_9 which represent states of the Cohort. Columns $t_0 - t_9$ represent transitions. The pre-insidence matrix contains weights of input arcs of given transition. In case of no arc between place i and transition j we enter digit 0. This matrix is marked as $[D^-]$.

Tabel 3. Pre-Incidence Matrix

D^-		t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
P_0	INITIAL	1	0	0	0	0	0	0	0	0
P_1	WAIT	0	0	0	2	2	0	0	0	0
P_2	ABORT	0	0	0	0	0	0	0	0	0
P_3	PRE-COMMIT	0	0	0	0	0	0	0	2	0
P_4	COMMIT	0	0	0	0	0	0	0	0	0
P_5	INITIAL	0	1	1	0	0	0	0	0	0
P_6	ABORT	0	0	0	0	0	0	0	0	0
P_7	READY	0	0	0	0	0	1	2	0	0
P_8	PRE-COMMIT	0	0	0	0	0	0	0	0	2
P_9	COMMIT	0	0	0	0	0	0	0	0	0

Rows of Tab. 4 contain places P_0, P_1, P_2, P_3, P_4 which represent the states of Co-ordinator and places P_5, P_6, P_7, P_8, P_9 which represent states of Cohort. Columns $t_0 - t_9$ represent transitions. The post-insidence matrix contains weights of output arc of given transition. In case of no arc between transition j and place i we enter digit 0. This matrix is marked as $[D^+]$.

Tabel 4. Post-Incidence Matrix

D^+		t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
P_0	INITIAL	0	0	0	0	0	0	0	0	0
P_1	WAIT	1	1	1	0	0	0	0	0	0
P_2	ABORT	0	0	0	0	1	1	0	0	0
P_3	PRE-COMMIT	0	0	0	1	0	0	1	0	0
P_4	COMMIT	0	0	0	0	0	0	0	1	1
P_5	INITIAL	1	0	0	0	0	0	0	0	0
P_6	ABORT	0	1	0	0	0	1	0	0	0
P_7	READY	0	0	1	1	1	0	0	0	0
P_8	PRE-COMMIT	0	0	0	0	0	0	1	1	0
P_9	COMMIT	0	0	0	0	0	0	0	0	1

Combined insidence matrix (Tab. 5) is computed by subtracting pre-incidence matrix from post-incidence matrix $[D^+ - D^-]$. It describes dynamic changes that occur in given Petri Net. It contains information about amount of tokens added and removed in every place after the t_j transitions is fired.

Table 5. Combined Incidence Matrix

D		t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
P_0	INITIAL	–1	0	0	0	0	0	0	0	0
P_1	WAIT	1	1	1	–2	-2	0	0	0	0
P_2	ABORT	0	0	0	0	1	1	0	0	0
P_3	PRE-COMMIT	0	0	0	1	0	0	1	–2	0
P_4	COMMIT	0	0	0	0	0	0	0	1	1
P_5	INITIAL	1	–1	–1	0	0	0	0	0	0
P_6	ABORT	0	1	0	0	0	1	0	0	0
P_7	READY	0	0	1	1	1	–1	–2	0	0
P_8	PRE-COMMIT	0	0	0	0	0	0	1	1	–2
P_9	COMMIT	0	0	0	0	0	0	0	0	1

6. Reachability analysis

Initial marking is represented in the form of vector M_0 = [10000000000] (Fig. 4). For given marking we check which transitions will became enabled. For instance token in the place $P1$ will cause firing of transitions t_0.

Algebraic pattern for designate markings after firing j transition is as follows:

$$M_k = M_{k\text{-}1} + e[\,t_j\,]D,$$

where M_k — new marking; $M_{k\text{-}1}$ — previous marking; for k = 1 it is initial marking; $e[\,t_j\,]$ — column vector of size i with digit 1 on index that corresponds fired transition; D — the incidence matrix.

For given marking M_0 we want to designate marking M_1 by firing of transition t_0:

$$e[t_0]*D = [–1\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0].$$

$$M_0 = [10000000000].$$

After addition of two vectors we will have following marking of Petri Net:

$$M_1 = [0100010000].$$

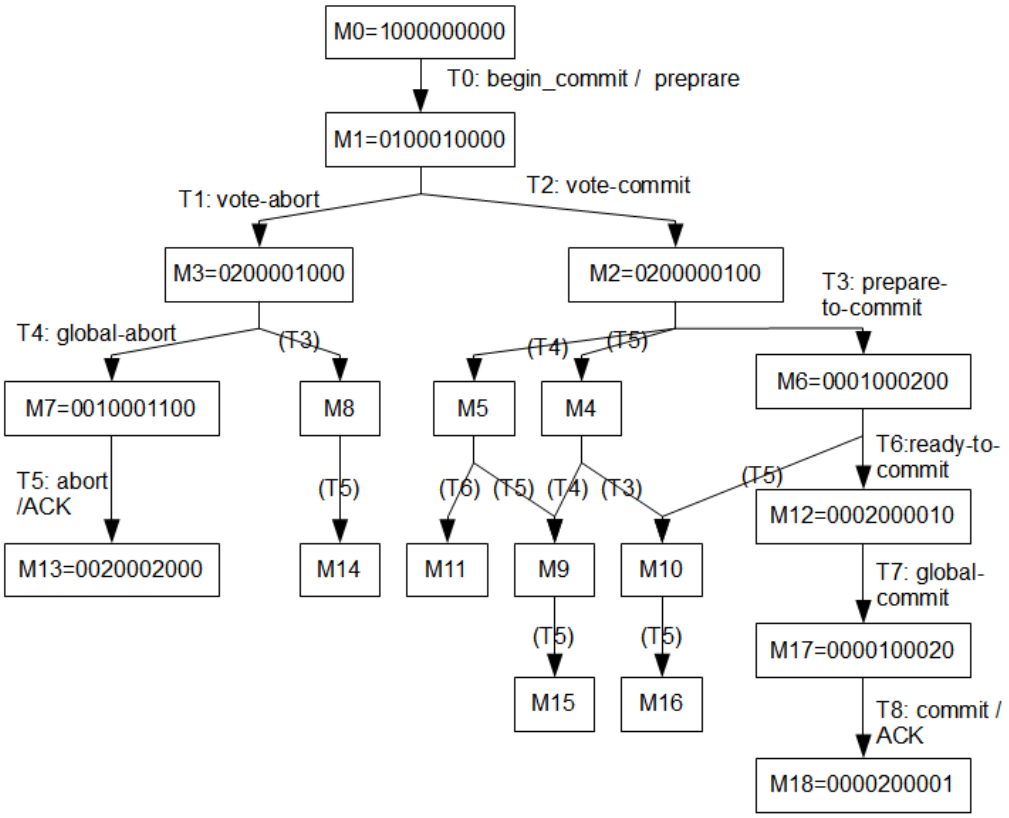

Fig. 4. Reachability tree for Petri net from Fig. 3
(undesirable states are shown without markings)

6.1. Properties of Petri Net model

Boundness — maximal amount of tokens in places is 2 therefore the net is bounded.

Safeness — there are markings with places where quantity of tokens is 2 therefore the net is not safe.

Conflictuality — in the presented model the conflicts take place in the Petri Net places where decision must be made. Accurate amount of tokens in places $P1$, $P2$ and $P4$ make all output transitions enabled ($P1(T1, T2)$, $P5(T3, T4)$, $P7(T5, T6)$). In place $P5$ Cohort process in case of error must send abort vote or in case of being ready it must send vote for commit.

Conservativnes — presented model is complex, the property cannot be validated.

7. Resume

In this work Petri Net model of 3PC protocol was presented. With usage of incidence matrixes the reachability tree was created. Undesireble but reachable states were identified on reachability tree.

During studies of presented Petri Net model a number of limitations and problems were pointed. Most of them come from the Ordinary Petri Network limitations. We must use higher level Petri Net in further research. Applying Coloured Petri Net should

resolve the problem of decision making in certain markings. It would help reducing number of undesirable states in the analysis of reachability.

1. *M. Tamer Özsu.* Principles of Distributed Database Systems / M. Tamer Özsu, Patrick Valduriez // Springer. — III ed. — 2011.

2. *Banaszak Z.* Procesy Współbieżne: Modele Efektywności Funkcjonowania. Rozdział 2 «Modele sieci Petriego» / Z. Banaszak, P. Majdzik, R. Wójcik. — Politechnika Koszalińska, Koszalin 2011. — Str. 93–143.

3. *Bidyut Biman Sarkar.* Transaction Management for Distributed Database using Petri Nets / Bidyut Biman Sarkar, Nabendu Chaki // International Journal of Computer Information Systems and Industrial Management Applications (IJCISIM). — 2010. — Vol. 2. — P. 069–076. — ISSN: 2150-7988.

4. *Bidyut Biman Sarkar.* Virtual Data Warehouse Modeling Using Petri Nets for Distributed Decision Making. doi:10.4156/jcit / Bidyut Biman Sarkar, Nabendu Chaki. — 2011. — Vol. 5. — Is. 5.1.

Received 01.07.2012